LOCOMOTION AND CONTROL OF A MODULAR SNAKE LIKE ROBOT

A THESIS SUBMITTED TO
THE GRADUATE SCHOOL OF NATURAL AND APPLIED SCIENCES
OF
MIDDLE EAST TECHNICAL UNIVERSITY

BY

ERGİN KURTULMUŞ

IN PARTIAL FULFILLMENT OF THE REQUIREMENTS
FOR
THE DEGREE OF MASTER OF SCIENCE
IN
MECHANICAL ENGINEERING

SEPTEMBER 2010

Approval of the thesis:

LOCOMOTION AND CONTROL OF A MODULAR SNAKE LIKE ROBOT

submitted by **ERGİN KURTULMUŞ** in partial fulfillment of the requirements for the degree of **Master of Science in Mechanical Engineering Department, Middle East Technical University** by,

Prof. Dr. Canan Özgen
Dean, Graduate School of **Natural and Applied Sciences** _____

Prof. Dr. Süha Oral
Head of Department, **Mechanical Engineering** _____

Prof. Dr. Reşit Soylu
Supervisor, **Mechanical Engineering Dept., METU** _____

Examining Committee Members:

Prof. Dr. M. Kemal Özgören
Mechanical Engineering Dept., METU _____

Prof. Dr. Reşit Soylu
Mechanical Engineering Dept., METU _____

Prof. Dr. S. Kemal İder
Mechanical Engineering Dept., METU _____

Asst. Prof. Dr. Yiğit Yazıcıoğlu
Mechanical Engineering Dept., METU _____

Prof. Dr. Kemal Leblebicioğlu
Electrical and Electronics Engineering Dept., METU _____

Date: 13.09.2010

I hereby declare that all information in this document has been obtained and presented in accordance with academic rules and ethical conduct. I also declare that, as required by these rules and conduct, I have fully cited and referenced all material and results that are not original to this work.

Name, Last name: Ergin KURTULMUŞ

Signature :

ABSTRACT

LOCOMOTION AND CONTROL OF A MODULAR
SNAKE LIKE ROBOT

Kurtulmuş, Ergin

M.Sc., Department of Mechanical Engineering

Supervisor: Prof. Dr. Reşit Soylu

September 2010, 184 Pages

In recent years, there has been a significant increase in the interest for snake like modular robots due to their superior locomotion capabilities in terms of versatility, adaptability and scalability. Passive wheeled planar snake like robots are a major category and they are being actively researched.

Due to the nonholonomic constraints imposed on them, certain configurations lead to the singularity which must be avoided at all costs. Furthermore, it is vital to generate a locomotion pattern such that they can track a wide range of trajectories. All of these objectives must be accomplished smoothly and in an energy efficient manner. Studies indicate that meeting all of these requirements is a challenging problem.

In this study, a novel form of the serpenoid curve is proposed in order to make the robot track arbitrary paths. A controller has been designed using the feedback linearization method. Afterwards, a new performance measure,

considering both the efficiency and sustainability of the locomotion, has been proposed to evaluate the locomotion. Optimal parameters for the proposed serpenoid curve and the linear controller have been determined for efficient locomotion by running series of simulations. Relations between the locomotion performance, locomotion speed and eigenvalues of the linear controller have been demonstrated. Simulation results show striking differences between the locomotion by using the proposed serpenoid curve with optimal parameters and the locomotion by purely tracking a given path. Obtained results also indicate that the aforementioned requirements are met successfully and confirm the validity and consistency of the proposed performance measure.

Keywords: Locomotion control, snake like robot, nonholonomic constraints, feedback linearization, serpenoid curve, optimal locomotion, trajectory tracking, singularity avoidance.

ÖZ

YILAN BENZERİ MODÜLER BİR ROBOTUN
HAREKET VE KONTROLÜ

Kurtulmuş, Ergin

Yüksek Lisans., Makine Mühendisliği Bölümü

Tez Yöneticisi: Prof. Dr. Reşit Soylu

Eylül 2010, 184 Sayfa

Son yıllarda, çok yönlülük, uyum sağlayabilme, ve uyarlanabilirlik bağlamında üstün hareket yeteneklerinden dolayı yılan benzeri modüler robotlara olan ilgide önemli bir artış vardır. Pasif tekerlekli, düzlemsel, yılan benzeri robotlar ana kategorilerden birini oluşturmakta ve aktif şekilde araştırılmaktadırlar.

Bu tür robotların üzerine empoze edilen holonomik olmayan kısıtlar nedeniyle bazı konfigürasyonlar her ne pahasına olursa olsun kaçınılması gereken tekillik durumuna yol açabilmektedirler. Dahası, çok çeşitli yörüngeleri takip edebilmelerini sağlayacak bir hareket paterni üretmek oldukça önemlidir. Tüm bunlar yumuşak geçişli ve enerji etkin bir şekilde başarılmalıdır. Bir çok çalışma tüm bu gereksinimleri sağlamanın zorlu bir problem olduğunu göstermektedir.

Rastgele seçilmiş yolları takip edebilmek amacıyla yeni tür bir serpenoid eğri tanımlanmıştır. Geri besleme doğrusallaştırması yöntemiyle bir kontrolcü tasarlanmıştır. Ardından, hareketi değerlendirmek amacıyla, hareketin hem

etkinliğini hem de sürdürülebilirliğini göz önüne alan yeni bir performans ölçütü önerilmiştir. Çoklu simulasyonlar koşturularak, önerilen serpenoid eğrisi ve doğrusal kontrolcü için etkin hareketi sağlayacak optimal parametreler belirlenmiştir. Hareket performansı, hareket hızı ve doğrusal kontrolcünün özdeğerleri arasındaki ilişkiler gösterilmiştir. Simulasyon sonuçları, optimal parametrelere sahip önerilen serpenoid eğrisiyle yapılan hareket ile yalın bir şekilde verilen yolu takip eden hareket arasındaki çarpıcı farkları göstermektedir. Elde edilen sonuçlar, bahsedilen gereksinimlerin başarıyla sağlandığını ve önerilen performans ölçütünün geçerliliğini ve tutarlılığını göstermektedir.

Anahtar Kelimeler: Hareket denetimi, yılan benzeri robot, holonomik olmayan kısıtlar, geri besleme doğrusallaştırması, serpenoid eğrisi, optimal hareket, yörünge takibi, tekillikten kaçınma.

To my family and the beloved ones

ACKNOWLEDGEMENTS

The author would like to thank his supervisor Prof. Dr. Reşit Soylu for his guidance, patience, encouragement and strong trust which made this study possible.

I am deeply grateful to my family for their unrequited support and love, anytime, anywhere.

I also profoundly appreciate the help of my friends Varlık Kılıç, Emre Demirocak and Meltem Yıldız whether in the form of casual conversations or in the form of brainstorming. Their support and confidence in me were particularly valuable.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

NOMENCLATURE

m	Mass of a single module of the modular robot.
J	Moment of inertia of a single module about its z-axis through the center of gravity.
x_i	x-coordinate of the center of gravity of the i^{th} module measured in the absolute frame.
y_i	y-coordinate of the center of gravity of the i^{th} module measured in the absolute frame.
θ_i	Orientation angle of the i^{th} module measured in the absolute frame.
φ_i	Joint angle between the $(i+1)^{th}$ and i^{th} module.
l	Distance between the center of gravity of a module and one of the identical joints.
x_h	x-coordinate of the robot head measured in the absolute frame.
y_h	y-coordinate of the robot head measured in the absolute frame.
x_r	x-coordinate of the robot head reference trajectory (modified serpenoid curve) measured in the absolute frame.
y_r	y-coordinate of the robot head reference trajectory (modified serpenoid curve) measured in the absolute frame.
x_t	x-coordinate of the robot head task trajectory measured in the absolute frame.

y_t	y-coordinate of the robot head task trajectory measured in the absolute frame.
e_t^x	Difference between the x-coordinates of actual robot head position and reference task trajectory.
e_t^y	Difference between the x-coordinates of actual robot head position and reference task trajectory.
e_r^x	Difference between the x-coordinates of actual robot head position and reference modified serpenoid curve.
e_r^y	Difference between the x-coordinates of actual robot head position and reference modified serpenoid curve.
s_{act}	Curve length of the actual path of the robot head between starting and end points of the locomotion.
p_{eff}	Effective locomotion displacement of the robot head during its locomotion.
f_i^R	Total rolling (tangential) friction force acting on the i^{th} module.
f_i^N	Total normal (lateral) friction force acting on the i^{th} module.
$f_{\max}^N$	Maximum normal (lateral) friction force that the ground can apply on a module.
μ_N	Coefficient of friction in the normal direction to the respective module.
μ_R	Coefficient of friction in the tangential direction to the respective module.

d_φ	Damping coefficient in the revolute joints.
u_i	Control torque applied by the i^{th} motor situated in between the i^{th} and $(i+1)^{th}$ modules.
α	Initial winding angle of the serpenoid curve.
K_n	Number of S-shapes in one period of the serpenoid curve.
L	Total effective length of the robot
s	Path length along the serpenoid curve from the beginning to the current point of the curve.
v_i	Synthetic control input signal for the i^{th} actuator (motor).
λ_i	Lagrangian multiplier corresponding the i^{th} module.
γ	Compensation angle for the modified serpenoid curve.
P_i	Rotational power supplied by the i^{th} actuator during the locomotion.
E	Total energy consumption of the robot for a specific time of locomotion.
v	Tracing speed of the robot head along its trajectory.
H	Performance measure related to the locomotion of the robot.
α^*	Optimal value of α.
K_n^*	Optimal value of K_n.
ω_x	Natural frequency of the linear system dynamics in the x direction.
ω_y	Natural frequency of the linear system dynamics in the y direction

CHAPTER 1

INTRODUCTION

1.1 Inspiration From Biological Snakes

The nature, through hundreds of millions of years of evolution, has granted the limbless animals rather interesting gait abilities unique to them. These unique abilities have proven to be highly effective in the harsh competition environment of the earth where the principal of "survival of the fittest" has been and is the most vital phenomenon behind the dynamics of evolution. Crawling creatures like snakes utilize locomotion patterns and gait mechanics fundamentally different from those that walk, swim, hop or fly which are abundantly seen in nature [1]. This fact originates from the lack of limbs "as we know them". Depending on certain conditions, the crawling locomotion can be as efficient as legged locomotion [2]. Besides, snakes in nature have extreme adaptability to the environment: they can continue locomotion on rough terrains, narrow passages, on steep cliffs, muddy ponds, in lakes, in the seas, in the tropical areas as well as in vast deserts [3]. Snakes are well adapted to moving on unstable surfaces like sand dunes or on marshlands because they can distribute their weight uniformly over the whole body and hence can propel themselves in a kinematically stable gait during the locomotion [4]. Their ability to survive originates from this very extreme adaptability, which in turn originates from their redundant body morphology. This explains why hundreds of species of snakes are existent in almost every habitable part of the world. Having about 2900 species in total, their size varies from 10 cm long thread snake to pythons and anacondas of up to

7.6 meters in length [5]. Snake, which is a vertebrate animal, depending on the species, has about 100-400 vertebrae in its backbone (largest number of any animal). The interconnections in the snake vertebrae are one of the most complex ones in nature. Despite the fact that only simple relative angular rotations are possible between two adjacent vertebrae, accumulation of these small rotations is able to generate large angular displacements.

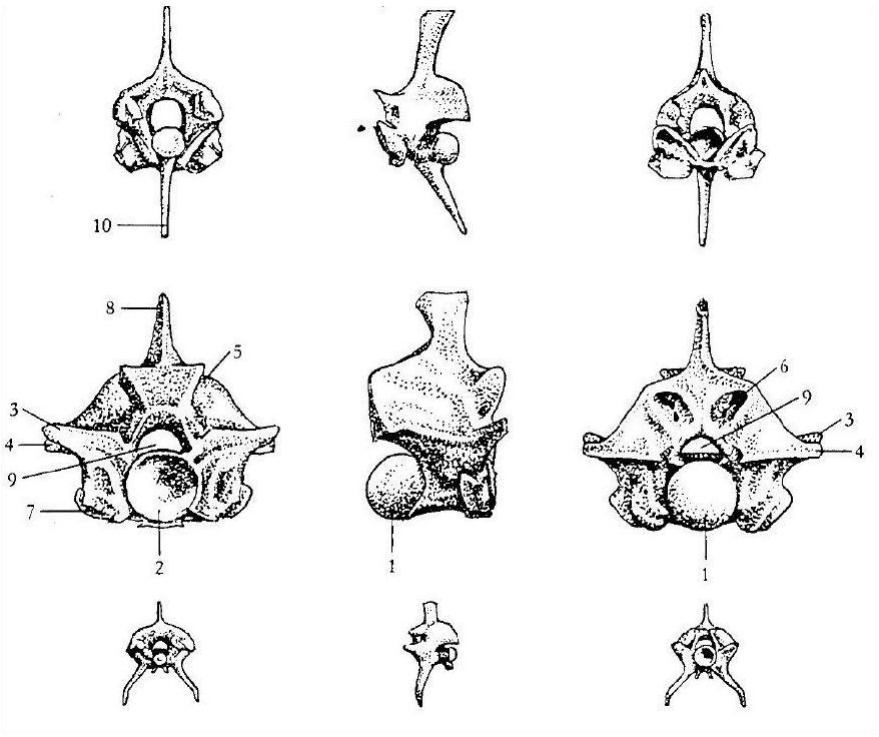


Figure 1.1: Illustration of Snake Vertebrae. Adapted from [6]

The vertebrae of the snake consist of ball and socket joints with included projections which allow certain amount of lateral and ventral angular displacements about 10-20 degrees and 2-3 degrees respectively [6], [7]. An illustration of the snake vertebrae can be seen in Figure 1.1. This remarkable vertebral structure of the snake effectively reduces the possible twists in the spinal cord during locomotion.

Snakes compose of repeated structures of joints and membranes. Unlike the other

limbed vertebrae in nature, they have only three distinct skeleton parts: skull, vertebrae and ribs [6], [7]. This feature makes them physiologically quite simple but at the same time robust to many sort of unexpected events and fluctuations in their surroundings. In case a part of the snake body fails to function for some reason, the whole body is able to continue to locomote. The redundancy of the snake body makes them quite versatile and scalable. Hence, they can utilize their bodies both as locomotors and manipulators [8]. This phenomenon can be observed when a snake moves through leafs approaching its prey and curling and wrapping around it tightly to eventually clench it.

An interesting feature of the snake body is its skin. Snake skin is composed of small scales with adjustable orientations. Each scale has its own muscular unit in

Figure 1.2: Ventral scales on the skin of a Python. Public Image

the neuromuscular system to control the orientation [9]. In Figure 1.2, rows of scales of a Python can be seen. Snakes use their scales to reduce the friction in the

preferred direction during the locomotion while some species can use the edges to grip branches. The basics of the propulsion of snakes on the ground are heavily dependent on the frictional anisotropy of their scales. According to the experiments done in [1], overlapping belly (ventral) scales of snakes snag on ground asperities. This process provides the snake a preferred direction of sliding on surfaces satisfying certain conditions like having sufficient roughness and compliance.

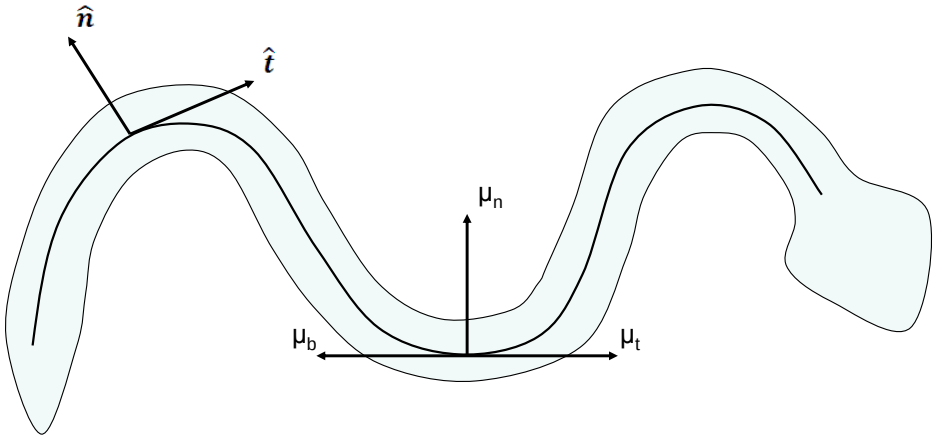


Figure 1.3: Schematic diagram of directions of frictional forces acting on a snake by the ground. Adapted from [1].

Frictional anisotropy is the notion of difference between the principal directions of frictional forces applied by the ground to the snake body during the locomotion. In Figure 1.3, μ_t, μ_n and μ_b are the sliding coefficients of friction in tangential, normal and backwards directions, respectively. Tangent and the normal unit vectors to the snake body are also depicted in Figure 1.3. Experiment results in [1] indicate $\mu_t =$

0.11 ± 0.011 when sliding forward, μ_b = 0.14 ± 0.015 when sliding backwards, μ_n = 0.20 ± 0.015 when sliding towards its flanks (sliding along the normal direction) on cloth surface using the least square estimation. These findings are mostly in agreement with the results in [10]. Furthermore, in their experiments, they observed that when the snakes are left to move on smooth surfaces they fail to generate any effective locomotion. This is attributed to the fact that, when the surface is relatively smooth, snakes cannot find any asperities for their scales to grind on and hence friction coefficients turn out to be almost independent of the direction. Thus, lack of anisotropy in the friction coefficients prevents the snake from performing displacements in the desired directions. However, snakes observed in nature are known to dynamically change their frictional interactions with the environment depending on numerous factors by adjusting their scales or by lifting their body in rather complex manners. Hence, it is not always very straight forward to assume certain coefficients of friction for them.

Despite seemingly high frictional forces imposed on them during their locomotion, it has been shown that they need comparable amount of energy to other morphologically similar animals [8], [2]. This fact can be explained by the following aspects:

- The bodies of snakes are not lifted significantly during their locomotion. Hence, they do not do considerable amount of work against the gravity
- Because of their simple repetitive body structure, they do not need to move their appendages which is the case in legged animals [2], [7].
- Despite the high frictional forces during the locomotion, direction of the translational velocity of the snakes differs considerably from the direction of the applied friction by the ground.

The last listed aspect prevents the snake from doing work against the frictional force applied by the ground. This very fact forms the core of the fundamental principal in the undulatory locomotion of snakes.

1.2 Locomotion of Snakes In Nature

There are major classes of motion patterns (gaits) of the snakes observed in nature. These are lateral undulation, concertina locomotion and sidewinding locomotion and they can be briefly outlined as follows:

1.2.1 Lateral Undulation

Lateral undulation is the locomotion gait that is used most abundantly by the snakes in nature. It is basically performed in the forms of propagating waves from the head to the tail of the snake by making use of the friction between the snake body and the ground. All of the body sections move simultaneously leaving sinus like traces on the ground as shown in Figure 1.4 a). It requires three contact points for continuous locomotion: two points to produce a propulsive force and a third one to regulate the direction of this force [11]. With the help of its scales, the snake sticks to the ground to prevent side slipping as much as possible. This type of gait is mainly observed on rough and muddy surfaces.

1.2.2 Concertina Locomotion

This kind of locomotion attains its name from a musical instrument similar to an accordion. Basically the snake folds and twists a certain part of its body and keeps it fixed while the rest of the body sections use it to push themselves forward as

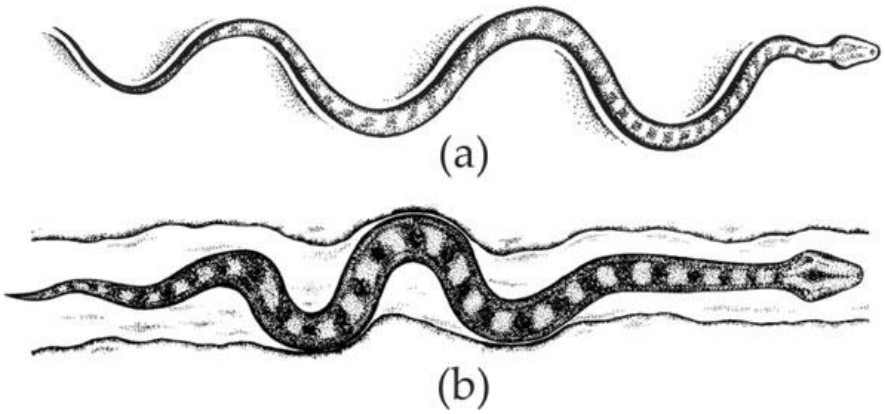


Figure 1.4: a) Lateral undulation and b) Concertina locomotion. Adapted from Casssell Illustrated [12]

shown in Figure 1.4 b). Consecutively every section takes the role of the fixed part and the whole body moves forward. This kind of gait is employed on the ground having a low effective area like tree branches and narrow passages.

1.2.3 Sidewinding Locomotion

Sidewinding might be the most startling form of gait encountered in snakes. This type of gait is usually utilized by snakes living in deserts where they have to move on the grounds with low shear strengths like sand. Basically the snake raises and oscillates its body in a travelling wave leaving two contact patches as shown in Figure 1.5 on the sand. As result a down force is applied by the snake body on the ground to minimize slippage. A total of one kilometer of continuous locomotion distances have been reported by utilizing sidewinding [13].

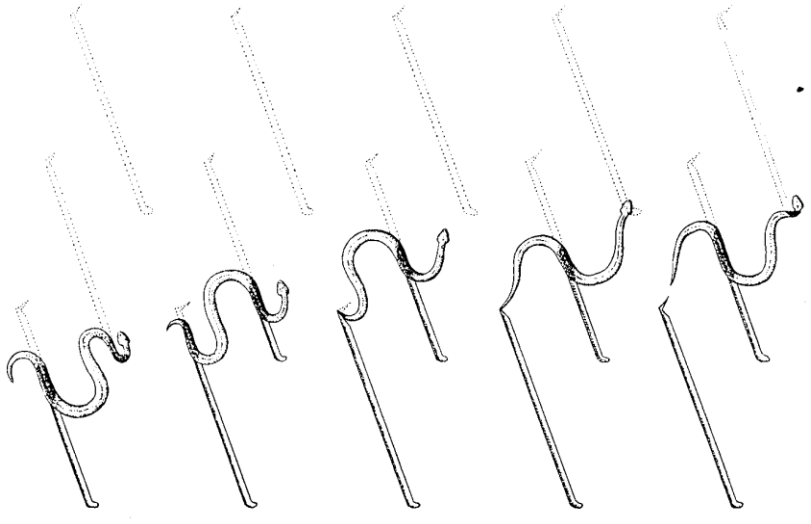


Figure 1.5: Sidewinding locomotion. Adapted from [6]

1.3 Range of Application of Snake Robots

Having a versatile and scalable structure, snake like modular robots have recently found quite vast areas of utilization. Apart from having locomotive abilities (which is the main focus of interest of this study) fixing the head (or tail) of these kind of robots quickly transforms them into manipulators. Although the robots which make use of actively propelled wheels have been extensively used and proved their success with high speeds and efficiency, when it comes to maintaining the locomotion on rough terrains or narrow passages, these mechanisms are visibly overcame by the snake like modular robots. The main areas of application of the serpentine robots can be categorized as exploration, inspection, medical, dangerous and hazardous environments, surveillance, reconnaissance, routing and most significantly search and rescue operations. Modular snake like robots can be used to reach for people trapped in collapsed buildings after earthquakes. They can sneak into narrow passages and provide the wounded with water, food and medicine. Snake robots with search and rescue capabilities are being researched by Carnegie Mellon University Biorobotics laboratory as in [14]. Serpentine robots

can be used in planetary exploration missions where unknown extreme conditions and terrains are involved. Such robots are currently being developed by Nasa Ames Research Center. Another important utilization area for the snake robots are inspection and surveillance of structures and zones which may be quite dangerous and hazardous for human beings such as nuclear power plants, military zones covered mines and long pipeline networks. Also, the snake like robots can be used for military and civil reconnaissance purposes such as intelligence gathering about the enemy fronts and surveillance of international borders. Furthermore, snake robots are being developed for surgical purposes with their ability to maneuver inside the human chest and internal organs. Due to the fact that the mechanisms of propulsion of snakes are quite similar in water and on the ground, by proper sealing of the modules and the joints they can also be used in marine environments. One such example is the amphibious snake-like robot "ACM-R5" developed by the Hirose-Fukushima Robotics Lab [15]. It is able to locomote both in the sea and on the ground.

CHAPTER 2

LITERATURE SURVEY

In this chapter, what has been done in the area of serpentine robots will be surveyed. Also the main difficulties and problems in generating locomotion for the snake like robots will be described. Finally, the goal and scope of this thesis will be given with an accompanying layout which will describe the order and methodology of the study.

2.1 Survey of Modular Snake Like Mobile Robots

As have been stated in the survey study [16], J. Gray was the first one to make research in serpentine locomotion in a qualitative manner in his paper [7] in 1946, while Shigeo Hirose was the first one to realize a functioning biologically inspired snake like robot in 1972. However, in [6] it is stated that earliest serpentine mechanism can be dated back to 1920's by the Russian constructivist artist Petr Miturich. He designed several mechanisms called *volnoviki* intended to work on the ground, in water and even in the air. All of his designs were passive ones without any power source or control. The serpentine robot developed by Hirose was called Active Cord Mechanism model ACM III which was 2 m long with 20 one-DOF joints as shown in Figure 2.1. The robot was put on passive wheels and the forward locomotion was accomplished by activating the joints to the sides in different patterns. The Active Cord Mechanism was the inception and many other

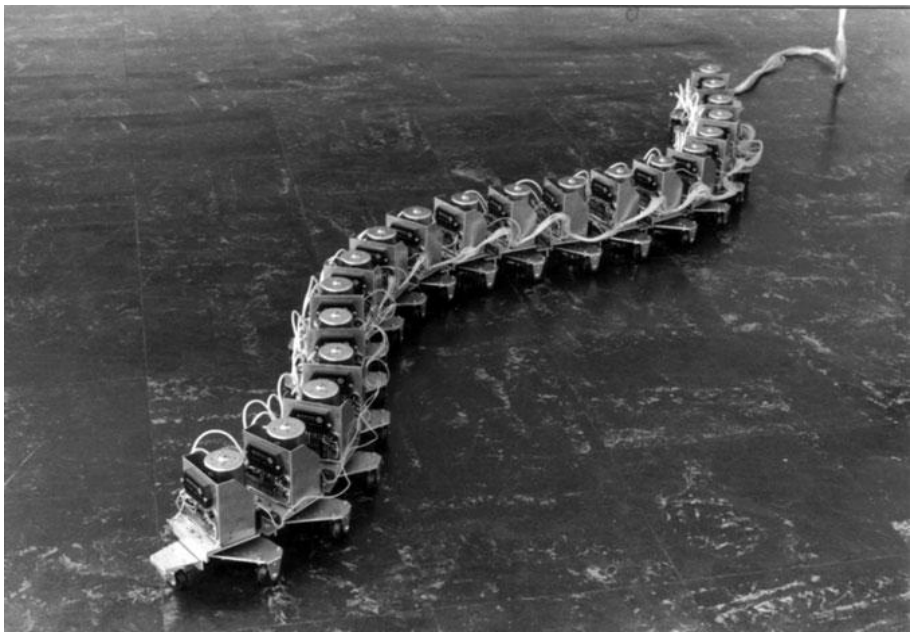


Figure 2.1: Active Cord Mechanism model ACM III [17], Adapted from Oxford University Press

snake like robots have been developed afterwards by many researchers with several different names with several different locomotion methods and aims. Especially during the last fifteen years, there has been a significant increase in the published literature on serpentine robots [18]. Snake like robot designs in the literature can be categorized according to different criteria. These include whether wheels are employed or not, activeness or passiveness of the wheels if they are employed, the type of gait (locomotion pattern) used and the dimension of the working space (2D or 3D). Frequently, the robots developed belong to more than one of these categories. Another categorization is the type of the mathematical model used to represent the robot and it usually depends on the design. These are mainly kinematic and dynamic models. Kinematic models ignore the dynamics of the system and generally focus on geometric aspects of the motion. It generally employs modeling techniques like the well known Denavit-Hartenberg convention, backbone curve and nonholonomic constraints. Kinematic models are usually used when wheels are employed during the locomotion since interaction between the ground and modules through friction are ignored. It is assumed that the wheels do

not slip to the sideways to represent the ideal snake skin properties. Some of the examples in the literature can be listed as [19], [20] and [21]. Most of the time, snake robots which are modeled considering the dynamics are without wheels. That is because the friction forces acting from the ground becomes significant. In dynamic models, generally the most widely used methods to derive the equations of motion of the system are Newton-Euler formulation and the Lagrangian formulation. As it should be, these two methods give the same results when the modules of the robots are considered to be rigid bodies. However, both of the methods have some advantages. Newton-Euler formulation is convenient when it is necessary to find the torques to be applied to attain a certain motion, while the Lagrangian formulation is very suitable when the complete time histories of the states or generalized coordinates of the robot are necessary [22]. Some of the published literature which uses dynamic models of snake robots can be listed as [23], [24], [25] and [26]. Of these papers, [23] and [26] use a robot model with passive wheels employed similar to the model in [17] except that the wheel axes are fixed, while [24] and [26] utilize a robot model without any wheels where friction and contact forces play a significant role. These listed papers provide mathematical models which are easy to implement and commonly cited in literature. In the models where no wheels are employed, several friction models are developed using a Coulomb friction, viscous friction model or a combination of both, which can be found in [23], [27], [28] and [6]. Such models attempt to construct friction models which can mimic the anisotropic friction ($\mu_R < \mu_N$) properties of the snake skin as much as possible. This anisotropy is attained maximally and even is exaggerated when the passive wheels are employed. It is still not clear what kind of friction and dynamic modeling is superior to each other as each has some advantages and disadvantages depending on simulation and analysis purposes of the particular cases. Apart from how the snake like robots are constructed, how to make them locomote in a *reasonable* manner is of high significance. In its broadest meaning, the control methods used to make snake like robots locomote can be divided into three main categories: sine-based approaches, model based approaches and CPG (Central Pattern Generator) based approaches.

Sine based approaches use simple sinusoidal functions to generate the reference signals for the actuators as in papers [29] and [30]. Advantage of this method is its simplicity and explicitly defined frequency, amplitude and wavelength of the reference signals. However, the major disadvantage is the inability to modify these simple sinusoidal functions online as such modifications lead to jerky signal in turn leading to possible damages to the actuators and power transmission systems. Apart from totally ignoring the robot model at all, it is not easy to integrate feedback signals to simple sinusoidal functions used as joint references [31]. Model based approaches, as the names implies, relies on the kinematic or the dynamic model of the robot. All of the papers mentioned so far and that will be mentioned apply model based control strategies unless otherwise is stated. Model based approaches make it possible to analyze the overall systems and determine the optimal parameters and the most efficient gaits, as well as allowing necessary feedbacks. This means, they are quite suitable for designing controllers. However, synthesized controllers cannot be always manipulated online and the accuracy of the developed models might sometimes decrease due to unexpected changes in the environment or uncertainty of the model parameters. CPG based approaches mainly aim to generate rhythmic patterns of locomotion like the locomotion mechanisms observed in the central neural systems of animals. CPGs use sets of dynamical systems (sets of differential equations) like coupled nonlinear oscillators or neural networks and their solutions, as limit cycles are used to generate the reference signals in the form of travelling waves. Some of the papers where CPG is used to control the locomotion are [31], [32], [33], [34] and [35]. In CPG based control architectures, the final forms of limit cycles can be modulated online and smoothly and hence allowing the operators of the robots to manipulate the overall locomotion. However, being a still actively researched area, it is still unclear how to design a CPG to obtain a particular locomotion pattern. That is because, most of the times, they do not have explicit relations between their parameters and the locomotion parameters [31]. A comprehensive survey on state of the art snake inspired robots can be found in [8] where several snake like robots are listed with their photographs, categorized and evaluated. List of some selected

snake like robots with their dimensions and performance are given in Table 2.1 which is adapted from [8].

Table 2.1: List of some selected snake like robots. Adapted from [8]

Robot	Type	Length (mm)	Cross Section (m^2)	Weight (kg)	Velocity (mm/s)
ACM III [17]	Passive Wheels	2000	0.023	28	400
AmphiBot II [36]	Passive Wheels	770	0.002	-	400
OmniTread (OT-8) [37]	Without Wheels 3D	1270	0.034	13.6	100
JL-I [38]	Modular, Separable, Without Wheels	1050	0.038	21	180
CMU (Carnegie Mellon Uni.)	3D, Without Wheels	840	0.003	1.26	102
Slim Slime Robot (SSR) [39]	Without Wheels	730	0.013	12	60

Snake like robots with passive wheels have a different locomotion mechanism than other wheeled mobile robots, because the wheels are not driven. Instead, the joints connecting the modules are driven. Such a mechanism is the main area of interest of this thesis. This is because, frictional anisotropy can be represented without complicated friction models and it is quite suitable for lateral undulation. Lateral undulation has been observed to be one of the fastest forms of gaits in natural snakes [7] and is therefore widely adopted for serpentine robots. Locomotion

control of snake like robots with one-DOF joints and passive wheels are discussed in papers like [24], [25], [40], [41] and [42] with different control techniques and approaches. Passive wheels are attached at the center of gravity of each module and prevent side slipping. Two examples of such snake like robots can be seen in Figure 2.2 and Figure 2.3. In robots with passive wheels and with no side slip constraint, there are certain singular configurations in which the robot cannot locomote. Those are the straight line and circular configurations. Hence, it becomes of uttermost importance to avoid those singularity postures. To avoid

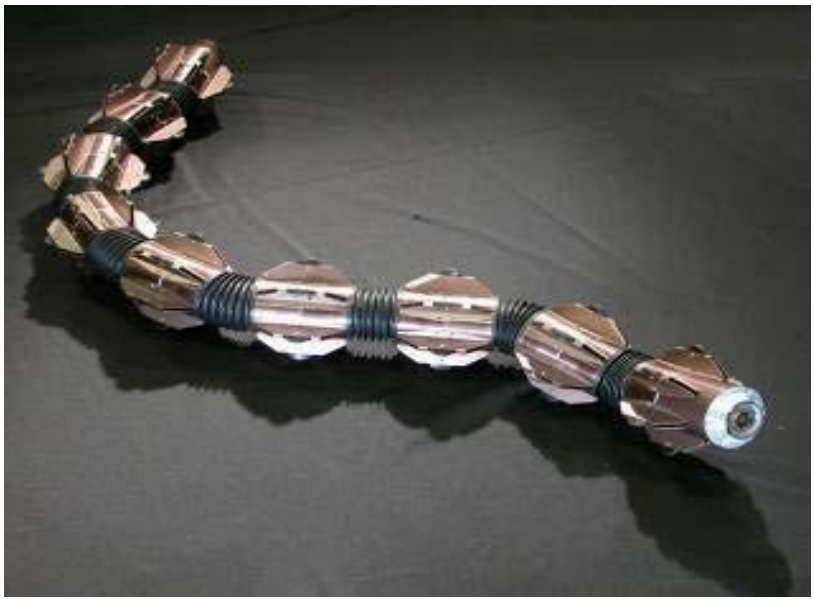


Figure 2.2: ACM-R5 (Shigeo Hirose, Hirose-Fukushima Robotics Lab)

Figure 2.3: AmphiBot II (Alessandro Crespi, Biologically Inspired Robotics Group at Ecole Polytechnique Fédérale de Lausanne) [36]

the singular configurations, Hirose proposed the serpenoid curve mimicking the natural winding motion of snakes in his seminal study [17]. Details and properties of this curve will be given in Chapter 6. Although singular postures are successfully avoided using his proposed approach, exact position or direction control of the robot is quite difficult, since the gait is a fixed structure. In [24], a control law for the robot head, based on Lyapunov function method, is proposed and the exact position control is achieved. Although exact position control is achieved, the control inputs turned out to be non-smooth and peaky. Then they attempted to use Hirose's serpenoid curve as a reference for the robot head and managed to decrease the magnitude of the control input by about 2/3. However, because of the nature of the serpenoid curve, the reference is still fixed and cannot cover arbitrary trajectories. Also in their study they attempted to tune the parameters of the serpenoid curve such that the applied control torques become minimal. As it will be clear in our simulations afterwards, obtaining a low profile for the actuator inputs does not necessarily imply low levels of energy consumption. Hence, their study lacks the energy consumption considerations. In the commonly cited paper [25], authors define a new manipulability measure such that it is high when large acceleration of the robot head can be obtained with small lateral forces (applied by the ground to the modules). By doing this, they attempt to avoid singularity, since the singularity condition implies the divergence of these lateral forces. Then they construct an acceleration based control. In this control strategy, they set a reference acceleration for the robot head such that it makes the robot head track a desired trajectory and maximize their newly defined manipulability. Although they succeeded in avoiding the singularity, obtained control inputs still have irregular and peaky profiles even when following a straight line. What is more, they tell nothing about the energy efficiency of the performed locomotion. Hence, it is not an easy task to design a control system for a snake like passive wheeled robot which avoids the singularity, tracks an arbitrary trajectory and at the same time is energy efficient.

2.2 Goal of the Thesis

The focus of this thesis is on a generic modular snake like robot with passive wheels which is composed of five modules. The goals of this study can be summarized as follows

- To synthesize a closed loop controller for the robot which generates a forward locomotion pattern
- To make the robot follow any arbitrary feasible path
- To avoid the singularity configuration
- To obtain an energy efficient locomotion pattern which results in small and smooth lateral forces.

Indeed, all of the mentioned goals are closely interrelated with each other.

Utilizing a serpenoid curve does not allow tracking arbitrary tracks because of its definition and nature. However, in several papers even where classical optimal control methods are employed, obtained locomotion patterns are compared to Hirose's serpenoid curve and it is concluded that the results are successful since they resemble it. That is because, if followed, the serpenoid curve has proved itself to be very efficient in terms leading to smooth and low actuator torques and avoiding singularity. On that account, what will be done can be briefly summarized as follows

- Tangential friction will be integrated to the dynamic model given in [24] and [25] in order to improve the model
- A controller will be designed by feedback linearization

- A modified version of the serpenoid curve will be proposed in order to track arbitrary trajectories

- A new rational performance measure will be proposed in order to evaluate the locomotion.

- Parameters which lead to optimally efficient locomotion will be determined by using the defined performance measure.

2.3 Layout of the Thesis

In Chapter 3, the kinematics of the robot will be discussed. Also the nonholonomic no side slip constraints and the singularity condition will be defined. In Chapter 4, the dynamics of the robot will be obtained by using the Lagrangian formulation. Also the tangential friction forces will be incorporated in the obtained equations of motion of the system. In Chapter 5, the equations of motion of the system will be represented in the state space. Afterwards, a controller will be synthesized using the feedback linearization technique. Also the stability and controllability issues of the system will be discussed. Finally, by running simulations, it will be shown that, under certain circumstances, the robot converges to the singularity. In Chapter 6, Hirose's serpenoid curve will be introduced. Then a modified serpenoid curve will be proposed which enables the robot to track arbitrary trajectories. Finally, simulations will be performed in order to observe how the robot successfully tracks the modified serpenoid curve as a reference. In Chapter 7, a new performance measure will be introduced. Afterwards, optimal parameters will be determined accordingly. In Chapter 8, discussion of the obtained results will be presented. Finally, the conclusion and possible future works will be given.

CHAPTER 3

KINEMATICS OF THE MODULAR ROBOT

General kinematic relations governing the overall structure and locomotion of the modular snake like robot will be given in this chapter. A brief introduction to the nature of nonholonomic constraints will be presented. Utilization of Pfaffian type constraints in the modular robot will be discussed.

3.1 Kinematic Representation

The robot[1] is designated to move or locomote in the x-y plane $\in \mathbb{R}^2$ of the Euclidean space $\mathbb{R}^3$. The robot is composed of five links which are consecutively connected by four revolute joints as shown in Figure 3.1. The links are called "the modules" because of their identical geometrical and dynamical structures. The modules of the robot are equipped with wheels in order to prevent the possible side slipping during the locomotion. The wheel axis is parallel to the normal direction of the module and located exactly at the vertical projection of the center of gravity. Longitudinal distance from the C.G (center of gravity) of a module to the center of a joint is *l* as shown in Figure 3.2. "The robot head" is the location on the first module coincident with the "would be joint" location. That is to say, while all of the modules include two joints, the first module includes a joint and a "head"

[1] The modular snake like robot under discussion will be referred simply as "the robot" wherever appropriate.

instead of a joint. The wheels at each module are passive with a locally "fixed" axis of rotation. This means axis of wheels is fixed to the modules with no relative translation or rotation. The sole actuation source of the robot is provided by the four identical DC motors which are assumed to apply necessary torques at the necessary angular speeds. Each identical module has a mass *m* and a moment of inertia *J* (about its CG).

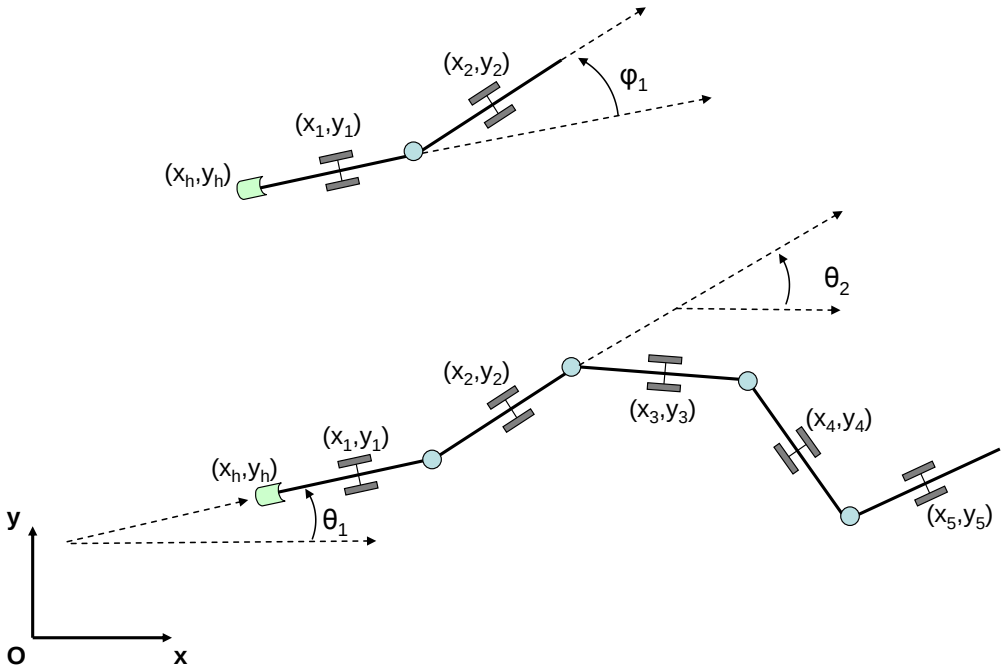


Figure 3.1: Kinematic Representation of the Modular Robot

General kinematic representation of the overall system is depicted in Figure 3.1. The robotic system is treated as an open-chain manipulator where modules form a single serial chain and consecutive modules are connected by one-DOF revolute joints. The modules are indexed from 1 to 5, the revolute joints are indexed from 1 to 4 where the i^{th} joint connects the i^{th} module to the $(i+1)^{th}$ module. Module 0 is assumed to be the ground which is fixed to the global inertial frame. θ_i is the

orientation of the i^{th} module measured in the global frame as shown in Figure 3.1. Furthermore, (x_i,y_i) denotes the coordinates of the center of mass of the i^{th} module measured in the global frame.

The head position of the robot is denoted by $(x_h, y_h) \in \mathbb{R}^2$, again in the global frame. The relative joint angle (simply *joint angle*) ϕ_i is defined as the difference between the two consecutive orientations. Namely:

$$\varphi_i = \theta_{i+1} - \theta_i \qquad i = 1,\ldots,m-1 \tag{3.1}$$

The configuration variables of a locomoting robot can be always partitioned into two classes [42]. The first class of configuration variables defines the position of the robot. Here, the position of the robot is taken to be the position and orientation of the right-handed frame attached to it. In our case this frame is located at (x_h,y_h) with z-axis pointing out of the paper. That is to say: $\boldsymbol{x}, \boldsymbol{y}, \boldsymbol{z} \in \mathbb{R}^3$ where $\boldsymbol{z} = \boldsymbol{x} \times \boldsymbol{y}$. The second class of configuration variables defines the internal configuration of the robot. This class of configuration variables is often called "shape variables" as they determine the overall posture or shape of the mechanism. A manifold of dimension *n* is a smooth *n dimensional hypersurface* which is *locally homoemorphic* to $\mathbb{R}^n$. That is to say, a manifold includes $\mathbb{R}^n$ as a subset but also includes generalized surfaces $\mathbb{S}^n$. The set of all possible shapes are generally described by a manifold *M* while set of all positions i.e., set of all frame displacements, are described by a manifold *G* where *G* is a Lie group. Hence, the total configuration space is $Q = G \times M$. The position and shape variables are interconnected by kinematic constraints of the system. Through manipulation of the shape variables, the position variables can be modified. Generally, the relationship between the manifolds *G* and *M* is a *connection.* *G* which represents the position configuration space of the robot is the product space of $\mathbb{R}^2$ with *SO(2),* which is denoted by *SE(2).* That is to say:

$$SE(2) = \{(r,R) : r \in \mathbb{R}^2, R \in SO(2)\} = \mathbb{R}^2 \times SO(2) \tag{3.2}$$

Here $r = [x_h, y_h]^T$ and *SO(2)* represents the orientation of the frame affixed at the robot head with respect to the global fixed frame i.e., it is a *special orthogonal* matrix satisfying certain properties or simply rotation group of $\mathbb{R}^2$. Then *SE(2)* is a Lie group representing the smooth manifold *G. SE(2)* formally stands for the special Euclidean group of two dimensions [43]. These two dimensions are the group of translations and rotations on the plane. The set of joint variables given by the absolute joint angles $\theta_i \in [0, 2\pi)$ for the i[th] module in the mechanism is associated with the unit circle. This unit circle is denoted by $\mathbb{S}^1$. Hence, $\theta_i \in \mathbb{S}^1$, which is measured in the counter clockwise sense along the direction of the rotation axis. This means that counter clockwise rotations on the x-y plane in Figure 3.1 are considered to be positive. Also from equation (3.1) , $\varphi_i \in [0, 2\pi)$ for the i[th] joint in the robot which is again associated with the unit circle $\mathbb{S}^1$. Thus, the shape configuration space (joint space) of the robot, *M* is given by:

$$M = \mathbb{Z}^n = \mathbb{S}^1 \times \ldots \times \mathbb{S}^1 \tag{3.3}$$

where $\mathbb{Z}^p$ represents the p-torus defined to be the Cartesian product of p-many $\mathbb{S}^1$

Hence, in our case where p is 4, leads to:

$$Q = G \times M = SE(2) \times (\mathbb{S}^1 \times \mathbb{S}^1 \times \mathbb{S}^1 \times \mathbb{S}^1) \tag{3.4}$$

Since φ_i and θ_i are topologically equivalent, configuration space *Q* can be defined by either φ_i or θ_i , whichever is appropriate. Therefore, *Q* can also be defined as

$$Q = \mathbb{R}^2 \times \mathbb{S}^1 \times \mathbb{S}^1 \times \mathbb{S}^1 \times \mathbb{S}^1 \times \mathbb{S}^1 \tag{3.5}$$

Thus, the total configuration variables *q* of the system can be expressed as:

$$q = (q_1, \ldots, q_n) = (q_1, q_2, q_3, q_4, q_5, q_6, q_7) \in Q \tag{3.6}$$

Referring to Figure 3.1, the angular and the position coordinates of the robot can

be defined as:

$$\theta = \begin{bmatrix} \theta_1 \\ \theta_2 \\ \theta_3 \\ \theta_4 \\ \theta_5 \end{bmatrix} \text{ and } r = \begin{bmatrix} x_h \\ y_h \end{bmatrix} \tag{3.7}$$

Thus, $q \in \mathbb{R}^{m+2}$, where m=5, represents a vector of generalized coordinates in the 7 dimensional hyper surface and it is expressed as:

$$q = \begin{bmatrix} \theta \\ r \end{bmatrix} = \begin{bmatrix} \theta_1 \\ \theta_2 \\ \theta_3 \\ \theta_4 \\ \theta_5 \\ x_h \\ y_h \end{bmatrix} \in \mathbb{R}^7 \tag{3.8}$$

Q becomes a 7 dimensional manifold which is smooth differentiable ($\mathbb{C}^\infty$) since it is a Lie group[2], and it is locally diffeomorphic to $\mathbb{R}^7$. That means the robot is kinematically redundant, since the degree of freedom is 7 while the workspace is in $\mathbb{R}^2$. The most general mathematical definition of undulatory locomotion of mechanism of a serial linked class robot can be given as:

Definition: *Undulatory locomotion is the process of generating net displacements of a robotic mechanism via a coupling of internal deformations to a continuous interaction between the robot and its environment* [*42*].

Thus, mechanically, the robot propels itself through constant modulation of joint variables φ_i to generate a net displacement in *r*. This definition is also a *natural*

[2] Properties of Lie groups and general definitions can be found in Appendix A.

one as it formalizes the principle of undulatory terrestrial locomotion seen in nature: displacement through coordinated body shape change.

Although the whole kinematics of the robot can be thought as a serial planar manipulator, there are two major differences unique to serpentine robots:

- All of the modules (links) of the robot translate in $\mathbb{R}^2$ as well as rotating about the vertical axis.
- The modules are in constant interaction with the environment through the friction forces applied by the ground.

These subtle looking major differences from the serial planar robotic manipulators alter both the kinematic and the dynamic behavior of the system dramatically.

3.2 Geometric Relations Governing the Modular Robot

Referring to Figure 3.1, the position and the velocity of the center of gravity of ith module can be expressed as:

$$\left.\begin{aligned} x_i &= x_h + 2l\sum_{j=1}^{i-1}\cos\theta_j + l\cos\theta_i \\ y_i &= y_h + 2l\sum_{j=1}^{i-1}\sin\theta_j + l\sin\theta_i \end{aligned}\right\} \; i = 1,\ldots,m \qquad (3.9)$$

By taking time derivatives of the position variables, the following velocity expressions for the center of gravity of ith module are obtained:

$$\left.\begin{aligned} \dot{x}_i &= \dot{x}_h - 2l\sum_{j=1}^{i-1}\dot{\theta}_j\sin\theta_j - l\dot{\theta}_i\sin\theta_i \\ \dot{y}_i &= \dot{y}_h + 2l\sum_{j=1}^{i-1}\dot{\theta}_j\cos\theta_j + l\dot{\theta}_i\cos\theta_i \end{aligned}\right\} \; i = 1,\ldots,m \qquad (3.10)$$

3.3 Nonholonomic Constraints and Overall Kinematics

A general geometric constraint that can be imposed on a mechanical system is of the form:

$$h_i(q) = 0 \quad i = 1,\ldots,k < n \tag{3.11}$$

These classes of constraints depend solely on the generalized coordinates, q , and restrict the possible motions of the system to an (n-k) dimensional submanifold of the n dimensional manifold Q. A mechanical system may also be subject to *kinematic* constraints apart from the usual geometric ones. Kinematic constraints may also involve time derivatives of the generalized coordinates (first order kinematic constraints) expressed in the form:

$$a_i(q,\dot{q}) = 0 \quad i = 1,\ldots,k < n \tag{3.12}$$

In most practical engineering applications, equation (3.12) is linear in the velocities and is expressed in the form:

$$a_i(q)\dot{q} = 0 \quad i = 1,\ldots,k < n \qquad \text{or} \qquad A(q)\dot{q} = 0 \tag{3.13}$$

where $A(q) \in \mathbb{R}^{k\times n}$ and $a_i(q)$ is the row vector describing one constraint on the directions in which $\dot{q}$ is allowed to evolve. These types of constraints are called Pfaffian type constraints. Pfaffian constraints sometimes can be integrable. That is to say, there may exist k functions h_i such that

$$\frac{\partial h_i(q(t))}{\partial q} = a_i(q) \qquad i = 1,\ldots,k < n \tag{3.14}$$

Existence of k functions h_i satisfying equation (3.14) means that the constraints are indeed geometric and restricts the whole motion to a submanifold of lower dimension [44]. A set of Pfaffian constraints is called *holonomic* if they are

integrable (geometric constraints) and is called *nonholonomic*, otherwise. In the case of the nonholonomic constraints, although at each certain state (configuration), the set of all possible infinitesimal motions (velocities) are restricted to the (*n*-*k*) dimensional null space of the constraint matrix $A(q)$, it is still possible to reach any configuration in *Q*, provided that the system evolves through the *feasible* velocities [43]. However, this is not the case in holonomic constraints where any configuration may not be reached due to the geometric restrictions. Determining whether a set of constraints imposed on a mechanical system is integrable or not (i.e. holonomic or nonholonomic) is not obvious and requires application of tools from differential geometry.

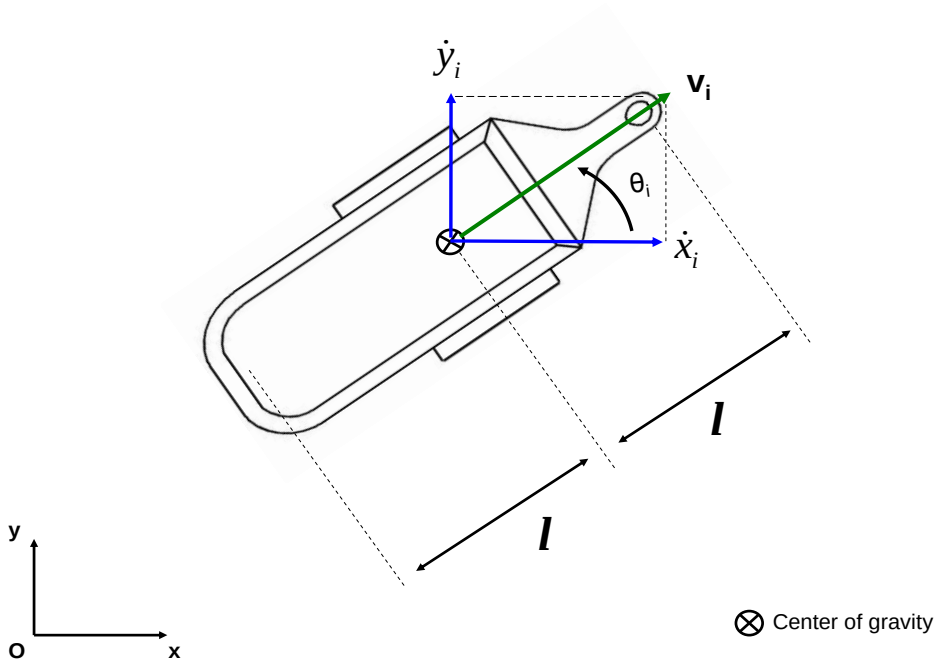


Figure 3.2: Representation of the no side-slip condition for the ith module

The wheels attached to each module prevent the modules from side slipping at any instant of locomotion. This is kinematically equivalent to the notion of having zero velocity of the center of gravity of each module along the normal direction. As depicted in Figure 3.2, the direction of the translational velocity of the center of

gravity of the i[th] module v_i, coincides with its longitudinal direction. In other words, no component of v_i is along the normal direction.

Having said this, the nonholonomic constraints governing the kinematics can be expressed as follows:

$$\dot{x}_i \sin\theta_i - \dot{y}_i \cos\theta_i = 0 \text{ , for } i = 1,\ldots,5 \tag{3.15}$$

Substituting equation (3.10) into equation (3.15) leads to:

$$\begin{aligned} &\dot{x}_h \sin\theta_i - 2l\sum_{j=1}^{i-1}\dot{\theta}_j \sin\theta_i \sin\theta_j - l\dot{\theta}_i \sin^2\theta_i - \\ &(\dot{y}_h \cos\theta_i + 2l\sum_{j=1}^{i-1}\dot{\theta}_j \cos\theta_i \cos\theta_j + l\dot{\theta}_i \cos^2\theta_i) = 0 \qquad i = 1,\ldots,5 \end{aligned} \tag{3.16}$$

Simplifying equation (3.16) leads to:

$$\dot{y}_h \cos\theta_i - \dot{x}_h \sin\theta_i + 2l\sum_{j=1}^{i-1}\dot{\theta}_j \cos(\theta_i - \theta_j) + l\dot{\theta}_i = 0 \qquad i = 1,\ldots,5 \tag{3.17}$$

The kinematic equations and the structure of the system are identical with the ones in [24] and to the one in [25], except for the measurement of the absolute orientation θ_i of the i[th] module such that it is measured from the *x-axis* in the counter clockwise direction in this study while it is measured from the *y-axis* in the clockwise direction in [24]. Equation (3.17) can be put into the generic Pfaffian form by collecting the terms linear in the derivatives of generalized coordinates, $\dot{q} = \begin{bmatrix} \dot{\theta}_1 & \dot{\theta}_2 & \dot{\theta}_3 & \dot{\theta}_4 & \dot{\theta}_5 & \dot{x}_h & \dot{y}_h \end{bmatrix}^T$. Keeping the same notation convention as in [24] and in [25], the following Pfaffian form is obtained:

$$\left[\begin{array}{ccccc:cc} l & 0 & 0 & 0 & 0 & -\sin\theta & \cos\theta_1 \\ 2l\cos(\theta_2-\theta_1) & l & 0 & 0 & 0 & -\sin\theta_2 & \cos\theta_2 \\ 2l\cos(\theta_3-\theta_1) & 2l\cos(\theta_3-\theta_2) & l & 0 & 0 & -\sin\theta_3 & \cos\theta_3 \\ 2l\cos(\theta_4-\theta_1) & 2l\cos(\theta_4-\theta_2) & 2l\cos(\theta_4-\theta_3 & l & 0 & -\sin\theta_4 & \cos\theta_4 \\ 2l\cos(\theta_5-\theta_1) & 2l\cos(\theta_5-\theta_2) & 2l\cos(\theta_5-\theta_3) & 2l\cos(\theta_5-\theta_4) & l & -\sin\theta_5 & \cos\theta_5 \end{array}\right]\begin{bmatrix}\dot{\theta}_1\\ \dot{\theta}_2\\ \dot{\theta}_3\\ \dot{\theta}_4\\ \dot{\theta}_5\\ \dot{x}_h\\ \dot{y}_h\end{bmatrix}=0 \quad (3.18)$$

Equation (3.18) is identical with the one in [24], except for the last two columns, because of the different orientation conventions. It can be written in a partitioned form as:

$$\begin{bmatrix}F_A & -F_B\end{bmatrix}\begin{bmatrix}\dot{\theta}\\ \dot{r}\end{bmatrix}=0 \quad (3.19)$$

where θ and r are defined in equation (3.7) while F_A and F_B can be given as follows :

$$F_A=\begin{bmatrix} l & 0 & 0 & 0 & 0 \\ 2l\cos(\theta_2-\theta_1) & l & 0 & 0 & 0 \\ 2l\cos(\theta_3-\theta_1) & 2l\cos(\theta_3-\theta_2) & l & 0 & 0 \\ 2l\cos(\theta_4-\theta_1) & 2l\cos(\theta_4-\theta_2) & 2l\cos(\theta_4-\theta_3 & l & 0 \\ 2l\cos(\theta_5-\theta_1) & 2l\cos(\theta_5-\theta_2) & 2l\cos(\theta_5-\theta_3) & 2l\cos(\theta_5-\theta_4) & l \end{bmatrix}\in\mathbb{R}^{5\times5} \quad (3.20)$$

$$F_B=\begin{bmatrix}\sin\theta_1 & -\cos\theta_1\\ \sin\theta_2 & -\cos\theta_2\\ \sin\theta_3 & -\cos\theta_3\\ \sin\theta_4 & -\cos\theta_4\\ \sin\theta_5 & -\cos\theta_5\end{bmatrix}\in\mathbb{R}^{5\times2} \quad (3.21)$$

From equation (3.19) :

$$\begin{aligned}&\dot{\theta}=F\dot{r} \quad \text{where}\\ &F=F_A^{-1}F_B\in\mathbb{R}^{5\times2}\end{aligned} \quad (3.22)$$

Multiplying equation (3.19) by F_A^{-1}, leads to:

$$\begin{bmatrix} I_5 & -F(\theta) \end{bmatrix} \begin{bmatrix} \dot{\theta} \\ \dot{r} \end{bmatrix} = A(q)\dot{q} = 0 \qquad A(q) \in \mathbb{R}^{5\times 7} \tag{3.23}$$

Hence, the constrained kinematic relations of the robot are in their most general form described by equation (3.13). Also from equation (3.1):

$$\varphi = E^T\theta \quad \text{where } E^T = \begin{bmatrix} -1 & 1 & 0 & 0 & 0 \\ 0 & -1 & 1 & 0 & 0 \\ 0 & 0 & -1 & 1 & 0 \\ 0 & 0 & 0 & -1 & 1 \end{bmatrix} \tag{3.24}$$

E^T is simply the transformation matrix between the absolute orientations and the joint angles.

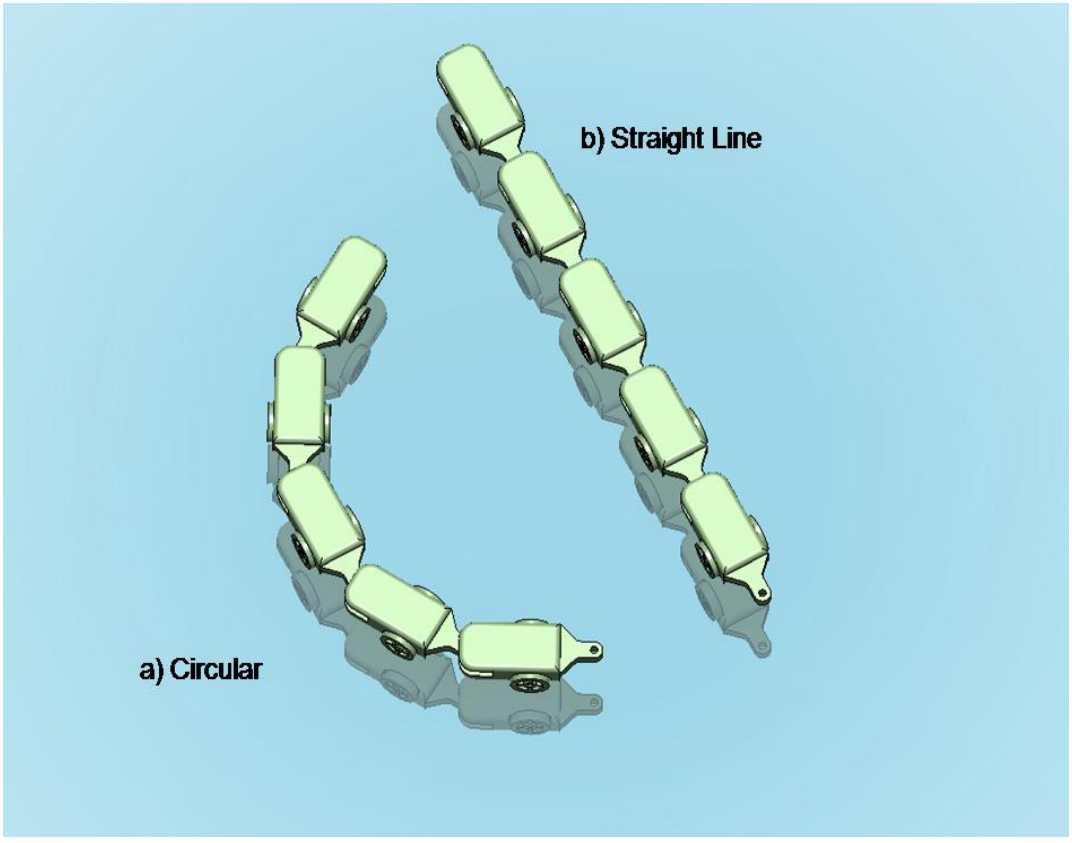


Figure 3.3: Illustration of the singular configurations of the modular robot. a) Circular shape configuration, b) Straight shape configuration

Taking the time derivative of equation (3.24) and using equation (3.22) gives:

$$\dot{\varphi} = E^T F \dot{r} \tag{3.25}$$

Lemma: $E^T F$ *which is of size* $(m-1)\times 2$ *, drops rank if and only if* $\varphi_1 = \varphi_2 = \ldots = \varphi_{m-1} \pmod{\pi}$ *holds* [24].

This means $E^T F$ drops ranks when the robot body shape becomes straight as in Figure 3.3.b) where $\varphi_1 = \varphi_2 = \ldots = \varphi_4 = 0$ or it forms an arc of a perfect ring as in Figure 3.3.a) where $\varphi_1 = \varphi_2 = \ldots = \varphi_4 = 30^o$.

Since *m* = 5 in our case (five modules), matrix $E^T F$ is of size (4×2) and can posses a rank of *min*(4,2) = 2, at most. Equation (3.25) defines the relation between the joint velocities of the robot and the robot head velocity. Since a *spatial manipulator Jacobian* maps the joints velocity vector at each configuration θ to the corresponding velocity of the end-effector [43], it can be concluded that $(E^T F)^+$ turns out to be the Jacobian of the robot. In our case, kinematically, the head of the robot can be considered to be the *end-effector*. Here $(E^T F)^+$ represents the Moore-Penrose pseudoinverse (the generalized pseudoinverse) [45] of $E^T F$. If the columns of $E^T F$ are linearly independent then its Moore-Penrose pseudoinverse exists and becomes the left inverse of $E^T F$ [46]. Hence, rank deficiency of $E^T F$ implies inexistence of its Moore-Penrose pseudoinverse and thus implies singularity. Mechanically it means that, when the singularity occurs, regardless of the value of $\dot{\varphi}$ the robot cannot be driven fully since only a limited subset of $\dot{r}$ can be generated. As explained in [24], when a modular robot possesses only two joints (three modules) with values φ_1 and φ_2, the condition $\varphi_1 = \varphi_2$ holds inevitably at some time instance during the locomotion in order to generate a symmetrical undulation. Therefore, at least four modules are needed to successfully generate the undulatory locomotion without attaining the singular configuration.

Furthermore, due to the mathematical nature of the nonholonomic constraints, they do not decrease the overall degree of freedom of the system like the holonomic (geometric) constraints do which are expressed in equation (3.9). If the holonomic constraints given by equation (3.9) or the nonholonomic constraints given by equation (3.23) had not existed, there would be a total of $m\times 3+2=5\times 3+2=15+2=17$ degrees of freedom. Here, three degrees of freedom come from translation of each module in x and y directions and orientation of each module with respect to the global fixed frame. Other extra two degrees of freedom comes from consideration of robot head position (x_h, y_h) as independent. It is clear that it does not make any sense and physically the robot would have 15 degrees of freedom if none of its module were connected. (x_h, y_h) are counted as extra two degrees of freedom for algebraic consistency.

Automatically, set of 2*m* holonomic (geometric) constraint equations given by equation (3.9) where $m=5$, reduces the degree of freedom of the system by 10 to 7. This means that the robot configuration is restricted to a 7 dimensional smooth manifold and this configuration is uniquely determined by the generalized coordinates $q\in\mathbb{R}^7$. However, it can be seen that imposition of nonholonomic constraints given by equation (3.23) does not reduce degree of freedom of the system. That is to say, while equation (3.9) dictates *which configuration is allowed to attain*, equation (3.23) dictates *how this configuration is allowed to attain.*

Namely, the robot can be driven to any configuration as long as the generalized velocities $\dot{q}$ obey equation (3.23). Furthermore there is a strong relationship between the accessibility and complete nonholonomy for the nonlinear systems with drift, and between controllability and complete nonholonomy for the driftless nonlinear systems [47], [48], [49], [50].

CHAPTER 4

DYNAMICS OF THE MODULAR ROBOT

Equation of motion of the n-module system is derived in [24] using the Lagrangian energy methods. Based on these equations, improvements are made regarding the rolling friction (tangential) force acting on the modules. In the derivation of the equations of motion in [24], coefficients of friction of translational and rotational motion are taken as linearly dependent on the velocities as if the system were moving on a fluid (a viscous friction model). This approach automatically makes these coefficients as damping constants. Although this assumption renders the mathematics of the dynamics a lot simpler, it is far from representing the actual dynamics. However, these set of equations are widely adopted in literature, for example in [25], [40], [51] and [26]. As an extreme example, in [52], they assumed zero rolling (tangential) friction and a boundless (infinitely large) normal friction at the wheels.

4.1 External Forces Acting On the System

During the locomotion of the robot, the externally applied forces can be categorized as the ones originating from the environment and the ones generated by the actuators of the system. Forces applied by the environment are in the form of friction. Nature of the applied friction force is rather complicated and still several studies are carried to fully comprehend and model it in a quantitative manner. A number of models describing the anisotropic friction characteristics of

snake locomotion are put forward. These anisotropic friction models combine Coulomb friction and viscous friction models with assumed *friction ellipses* in order to have a better estimation. However, in order not to complicate the physical model and to focus on the actual goal of the thesis, an approach based on simple Coulomb friction will be utilized. In Figure 4.1, the classical free body diagram for

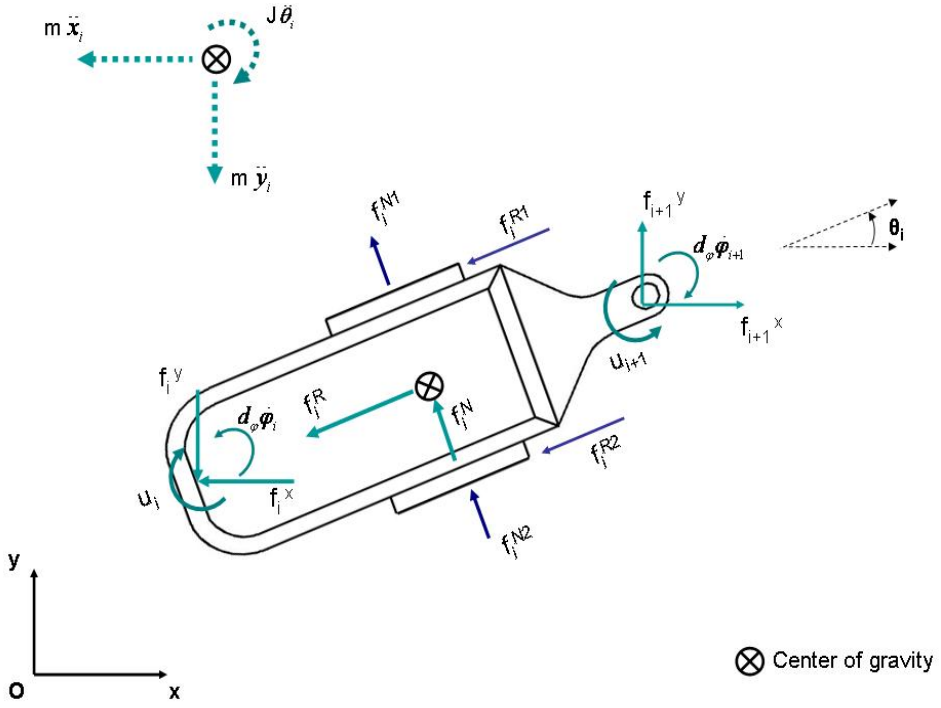


Figure 4.1: Free Body diagram for a single module

a single module of the robot is shown. The forces acting on the i[th] module (with mass *m* and moment of inertia *J*) due to the interaction with the environment are the rolling friction forces acting on two wheels $f_i^{R_1}$, $f_i^{R_2}$ and the normal (lateral) friction forces acting on the two wheels $f_i^{N_1}, f_i^{N_2}$. The reaction forces f_i^x, f_i^y are the forces applied by the (i-1)[th] module to the i[th] module in the x and y directions, respectively. Similarly, the reaction forces f_{i+1}^x, f_{i+1}^y are the forces applied by the (i+1)[th] module to the i[th] module in the x and y directions, respectively. Clearly, the

reaction forces applied by the ith module to (i-1)th and (i+1)th modules are the same in magnitude but in opposite directions. The damping torques generated at the ith revolute joint is simply $d_\varphi \dot{\varphi}_i$. On each module (except for the head and tail modules, where there is only one) there are two identical motors. These motors apply the control torques u_i and u_{i+1} to the ith module as shown in Figure 4.1. Clearly from action-reaction principle, the application of the control torque u_{i+1} to the ith module by the (i+1)th actuator implies application of control torque $-u_{i+1}$ to the (i+1)th module. It is convenient to lump symmetrical frictional forces to the center of gravity of each module by assuming that $f_i^{R_1} = f_i^{R_2}$ and $f_i^{N_1} = f_i^{N_2}$. The resulting normal and rolling friction forces then become

$$\left.\begin{aligned} f_i^R &= f_i^{R_2} + f_i^{R_1} = 2 f_i^{R_1} \\ f_i^N &= f_i^{N_2} + f_i^{N_1} = 2 f_i^{N_1} \end{aligned}\right\} \; i = 1, \ldots, m \qquad (4.1)$$

These lumped forces are also depicted in Figure 4.1. Thus the vector of control torques becomes:

$$u = \begin{bmatrix} u_1 \\ u_2 \\ u_3 \\ u_4 \end{bmatrix} \in \mathbb{R}^4 \qquad (4.2)$$

Vector of normal frictional forces are given by:

$$f^N = \begin{bmatrix} f_1^N \\ f_2^N \\ f_3^N \\ f_4^N \\ f_5^N \end{bmatrix} \in \mathbb{R}^5 \qquad (4.3)$$

And the vector of rolling frictional forces is given by:

$$f^R = \begin{bmatrix} f_1^R \\ f_2^R \\ f_3^R \\ f_4^R \\ f_5^R \end{bmatrix} \in \mathbb{R}^5 \tag{4.4}$$

As explained in Chapter 1, tangential frictional forces acting on snakes are considerably smaller than normal (lateral) ones. In the same manner, $\left\| f^R \right\| \in \mathbb{R}$ should be substantially smaller than $\left\| f^N \right\| \in \mathbb{R}$. Although this is the case when the locomotion surface is smooth with small disparities due to employment of passive wheels, it well may not be the case on rough surfaces. Hence, consideration of rolling friction is helpful in obtaining a more realistic simulation in such cases. Since f_i^x, f_i^y for $i = 1, \ldots, 4$ are internal forces when the robotic system is considered as a whole body and since Euler-Lagrange equations of motion will be employed, they are not necessary for the dynamic analysis. However, these joint forces can be readily found after solving the system dynamics, whenever needed.

4.2 Equations of Motion

Equations of motion of the system have already been derived in [24] and used by many in literature. These equations of motion are derived using the Lagrangian approach since it provides a convenient mathematical representation of the dynamics for control purposes. What is more, in this approach, internal structural forces which are out of the focus of this study, are eliminated. Starting with the kinetic energy and using the generalized coordinates given by equation (3.8), the kinetic energy of the system is given by

$$T = \frac{1}{2} \sum_1^m \left[m(\dot{x}_i^2 + \dot{y}_i^2) + J\dot{\theta}_i^2 \right] = \frac{1}{2} \dot{q}^T M(\theta) \dot{q} \tag{4.5}$$

where $m=5$, and $M(\theta)$ is the (7×7) positive definite symmetric generalized inertia matrix of the system depending only on the orientation variables θ and system constants *m, L, J.* Since the robot operates on the *x-y* plane and gravitational acceleration acts in the negative *z-direction*, the potential energy is always constant, i.e. *U = c, where c* is a constant. Dissipative energy of the system is given by

$$D=\frac{1}{2}\sum_{1}^{n-1} d_{\varphi}\dot{\varphi}_i^2=\frac{1}{2}\dot{q}^T N(\theta)\dot{q} \qquad (4.6)$$

where $N(\theta)$ is a (7×7) matrix depending only on the orientation variables θ and damping constant d_{φ} of the revolute joints. Thus, the Lagrangian of the system becomes $L=T-U=T$ since *c* can be taken as zero. Here, dissipative energy given by equation (4.6) is different from the one in [24] as the friction forces acting on the module are not assumed to be viscous in this study. *D* is actually, a Rayleigh dissipation function with its most general form $D=\frac{1}{2}\sum_{i=1}^{n}\sum_{j=1}^{n} c_{ij}\dot{q}_i\dot{q}_j$ [53] where c_{ij} are the general damping coefficients. The general Euler-Lagrange equation as expressed in [54] is

$$\frac{d}{dt}\frac{\partial L}{\partial \dot{q}_i}-\frac{\partial L}{\partial q_i}+\frac{\partial D}{\partial \dot{q}_i}=Q_i+\tau_i \quad i=1,\ldots,n \qquad (4.7)$$

Here *k=n+2* is the number of total generalized coordinates. $Q_i=\sum_{j=1}^{m} F_j\cdot\frac{\delta p_j}{\delta q_i}$ is the generalized force including all of the *externally applied non conservative forces* F_j by the environment (excluding the damping forces due to the revolute joints which is $\frac{\delta D}{\delta \dot{q}_i}$). Here, p_j denotes the position vector of the point of application of the externally applied force F_j and *m* is the number of F_j's. Thus in general, Q_i becomes the mapping of the rolling and normal friction forces to the generalized coordinate space. τ_i, on the other hand, is the mapping of the actuator

forces (motor torques u in our case) to the generalized coordinate space. By using the principle of virtual work $u_\theta^T \partial\theta = u_\varphi^T \partial\varphi$

$$\tau = \begin{bmatrix} \tau_1 \\ \tau_2 \\ \tau_3 \\ \tau_4 \\ \tau_5 \\ \tau_6 \\ \tau_7 \end{bmatrix} = \begin{bmatrix} Eu \\ 0 \\ 0 \end{bmatrix} \tag{4.8}$$

where u_θ and u_φ are the actuator torques in generalized coordinate space and joint space, respectively. The Lagrangian equations of motion of the robot given by equation (4.7) do not include the kinematic constraints imposed on it. Thus, the Pfaffian constraints expressed by equation (3.23) are to be incorporated into equation (4.7).

4.3 Friction as Generalized External Force

As mentioned earlier, the external forces acting on the system by the environment are rolling (or sliding) friction and normal friction. Normal friction is assumed to be Coulomb friction which is the force that prevents the modules from side slipping. That is to say, normal friction f^N acts as a constraint force to ensure that equation (3.23) holds at all times. Constraint forces for a set of Pfaffian constraints prevent the motion of the system in the directions that would violate the constraints. These directions are given by the rows of the matrix $A(q) \in \mathbb{R}^{5\times 7}$ in equation (3.23) . These constraint forces are at the configuration space $q \in Q$ [43] and they are given by

$$\Gamma = A^T(q)\lambda \tag{4.9}$$

where $\lambda = \begin{bmatrix} \lambda_1 \\ \lambda_2 \\ \lambda_3 \\ \lambda_4 \\ \lambda_5 \end{bmatrix} \in \mathbb{R}^5$ is the vector whose elements are the Lagrange multipliers which give the relative magnitudes of the constraint forces acting on each module. Here it is assumed that the constraints are everywhere smooth and linearly independent and the forces of constraint do *no work* on the overall system which is called *d'Alembert's* principle. [43]. This assumption is trivially justified as long as equation (3.23) holds, at all times. Thus, the equation of motion of the system is formed by considering the constraint forces as external forces acting on the system. Therefore, the equations of motion of the system including the kinematic constraints are given as

$$M(\theta)\begin{bmatrix} \ddot{\theta} \\ \ddot{r} \end{bmatrix} + C\left(\dot{\theta},\theta\right)\begin{bmatrix} \dot{\theta} \\ \dot{r} \end{bmatrix} + N\begin{bmatrix} \dot{\theta} \\ \dot{r} \end{bmatrix} - \begin{bmatrix} Eu \\ 0 \\ 0 \end{bmatrix} - \begin{bmatrix} I_5 \\ -F^T(\theta) \end{bmatrix}\lambda = Q \qquad (4.10)$$

where $C\left(\theta,\dot{\theta}\right) \in \mathbb{R}^{7\times 7}$ is the Coriolis matrix of the system, while $C\left(\dot{\theta},\theta\right)\begin{bmatrix} \dot{\theta} \\ \dot{r} \end{bmatrix}$ gives the Coriolis and centrifugal forces. Furthermore, $Q = \begin{bmatrix} Q_1 \\ Q_2 \\ Q_3 \\ Q_4 \\ Q_5 \\ Q_6 \\ Q_7 \end{bmatrix} \in \mathbb{R}^{7\times 1}$ is the vector of generalized forces. Substituting equation (3.10) into equation (4.5) $M(\theta) \in \mathbb{R}^{7\times 7}$ is obtained to be:

$$M(\theta)=\begin{bmatrix} 17ml^2+J & 14ml^2\cos(\theta_2-\theta_1) & 10ml^2\cos(\theta_3-\theta_1) & 6ml^2\cos(\theta_4-\theta_1) & 2ml^2\cos(\theta_5-\theta_1) & -9ml\sin(\theta_1) & 9ml\cos(\theta_1) \\ 14ml^2\cos(\theta_2-\theta_1) & 13ml^2+J & 10ml^2\cos(\theta_3-\theta_1) & 6ml^2\cos(\theta_4-\theta_2) & 2ml^2\cos(\theta_5-\theta_2) & -7ml\sin(\theta_2) & 7ml\cos(\theta_2) \\ 10ml^2\cos(\theta_3-\theta_1) & 10ml^2\cos(\theta_3-\theta_1) & 9ml^2+J & 6ml^2\cos(\theta_4-\theta_3) & 2ml^2\cos(\theta_5-\theta_3) & -5ml\sin(\theta_3) & 5ml\cos(\theta_3) \\ 6ml^2\cos(\theta_4-\theta_1) & 6ml^2\cos(\theta_4-\theta_2) & 6ml^2\cos(\theta_3-\theta_4) & 5ml^2+J & 2ml^2\cos(\theta_5-\theta_4) & -3ml\sin(\theta_4) & 3ml\cos(\theta_4) \\ 2ml^2\cos(\theta_5-\theta_1) & 2ml^2\cos(\theta_5-\theta_2) & 2ml^2\cos(\theta_3-\theta_5) & 2ml^2\cos(\theta_5-\theta_4) & ml^2+J & -ml\sin(\theta_5) & ml\cos(\theta_5) \\ -9ml\sin(\theta_1) & -7ml\sin(\theta_2) & -5ml\sin(\theta_3) & -3ml\sin(\theta_4) & -ml\sin(\theta_5) & 5m & 0 \\ 9ml\cos(\theta_1) & 7ml\cos(\theta_2) & 5ml\cos(\theta_3) & 3ml\cos(\theta_4) & ml\cos(\theta_5) & 0 & 5m \end{bmatrix} \quad (4.11)$$

Furthermore the i,j^{th} element of the Coriolis matrix $C\left(\dot{\theta},\theta\right)$ is defined to be [54]

$$c_{ij}=\sum_{k=1}^{n}\frac{1}{2}\left\{\frac{\partial m_{ij}}{\partial q_j}+\frac{\partial m_{ik}}{\partial q_j}-\frac{\partial m_{kj}}{\partial q_i}\right\}\dot{q}_k \quad (4.12)$$

where m_{ij} is the i,j^{th} element of the inertia matrix $M(\theta)$ and the term $c_{kji}=\frac{1}{2}\left\{\frac{\partial m_{ij}}{\partial q_j}+\frac{\partial m_{ik}}{\partial q_j}-\frac{\partial m_{kj}}{\partial q_i}\right\}$ is called the *Christoffel symbol* of the first kind. Then using equation (4.11) and (4.12) , $C\left(\dot{\theta},\theta\right)$ can be found to be

$$C(\theta,\dot{\theta})=\begin{bmatrix} 0 & 14ml^2\sin(\theta_2-\theta_1)\dot{\theta}_2 & 10ml^2\sin(\theta_3-\theta_1)\dot{\theta}_3 & 6ml^2\sin(\theta_4-\theta_1)\dot{\theta}_4 & 2ml^2\sin(\theta_5-\theta_1)\dot{\theta}_5 & 0 & 0 \\ -14ml^2\sin(\theta_2-\theta_1)\dot{\theta}_1 & 0 & 10ml^2\sin(\theta_3-\theta_2)\dot{\theta}_3 & 6ml^2\sin(\theta_4-\theta_2)\dot{\theta}_4 & 2ml^2\sin(\theta_5-\theta_2)\dot{\theta}_5 & 0 & 0 \\ -10ml^2\sin(\theta_3-\theta_1)\dot{\theta}_1 & -10ml^2\sin(\theta_3-\theta_1)\dot{\theta}_2 & 0 & 6ml^2\sin(\theta_4-\theta_3)\dot{\theta}_4 & 2ml^2\sin(\theta_5-\theta_3)\dot{\theta}_5 & 0 & 0 \\ -6ml^2\sin(\theta_4-\theta_1)\dot{\theta}_1 & -6ml^2\sin(\theta_4-\theta_2)\dot{\theta}_2 & -6ml^2\sin(\theta_3-\theta_4)\dot{\theta}_3 & 0 & 2ml^2\sin(\theta_5-\theta_4)\dot{\theta}_5 & 0 & 0 \\ -2ml^2\sin(\theta_5-\theta_1)\dot{\theta}_1 & -2ml^2\sin(\theta_5-\theta_2)\dot{\theta}_2 & -2ml^2\sin(\theta_3-\theta_5)\dot{\theta}_3 & -2ml^2\sin(\theta_5-\theta_4)\dot{\theta}_4 & 0 & 0 & 0 \\ -9ml\cos(\theta_1)\dot{\theta}_1 & -7ml\cos(\theta_1)\dot{\theta}_2 & -5ml\cos(\theta_3)\dot{\theta}_3 & -3ml\cos(\theta_4)\dot{\theta}_4 & -ml\cos(\theta_1)\dot{\theta}_5 & 0 & 0 \\ -9ml\sin(\theta_1)\dot{\theta}_1 & -7ml\sin(\theta_2)\dot{\theta}_2 & -5ml\sin(\theta_3)\dot{\theta}_3 & -3ml\sin(\theta_4)\dot{\theta}_4 & -ml\sin(\theta_5)\dot{\theta}_5 & 0 & 0 \end{bmatrix} \quad (4.13)$$

Lemma: System matrices in equation (4.10) satisfy the following properties [43]:

1. $M(\theta)\in\mathbb{R}^{7\times7}$ is symmetric and positive definite
2. $\dot{M}(\theta)-2C\left(\dot{\theta},\theta\right)\in\mathbb{R}^{7\times7}$ is a skew symmetric matrix.

Skew symmetricity of $\dot{M}(\theta)-2C\left(\dot{\theta},\theta\right)$ is called the passivity property since it states that when any nonconservative force is absent in the system, the net energy of the system is conserved. This property is fundamentally used in control strategies employed in robotic manipulators. [43].

Furthermore, from equation (4.6) $N \in \mathbb{R}^{7\times 7}$ can be obtained to be

$$N = \begin{bmatrix} d_\varphi & -d_\varphi & 0 & 0 & 0 & 0 & 0 \\ -d_\varphi & 2d_\varphi & -d_\varphi & 0 & 0 & 0 & 0 \\ 0 & -d_\varphi & 2d_\varphi & -d_\varphi & 0 & 0 & 0 \\ 0 & 0 & -d_\varphi & 2d_\varphi & -d_\varphi & 0 & 0 \\ 0 & 0 & 0 & -d_\varphi & d_\varphi & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix} \tag{4.14}$$

Thus, it is seen that damping matrix *N* solely depends on the damping coefficient d_φ of the revolute joints (as opposed to the one in [24] where it also depends on θ because of the previously stated reasons). Hence, there are *n=7* configuration variables *q* and *k* Lagrangian multipliers $\lambda_1,\ldots,\lambda_k$ which makes a total of *n+k = 7+5 =12* unknowns. At the same time, there are *n nonlinear second order differential equations* expressed in equation (4.10) and *k constraints equations* expressed in equation (3.23) making *n+k=12* equations in total. Here, it is assumed that the initial configuration $q(t_0) = q_0$ and the actuator torques are known. Therefore solving equations (3.23) and (4.10) simultaneously completely describes the system dynamics for all times in the simulation. In our case, λ is a function of configuration q, its velocity $\dot{q}$, its acceleration $\ddot{q}$, actuator control torques *u* and the rolling friction f^R.

The normal friction forces f^N are the constraint forces which ensure that no motion occurs along the constrained directions and they are included in equation (4.10) in the form given by equation (4.9). Therefore *Q* on the right hand side of equation (4.10) is due to only the rolling friction forces . Here, the notion of rolling friction represents a general frictional term f^R and it may also represent a sliding friction with a higher value than the rolling one. Thus, the generalized force Q_i [54] is as follows

$$Q_i = \sum_{j=1}^{k} f^R{}_j \frac{\partial p_j}{\partial q_i} \quad i = 1,\ldots,7 \tag{4.15}$$

where k is the number of external forces contributing to Q_i, (which is 5 in our case since it is assumed that five lumped distinct rolling friction forces are given by equation (4.4)) and p_j is the point of application of $f^R{}_j$. Thus

$$p_j = x_j \hat{\imath} + y_j \hat{\jmath} \quad j = 1,\ldots,5 \tag{4.16}$$

Using equation (3.9) p_j can be written as a function of x_h, y_h and θ. Thus p_j can be expressed in terms q_i. Also in the same manner $f^R{}_j$ can be decomposed to its components as:

$$\vec{f}^R{}_i = f^R{}_i \cos(\theta_i)\hat{\imath} + f^R{}_i \sin(\theta_i)\hat{\jmath} \quad i = 1,\ldots,5 \tag{4.17}$$

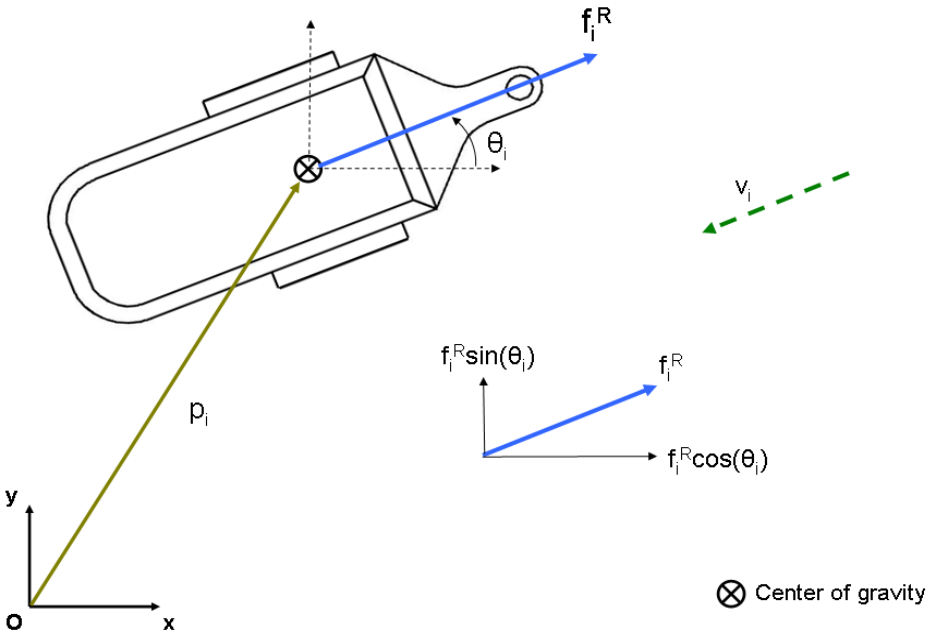


Figure 4.2: Illustration of the lumped rolling friction with its components

In Figure 4.2 , $f^R{}_i$, its decomposition and p_i is illustrated where v_i is the assumed direction of translational velocity of the i[th] module. Thus, $f^R{}_i$ is modeled to be always in the opposite direction of v_i as will be discussed later in detail. Substituting equation (4.16) and equation (4.17) into equation (4.15), the following expressions for the generalized forces Q_i are obtained

$$
\begin{aligned}
Q_1 &= 2lf_2^R \sin(\theta_2-\theta_1) + 2lf_3^R \sin(\theta_3-\theta_1) + 2lf_4^R \sin(\theta_4-\theta_1) + 2lf_5^R \sin(\theta_5-\theta_1)\\
Q_2 &= 2lf_3^R \sin(\theta_3-\theta_2) + 2lf_4^R \sin(\theta_4-\theta_2) + 2lf_5^R \sin(\theta_5-\theta_2)\\
Q_3 &= 2lf_4^R \sin(\theta_4-\theta_3) + 2lf_5^R \sin(\theta_5-\theta_3)\\
Q_4 &= 2lf_5^R \sin(\theta_5-\theta_4)\\
Q_5 &= 0\\
Q_6 &= f_1^R \cos(\theta_1) + f_2^R \cos(\theta_2) + f_3^R \cos(\theta_3) + f_4^R \cos(\theta_4) + f_5^R \cos(\theta_5)\\
Q_7 &= f_1^R \sin(\theta_1) + f_2^R \sin(\theta_2) + f_3^R \sin(\theta_3) + f_4^R \sin(\theta_4) + f_5^R \sin(\theta_5)
\end{aligned}
\qquad (4.18)
$$

From equation (4.18), it is observed that each individual generalized force $Q_i,\ldots,Q_5$ depends on the orientation angles and the rolling frictions with succeeding index numbers, i.e. $\theta_i,\theta_{i+1},\ldots,\theta_5$ and $f_{i+1}^R, f_{i+2}^R,\ldots, f_5^R$ where i starts from one. Therefore, it is quite logical to have $Q_5 = 0$ since otherwise it would have been in the form $Q_5 = 2Lf_6^R \sin(\theta_6-\theta_5)$ which is rather irrelevant because there is not a sixth module. Furthermore, it turns out that Q_6 and Q_7, which are the generalized forces in the equation of motion corresponding to the translational generalized coordinates x_h and y_h, depends on $\theta_1,\theta_2,\ldots,\theta_5$ and $f_1^R, f_2^R,\ldots, f_5^R$. Actually it is seen that they are the summation of the components of $f_1^R, f_2^R,\ldots, f_5^R$ in the respective directions. This is again quite sound since the generalized forces effecting the overall location of the robot should somehow depend on all of the individual rolling friction forces.

Equation (4.18) can be written in the more compact form

$$Q = Z(\theta) f^R \tag{4.19}$$

where $Z(\theta) \in \mathbb{R}^{7\times5}$ is given by

$$Z(\theta) = \begin{bmatrix} 0 & 2l\sin(\theta_2-\theta_1) & 2l\sin(\theta_3-\theta_1) & 2l\sin(\theta_4-\theta_1) & 2l\sin(\theta_5-\theta_1) \\ 0 & 0 & 2l\sin(\theta_3-\theta_2) & 2l\sin(\theta_4-\theta_2) & 2l\sin(\theta_5-\theta_2) \\ 0 & 0 & 0 & 2l\sin(\theta_4-\theta_3) & 2l\sin(\theta_5-\theta_3) \\ 0 & 0 & 0 & 0 & 2l\sin(\theta_5-\theta_4) \\ 0 & 0 & 0 & 0 & 0 \\ \cos(\theta_1) & \cos(\theta_2) & \cos(\theta_3) & \cos(\theta_4) & \cos(\theta_5) \\ \sin(\theta_1) & \sin(\theta_2) & \sin(\theta_3) & \sin(\theta_4) & \sin(\theta_5) \end{bmatrix} \tag{4.20}$$

Detailed modeling of the friction phenomena is a broad area of research in tribology and there are different models at different complexities. However, utilization of Coulomb's dry friction model serves well for many engineering applications, as it has been successfully applied for numerous practical problems. Energy dissipation during rolling or sliding has been studied under three headings in [55] where a detailed discussion about the nature friction can be found.

- Friction and slip on the contact surfaces of bodies on micro scale
- Inelastic deformation of the materials
- Physical roughness of the contact surfaces

As stated in [55], rolling friction force can be formulated as $f^R = N^f \mu_R$ where μ_R is the *coefficient of rolling friction* which depends on the contacting materials, contact conditions and the radius of wheels. N^f is the normal force acting on the contact surface which is simply *mg* in our case, where *g* is the gravitational acceleration. Hence, mathematically, rolling friction can be treated as Coulomb friction simply with a different coefficient of friction. Coulomb dry friction model depends on the following observations [56]

- Frictional force generated is directly proportional to the applied normal load.
- Frictional force is independent of the contact area of two bodies.
- Kinetic friction is independent of the sliding speeds of two bodies.

As stated in [56] these observations are first recorded by Leonardo Da Vinci and then formalized and explained by Charles Coulomb. Hence, dry friction model for sliding or rolling may be described mathematically as

$$f_i^R = \begin{cases} -\mu_R mg & \text{if } v_i > 0 \\ 0 & \text{if } v_i = 0 \\ \mu_R mg & \text{if } v_i < 0 \end{cases} \quad \text{for } i = 1,\ldots,5 \qquad (4.21)$$

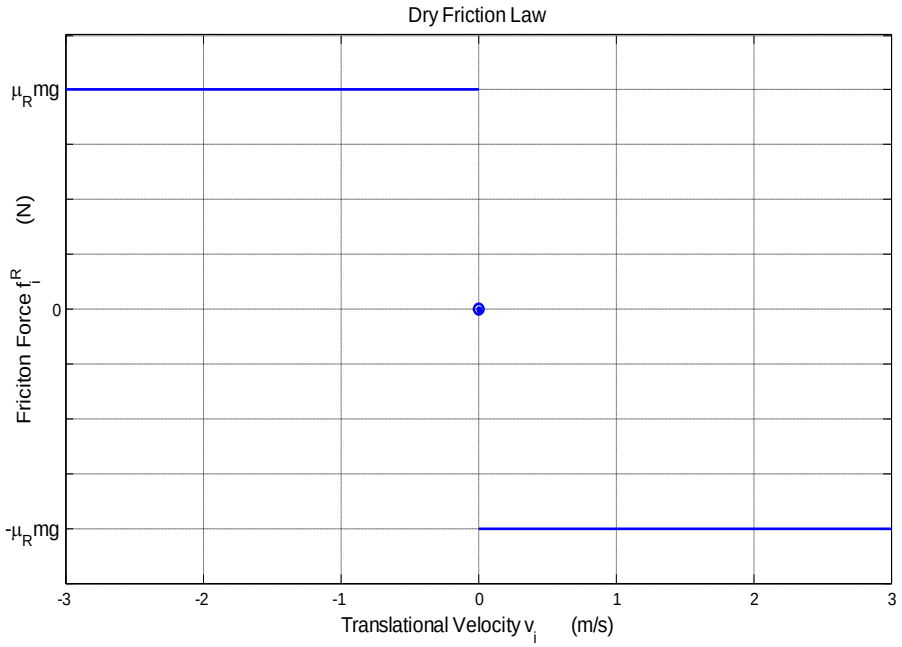


Figure 4.3: Plot of Dry Friction Model

As can be seen from equation (4.21) f_i^R is a discontinuous function due to having many values at $v_i = 0$. However, in literature it is assumed that $f_i^R = 0$ at $v_i = 0$ [56]. Here, from Figure 3.2 it is observed that translational velocity of the i^{th}

module is $v_i = \dot{x}_i \cos\theta_i + \dot{y}_i \sin\theta_i$ as shown in Figure 3.2 Illustration of the standard dry friction model can be seen in Figure 4.3. Utilization of the piece-wise constant dry friction model illustrated in Figure 4.3 poses mathematical difficulties during the solution of the Euler-Lagrange equations of motion given by equation (4.10). Thus, it is convenient to use a continuous and a smooth enough mathematical expression as an approximation of the dry friction. One suggestion that is given in [56] is in the following form

$$f_i^R = -\mu_R mg \frac{2}{\pi} \arctan\left(cv_i\right) \quad \forall v_i \tag{4.22}$$

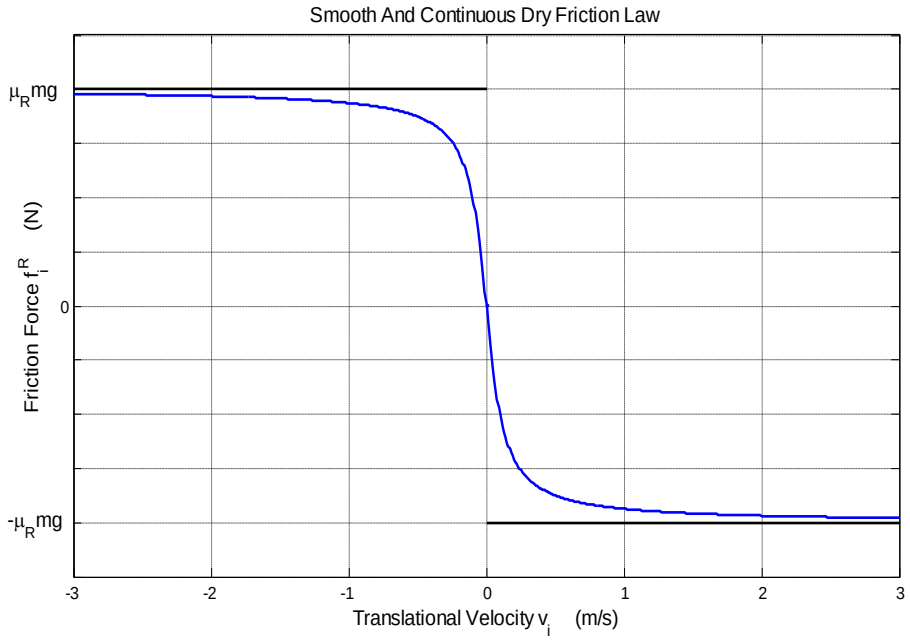


Figure 4.4: Plot of Dry Friction Model suggested by [56]

According to the dry friction model given by equation (4.22) there is a smooth transition from the asymptote $\mu_R mg$ to the asymptote $-\mu_R mg$ due to the inverse tangent function. In equation (4.22), *c* is simply a constant determining the rate of

convergence to the mentioned asymptotes. Plot of the smoothed out dry friction model is given in Figure 4.4 . As can be seen, friction force is again assumed to be zero when the translational velocity vanishes. Although it is known that equation (4.22) is an approximation to the reality, it must be somehow better than equation (4.21). This is because nothing in nature is so "*piece-wise*" and so "*discontinuous*" and there are always some *transitions* no matter how sharp or abrupt. However, one thing that may be "not so correct" in equation (4.22) is the notion of having zero frictional force when the modules are not moving. Indeed, when there is no motion, there is still a frictional force resisting the motion which is in the opposite direction of net force.

4.4 Reduced Equations of Motion

Equations of motion of the system including the nonholonomic constraints are given in their most general form in equation (4.10). It is reduced by simply multiplying by $\left[F^T I_2\right]$ from left [24] in order to eliminate the Lagrangian multipliers λ . $\left[F^T I_2\right]$ is the transpose of the *null space* of the constraint matrix $A(q)$ given by equation (3.23). From equation (3.22) one obtains

$$\ddot{\theta} = \dot{F}\dot{r} + F\ddot{r} \tag{4.23}$$

Using equations (3.22), (4.10), (4.20), and (4.23), the following equation is obtained

$$\begin{aligned}&\left[F^T I_2\right] M(\theta)\begin{bmatrix}\dot{F}\dot{r} + F\ddot{r}\\ \ddot{r}\end{bmatrix} + \left[F^T I_2\right] C\left(\dot{\theta},\theta\right)\begin{bmatrix}F\dot{r}\\ \dot{r}\end{bmatrix} + \left[F^T I_2\right] N\begin{bmatrix}F\dot{r}\\ \dot{r}\end{bmatrix} - \\ &\left[F^T I_2\right]\begin{bmatrix}Eu\\ 0\\ 0\end{bmatrix} - \left[F^T I_2\right]\begin{bmatrix}I_5\\ -F^T(\theta)\end{bmatrix}\lambda = \left[F^T I_2\right] Z(\theta) f^R\end{aligned} \tag{4.24}$$

Equation (4.24) may be written in the form

$$M'(\theta)\ddot{r} + C'(\dot{\theta},\theta)\dot{r} + N'\dot{r} - F^T E u = \left[F^T I_2\right] Z(\theta) f^R \tag{4.25}$$

where the *primed* matrices in equation (4.25) are as follows

$$\begin{aligned} M'(\theta) &= \left[F^T I_2\right] M(\theta) \begin{bmatrix} F \\ I_2 \end{bmatrix} \\ C'(\dot{\theta},\theta) &= \left[F^T I_2\right] C(\dot{\theta},\theta) \begin{bmatrix} F \\ I_2 \end{bmatrix} + \left[F^T I_2\right] M(\theta) \begin{bmatrix} \dot{F} \\ \begin{bmatrix} 0 & 0 \\ 0 & 0 \end{bmatrix} \end{bmatrix} \\ N' &= \left[F^T I_2\right] N \begin{bmatrix} F \\ I_2 \end{bmatrix} \end{aligned} \tag{4.26}$$

where $M'(\theta) \in \mathbb{R}^{2\times 2}, C'(\dot{\theta},\theta) \in \mathbb{R}^{2\times 2}$ and $N' \in \mathbb{R}^{2\times 2}$ are the reduced inertia matrix, the reduced Coriolis matrix and the reduced damping matrix, respectively. $F(\dot{\theta},\theta)$ and $\dot{F}(\dot{\theta},\theta)$,which is the total time derivative of $F(\dot{\theta},\theta)$, are given in the Appendix A due to their lengthy forms. Equation (4.25) is identical to the ones in [24], [25] and [40] except for the term $\left[F^T I_2\right] Z(\theta) f^R$ which accounts for the rolling or sliding friction forces and for different damping matrix *N*, which is due to the different scope of damping assumptions.

CHAPTER 5

STATE SPACE REALIZATION AND CONTROL

State space realization of the equations of motion of the system given in equation (4.10) will be given in this section. After expressing the nonlinear system in the standard form, the control objectives will be expressed in a quantitative manner. Considering the nonlinear dynamics of the robot, the feedback linearization method will be employed to control the head position. Issues of controllability and stability will also be discussed.

5.1 State Space Realization and Standard Form

It is known that almost no phenomena in nature is completely linear and most of the time, the standard procedure of linear approximation about some possible operation points of the system fails to represent actual dynamics. The modular robot dealt with in our case is of no exception. Nonlinear system of the general form $\dot{x} = f(x,u)$ sometimes has the special standard form[3]

$$\dot{x} = f(x) + \sum_{i=1}^{m} g_i(x) u_i \qquad (5.1)$$

where $x \in X$, $u \in U$ and X is a smooth differentiable manifold of dimension n while $U = \mathbb{R}^m$ for some $m \leq n$ with smooth functions f and g_i. Then the system

[3] Here x has nothing to do with CG positions of the modules in Chapter 2. It will represent state variables from now on

given by equation (5.1) is called a control-affine system [57]. Such systems are nonlinear in their dynamics, but they are linear concerning the applied control actions. Control-affine systems have a wide area of application in nonlinear control theory and have been studied thoroughly mainly by [58], [59]. Furthermore mathematical theory of nonholonomic systems is mainly restricted to control-affine systems [57]. The term $f(x)$ in equation (5.1) is called *the drift term* and complicates the systems in the sense of controllability. However, because of the existence of the drift term it is not necessarily possible to make the derivatives of the states vanish [57].

In the case of our modular robotic model, assuming that it has been successfully represented in the form given by equation (5.1), having a *drift* means that, during the continuous locomotion phase where the controllers will be working to satisfy some control objective set, $\dot{x}$ is not going to vanish. This is because, in order to make $\dot{x}$ equal to zero, it necessary to set the controllers such that $\sum_{i=1}^{m} g_i(x)u_i = -f(x)$ holds, which will conflict with the determined control objectives (unless it explicitly to stop the robot). Hence, it will not be necessary to deal with exact modeling of f_i^R around the region $v_i = 0$ to have a realistic simulation. That will be the case even if somehow it is attempted to stop the robot while it is moving because by the time the condition $v_i = 0$ holds, the robot locomotion will have already stopped.

Referring to the final reduced equations of motion given by (4.25), there are two coupled nonlinear second order differential equations with two unknowns x_h, y_h. Simply making the following variable transformations

$$\begin{aligned} x_1 &= x_h & x_2 &= y_h \\ x_3 &= \dot{x}_h & x_4 &= \dot{y}_h \end{aligned} \tag{5.2}$$

state space variables $x = \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{bmatrix} = \begin{bmatrix} r \\ \dot{r} \end{bmatrix}$ in $\mathbb{R}^4$ are obtained. Equation (4.25) and equation (5.2) lead to

$$\dot{x}_1 = x_3 \tag{5.3}$$

$$\dot{x}_2 = x_4 \tag{5.4}$$

$$\begin{bmatrix} \dot{x}_3 \\ \dot{x}_4 \end{bmatrix} = M'(\theta)^{-1}\left(\left[F^T I_2 \right] Z(\theta) f^R + F^T E u - \left(C'(\dot{\theta},\theta) + N' \right) \begin{bmatrix} x_3 \\ x_4 \end{bmatrix} \right) \tag{5.5}$$

Hence, the system has been successfully converted to the standard form $\dot{x} = f(x) + g(x)u$, where from equations (5.3), (5.4) and (5.5)

$$f(x) = \begin{pmatrix} \begin{matrix} 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{matrix} \\ M'(\theta)^{-1} \begin{bmatrix} 0 & 0 \\ 0 & 0 \end{bmatrix} - \left(C'(\dot{\theta},\theta) + N' \right) \end{pmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{bmatrix} + \begin{bmatrix} 0 \\ 0 \\ M'(\theta)^{-1} \left[F^T I_2 \right] Z(\theta) f^R \end{bmatrix} \tag{5.6}$$

$$g(x) = \begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ M'(\theta)^{-1} F^T E \end{bmatrix} \tag{5.7}$$

where in equation (5.6) the expression $-\left(C'(\dot{\theta},\theta) + N' \right)$ is a 2×2 matrix while the expression $\left[F^T I_2 \right] Z(\theta) f^R$ is a 2×1 vector.

5.2 Control Objectives

The main control objectives for the robot can be listed as follows

1) To generate a sustainable *successful* forward locomotion of the robot.

2) To make the head follow *any* prescribed *feasible*, *smooth enough* path during the locomotion.

3) To avoid the singular configuration whose description is given in section 3.3.

4) To prevent the side slippage of *each of the robot modules* during the locomotion.

These control problems are fundamentally different from those that kinematically and structurally look similar. In literature, these mechanical systems are called wheeled mobile robots (WMR) and they can be in a single unit or chained forms like n-link trailers. In WMRs, position and orientation (and/or their time derivatives) of the leading module are chosen to generate the locomotion as in [60], [61], [62], [63] and [64]. In traditional WMRs, the attached wheels are driven and the torques applied to them are the main sources of actuation. However, in our case, the wheels are passive without any actuation. Due this fact, the main difficulty arises from the problem of avoiding the singular configuration which occurs when $E^T F$ becomes rank deficient as explained before. The first objective stated above is closely related to the fourth objective as the occurrence of singularity makes the robot cease. This explains the notion *sustainable* in the first objective. On the other hand, one can attach countless many meanings to the word *successful* concerning the locomotion. What it is meant by a *successful* locomotion actually will be made clearer during the following chapters. Namely its meaning will be quantified by defining a relevant and sound performance measure for success. The second objective clearly states that the robot needs to follow a

feasible desired path in $\mathbb{R}^2$. By *feasible*, it is meant that the curvature of the desired path should not breach the structural geometric limitations i.e., it should not enforce the robot modules to collide with each other. Thus, one of the objectives is to make the robot head track the given reference trajectory in task space.

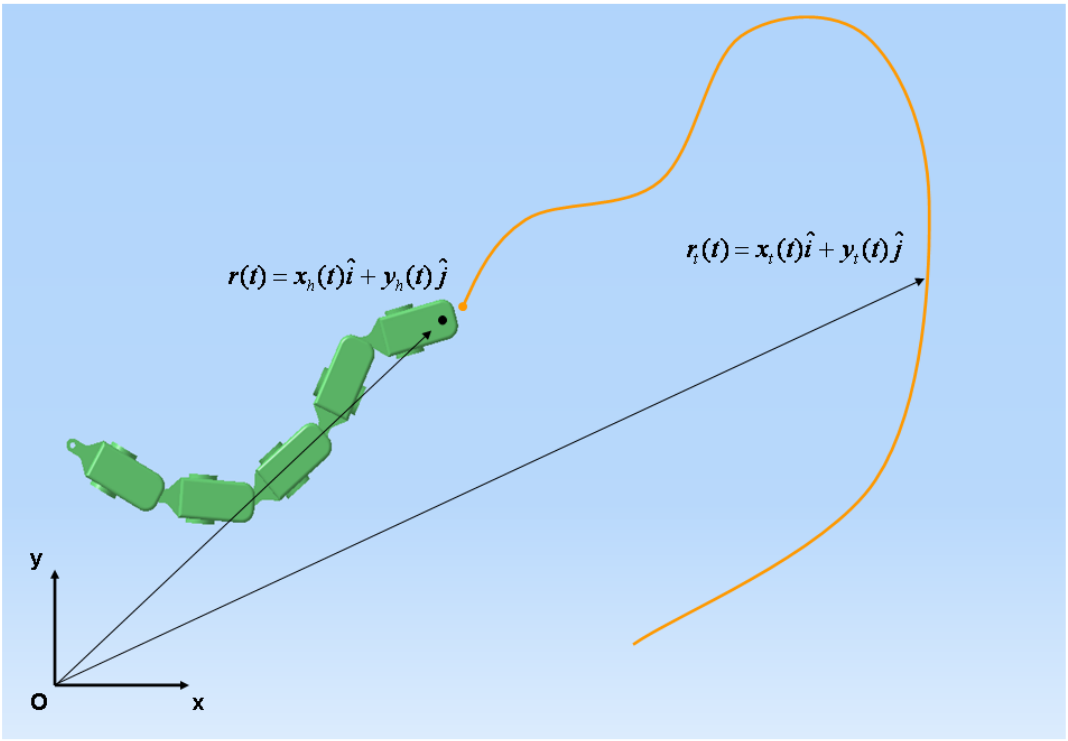


Figure 5.1: Illustration of the head position and the task trajectory to track

In Figure 5.1, the robot head position vector $r(t)$ and the reference task trajectory $r_t(t)$ to be tracked are illustrated. Here the parametric argument t is not necessarily the time. The global control objective is to minimize the error between $r(t)$ and $r_t(t)$. In a standard reference tracking or regulation problem this objective could be expressed in the following form

$$e(t) \to 0 \quad \text{as} \quad t \to \infty \qquad \text{where} \qquad e(t) = \left\| r(t) - r_t(t) \right\| \qquad (5.8)$$

Here in equation (5.8), *t* explicitly stands for time. Equation (5.8) actually dictates the exponential stability for $e(t)$. However, in a challenging problem like ours, where it is attempted to track "any" feasible reference task trajectory, it is necessary somehow to relax the requirements. That is, since it is known that $r_t(t)$ might change its direction and radius of curvature abruptly, at some time instant during the locomotion actuators installed on the robot may not suffice to make equation (5.8) hold. They can only guarantee that $r(t)$ is not far away from $r_t(t)$. Hence, equation (5.8) is modified as follows

$$e(t) \le \delta \qquad \forall t \ge T \quad \text{where} \quad \delta, T \in \mathbb{R}^+ \ \text{and} \ \ e(t) = \left\| r(t) - r_t(t) \right\| \qquad (5.9)$$

Equation (5.9) [65] states that $e(t)$ at least never diverges, (for all times during the locomotion) since δ is a bounded real positive number. Actually equation (5.9) requires the stability of $e(t)$ in the sense of Lyapunov. Selection of δ determines the performance requirements from the control system. The fourth objective is actually the verification of our previous assumption that none of the modules side-slips. The wheels are employed to prevent side slipping (nonholonomic kinematic constraint) and to reduce the friction force acting along the longitudinal direction. The maximum amount of normal frictional force that the ground can apply to a module of the robot $f^N_{\max}$ is limited and is simply given by

$$f^N_{\max} = mg\mu_N \qquad (5.10)$$

The constraint forces which ensure equation (3.23) holds for all times are generated by the normal friction forces applied by the ground. Hence, it is necessary to make sure that these constraints forces never exceed $f^N_{\max}$. If any of the constraint forces required to make equation (3.23) hold exceeds $f^N_{\max}$, no-slip condition will be violated and derived equations of motion for the robot will no longer be valid. Furthermore the violation of no side-slip condition will lead to energy dissipation due to the fact that friction, which is a nonconservative force, will then do some work against the direction of motion. None of the mentioned

papers have elaborated the scope of validity of the no-side slip assumption. This is due their assumption that the ground can always apply the required friction force in order to prevent the modules from side slipping. However, this assumption may not be correct always (for example on slippery and muddy grounds or when the actuators apply a large torque or when the radius of curvature of the reference trajectory turns out to be small).

5.3 Feedback Linearization and Control Law

Mechanical systems that have less number of control inputs, *m,* than their degree of freedom *n* are called underactuated systems [66]. In our case *n* is seven and *m* is four as have been previously stated. Thus, without the consideration of the nonholonomic no-slip constraints of the first order , our robotic system can also be considered to be an underactuated one. However, due to the reasons stated in [67], the robot with the reduced equations of motion given by equation (4.25) becomes an over-actuated one. This is because in our case $m+k>n$, where $m=4$ is the number of control inputs given by equation(4.2), $k=5$ is the number of nonholonomic constraints given by equation (3.15) and $n=7$ is the number of configuration variables given by equation (3.8). Mechanical systems with nonholonomic constraints that become over-actuated when reduced have been discussed in [68].

Authors of [24] applied a Lyapunov based PD control approach while authors in [25] applied an acceleration based control where the head of the robot is made to follow a desired acceleration based on a newly proposed manipulability. Briefly, manipulability can be described as a measure to determine how far the robot is from the singular configuration. Authors of [69] defined a performance measure which mainly includes the head position error and the nonholonomic constraint forces imposed on the wheels. Then, a full state feedback control law which is obtained by the solution of the well known Riccati equation is applied. Here, the control law is developed based on a control technique known as feedback

linearization. Basically it is the transformation of a nonlinear system to a linear one by applying an appropriate full-state nonlinear feedback. It is rather different from a conventional linearization procedure where the equations of motion of the system are approximated by linear ones near the anticipated operating points. Feedback linearization is mathematically *exact* and represents the system dynamics without any approximation. In certain multibody robotic applications and WMR's it reduces to the well known computed torque control method. However, it requires the exact knowledge of the system dynamics and can be computationally expensive especially when the control action is to be applied *online.* Furthermore it may require inputs with high magnitudes from the actuators. However, for the case when the dynamical systems to be controlled are robotic manipulators or WMR's, due to the boundedness of the inertia matrix the required control torques remain always bounded except for the singularity state. What is more, computational resources and technology available today are vast and exponentially increasing and they can meet the online computational demands. Furthermore, several reports from experimental results indicate a good tracking performance and this method is being widely applied nowadays [43].

Consider an affine nonlinear time invariant system in the form

$$\dot{x} = f(x) + g(x)u; \qquad x \in \mathbb{R}^n,\ u \in \mathbb{R}^m \tag{5.11}$$

where $f(x)$ and $g(x)$ are smooth and differentiable vector fields on $\mathbb{R}^n$ and $f(0) = 0$. Using the definition in [54], for the input $u \in \mathbb{R}$ and the output $y = h(x) \in \mathbb{R}$ (i.e., for the SISO case) the system given by equation (5.11) is said to be feedback linearizable if there exists a region $U \in \mathbb{R}^n$ containing the origin, a diffeomorphism[4] $T : U \to \mathbb{R}^n$ and nonlinear feedback in the form

$$\upsilon = \alpha(x) + \beta(x)u \tag{5.12}$$

[4] A diffeomorphism is a smooth (C^∞) locally invertible function such that it is differentiable, and its inverse exists and is also differentiable.

with $\beta(x) \neq 0$ on U such that transformed variables

$$z = T(x) \tag{5.13}$$

satisfy the system of equations

$$\dot{z} = Az + b\upsilon \tag{5.14}$$

where

$$A = \begin{bmatrix} 0 & 1 & 0 & & 0 \\ 0 & 0 & 1 & & . \\ & & & & . \\ & & & & . \\ & & & & 1 \\ 0 & 0 & & 0 & 0 \end{bmatrix} \qquad b = \begin{bmatrix} 0 \\ 0 \\ . \\ . \\ . \\ 1 \end{bmatrix} \tag{5.15}$$

The linear system given by equation (5.14) is the linearized form of equation (5.11) [54] and it is called controllable canonical form. Here the new transformed variables z are given by equation (5.16):

Lemma: If $y = h(x) \in \mathbb{R}$ is the output of the system (5.11) and linearizing the input-output of the system is the point of interest, the transformation function which linearizes the system is given by [70]

$$\begin{aligned} z_1 &= y = L_f^0 h(x) = T_1(x) \\ z_2 &= \dot{y} = L_f h(x) + L_g h(x)u = L_f h(x) = T_2(x) \\ z_3 &= \ddot{y} = L_f^2 h(x) + L_g L_f h(x)u = L_f^2 h(x) = T_3(x) \\ &\vdots \\ z_n &= y^{(n-1)} = L_f^{(n-1)} h(x) + L_g L_f^{(n-2)} h(x)u = L_f^{(n-1)} h(x) = T_n(x) \\ &\textit{and} \\ \dot{z}_n &= y^{(n)} = L_f^{(n)} h(x) + L_g L_f^{(n-1)} h(x)u = \upsilon \end{aligned} \tag{5.16}$$

where n is the relative degree[5] of $y = h(x)$ with respect to u around $x = 0$, $L_f h(x)$ is the Lie derivative of $h(x)$ in the direction of vector field $f(x)$ given by the following definition.

Definition: If $f : \mathbb{R}^n \to \mathbb{R}^n$ is a smooth vector field on $\mathbb{R}^n$ and $h : \mathbb{R}^n \to \mathbb{R}$ is scalar, then Lie derivative is given by [54]

$$\begin{aligned} L_f h(x) &= \frac{\partial h}{\partial x} f(x) = \sum_{i=1}^{n} \frac{\partial h}{\partial x_i} f_i(x) : \mathbb{R}^n \to \mathbb{R} \\ L_f^k h &= L_f (L_F^{k-1} h) \quad \text{for } k = 1, \ldots, n \\ L_f^0 h &= h \end{aligned} \tag{5.17}$$

Furthermore in equation (5.16),

$$\begin{aligned} &L_g L_f^0 h = L_g L_f^1 h = \ldots L_g L_f^{n-2} h = 0 \\ &L_g L_f^{n-1} h \neq 0 \end{aligned} \tag{5.18}$$

That means $T_1(x), \ldots T_n(x)$ are linearly independent of each other and of u, while $y^{(n)} = \upsilon = L_f^{(n)} h(x) + L_g L_f^{(n-1)} h(x) u$ is the first derivative of the output which explicitly depends on the input u. Necessary and sufficient conditions under which the system given by equation (5.11) can be transformed to the one given by equation (5.14), derivation and proofs of the results given here can be found in [49], [58], [71], [65] and [72]. Hence, υ, which is called *exogenous input* or *synthetic input*, is simply a chain of integrators on the system output $y = h(x)$ in the manner $\upsilon = \frac{\partial^n y(t)}{\partial t^n}$. Since it also stands as a linear input to the linearized (transformed) system given by equation (5.14), it can be designed to satisfy the tracking or regulation requirements. Furthermore, equations (5.12) and (5.16) lead to

[5] Definion of the relative degree will be given in Appendix A

$$\alpha(x) = L_f^n h(x)$$
$$\beta(x) = L_g L_f^{(n-1)} h(x) \tag{5.19}$$

Thus, once the synthetic input υ is designed, the actual required control input *u* can be obtained via the equation

$$u = \frac{1}{\beta(x)}\left[\alpha(x) - \upsilon\right] \tag{5.20}$$

Now, let *r* denote the relative degree of the system. In the case where *r* is less than the number of generalized coordinates (i.e. $r < n$) the nonlinear transformation given by equation (5.16) is able to transform only *r-many* coordinates. The remaining *n-r* coordinates which are generally called *the internal states*, can be found without any difficulty from the requirement that *T(x)* must be a diffeomorphism. However, when $\upsilon(t)$ is designed such that $z_1, \ldots z_r$ converges to their desired values, control over those internal states $\zeta \in \mathbb{R}^{n-r}$ is lost such that they may even diverge. This problem is called internal stability and is mainly discussed by [71] and [49]. However, the details will not be given here since there will not be such a case in our problem.

So far, the discussion about the input-output linearization of the system given by equation (5.11) has been limited to the single input-single output (SISO) case. However, when the reduced equations of motion given by equation (4.25) is observed, it should be noted that there are four distinct inputs. Furthermore, when the control or stabilization of the robot head around a desired trajectory is of interest, the head position is a natural choice for the output of the system. Thus the output can be defined to be

$$y = \begin{bmatrix} y_1 \\ y_2 \end{bmatrix} = \begin{bmatrix} h_1(x) \\ h_2(x) \end{bmatrix} = r = \begin{bmatrix} x_h \\ y_h \end{bmatrix} \in \mathbb{R}^{2\times 1} \tag{5.21}$$

Hence, the system under consideration is a multi input-multi output (MIMO) one. Extension of input-output linearization to the square MIMO systems is discussed

in [59]. Generally, a square MIMO system is the one which has equal number of inputs and outputs. A necessary condition that input-output linearization method can be applied to square systems is that *the decoupling matrix* must be invertible. However, the system dynamics given by equations (5.11) and (5.21) is not a square system since there are four inputs and two outputs. The cases where the MIMO systems are not square are discussed in [58] and [73] where it has been stated that a static state feedback can be applied to control the system provided that *the decoupling matrix* is not rank deficient. The background of input-output linearization and dynamic input-output decoupling can be found in [59]. Having said that, for the first output $y_1 = h_1$, equations (5.2), (5.3) and (5.21) yield

$$\begin{aligned} y_1 &= x_h = L_f^0 h_1 = x_1 \\ \dot{y}_1 &= \dot{x}_h = L_f h_1 + \sum_{i=1}^{m} (L_{g_i} h_1) u_i = \dot{x}_1 = x_3 \\ \ddot{y}_1 &= \ddot{x}_h = L_f^2 h_1 + \sum_{i=1}^{m} (L_{g_i} (L_f h_1)) u_i = \ddot{x}_1 = \dot{x}_3 \end{aligned} \tag{5.22}$$

Furthermore, for the second output $y_2 = h_2$, equations (5.2), (5.4) and (5.21) yield

$$\begin{aligned} y_2 &= y_h = L_f^0 h_2 = x_2 \\ \dot{y}_2 &= \dot{y}_h = L_f h_2 + \sum_{i=1}^{m} (L_{g_i} h_2) u_i = \dot{x}_2 = x_4 \\ \ddot{y}_2 &= \ddot{y}_h = L_f^2 h_2 + \sum_{i=1}^{m} (L_{g_i} (L_f h_2)) u_i = \ddot{x}_2 = \dot{x}_4 \end{aligned} \tag{5.23}$$

From equations (5.22), (5.23) and (5.5) it is seen that, input u appears for the first time in the second derivatives of y_1 and y_2. Hence, it can concluded that our MIMO system has *a vector relative degree* $(2,2)$. Thus, taking the second derivative of each output, the following vector equations are obtained:

$$\begin{bmatrix} \upsilon_1 \\ \upsilon_2 \end{bmatrix} = \begin{bmatrix} \ddot{y}_1 \\ \ddot{y}_2 \end{bmatrix} = \begin{bmatrix} \dot{x}_3 \\ \dot{x}_4 \end{bmatrix} = \begin{bmatrix} L_f^2 h_1 \\ L_f^2 h_2 \end{bmatrix} + B(x) \begin{bmatrix} u_1 \\ u_2 \\ u_3 \\ u_4 \end{bmatrix} \tag{5.24}$$

where $B(x) \in \mathbb{R}^{2\times 4}$ is the *decoupling matrix* with full rank in order to have a defined static feedback given by equation (5.24) [73]. That means it is necessary for the two rows of $B(x)$ to be linearly independent from each other. Substituting equation (5.5) into equation (5.24) the following equations are obtained

$$\begin{bmatrix} L_f^2 h_1 \\ L_f^2 h_2 \end{bmatrix} = M^{'}(\theta)^{-1} \left(\begin{bmatrix} F^T I_2 \end{bmatrix} Z(\theta) f^R - \left(C^{'}(\dot{\theta}, \theta) + N^{'} \right) \begin{bmatrix} x_3 \\ x_4 \end{bmatrix} \right) \tag{5.25}$$

$$B(x) = M^{'}(\theta)^{-1} F^T E \tag{5.26}$$

Thus, equations (5.14) and (5.24) yield the following input-output linearized system

$$\begin{bmatrix} \dot{z}_{1_1} \\ \dot{z}_{1_2} \\ \dot{z}_{2_1} \\ \dot{z}_{2_2} \end{bmatrix} = \begin{bmatrix} 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} z_{1_1} \\ z_{1_2} \\ z_{2_1} \\ z_{2_2} \end{bmatrix} + \begin{bmatrix} 0 & 0 \\ 0 & 1 \\ 0 & 0 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} \upsilon_1 \\ \upsilon_2 \end{bmatrix} \tag{5.27}$$

where

$$\begin{bmatrix} z_{1_1} \\ z_{1_2} \\ z_{2_1} \\ z_{2_2} \end{bmatrix} = \begin{bmatrix} y_1 \\ \dot{y}_1 \\ y_2 \\ \dot{y}_2 \end{bmatrix} = \begin{bmatrix} h_1 \\ \dot{h}_1 \\ h_2 \\ \dot{h}_2 \end{bmatrix} = \begin{bmatrix} x_1 \\ x_3 \\ x_2 \\ x_4 \end{bmatrix} = \begin{bmatrix} x_h \\ \dot{x}_h \\ y_h \\ \dot{y}_h \end{bmatrix} \tag{5.28}$$

are the transformed coordinates given by equations (5.22), (5.23), (5.25) and (5.26) . It should be noted that the linearized system given by equation (5.27) is the

augmentation of the SISO case equation (5.14). Letting r_j denote the relative degree of the j^{th} output of *p-many* outputs, by the definition a total of $\sum_{j=1}^{p} r_j$ transformed coordinates are obtained. However, as have been shown in equation (5.22) and (5.23) , there exists *a relative degree of two* for each of the outputs y_1 and y_2. Thus, $\sum_{j=1}^{p} r_j = 4$, which equals to the number of state variables of the reduced system represented in the state-space by equations (5.3), (5.4) and (5.5). If this is the case, all of the original states of the system given by equations (5.3), (5.4) and (5.5) are observable in the *transformed, decoupled and linearized* system given by equation (5.27). This means there is no internal dynamics in the linearized control system, i.e. as long as the system is controlled such that the transformed states given by equation (5.28) stay bounded, the original system states will also be bounded [59]. That is to say, stability of the transformed system ensures the stability of the original nonlinear system.

Since the aim is to track a given smooth feasible trajectory in the task space, the synthetic input υ can now be readily designed as follows

$$\begin{aligned} \upsilon_1 &= \ddot{x}_t - K_v^x(\dot{x}_h - \dot{x}_t) - K_p^x(x_h - x_t) \\ \upsilon_2 &= \ddot{y}_t - K_v^y(\dot{y}_h - \dot{y}_t) - K_p^y(y_h - y_t) \end{aligned} \tag{5.29}$$

The task space error e_t can be defined as

$$\begin{aligned} e_t^x &= x_h - x_t \\ e_t^y &= y_h - y_t \end{aligned} \tag{5.30}$$

where $r_t = x_t\hat{i} + y_t\hat{j}$ is the parametric reference trajectory for the robot head. From equations (5.2), (5.3), (5.4) , (5.24), (5.29) and (5.30), the following error dynamics in the form of linear second order differential equations are obtained

$$\ddot{e}_t^x + K_v^x \dot{e}_t^x + K_p^x e_t^x = 0$$
$$\ddot{e}_t^y + K_v^y \dot{e}_t^y + K_p^y e_t^y = 0 \qquad (5.31)$$

where K_P^x, K_v^x and K_P^y, K_v^y are the proportional and velocity gains for e_t^x and e_t^x, respectively. As have been mentioned earlier, the problem of tracking control of the modular robot by input-output feedback linearization is reduced to the *computed torque control* method in the means of error dynamics. K_P^x, K_v^x and K_P^y, K_v^y determines how the errors behave during the locomotion of the robot and they are the design parameters which affect how well the robot tracks the given trajectory. Once the control law for the synthetic input υ given by equation (5.29) is designed, the actual control input from the equation (5.24) can be generated. When equation (5.24) is observed, it is seen that u cannot be isolated. That is because, a left inverse of the decoupling matrix $B(x) \in \mathbb{R}^{2\times 4}$ does not exist since it has more columns than rows, it only has a right inverse [46]. Utilizing this fact, *u* can be synthesized such that equation (5.24) trivially holds. Hence, the actual control input can defined to be

$$u = B(x)^+ \left(\upsilon - \begin{bmatrix} L_f^2 h_1 \\ L_f^2 h_2 \end{bmatrix} \right) \qquad (5.32)$$

where $B(x)^+ = \left(F^T E\right)^+ M'(\theta) \in \mathbb{R}^{4\times 2}$ is the right inverse of $B(x)$. Expanding equation (5.32) to its most explicit form, the following feedback control law is obtained

$$u = \begin{bmatrix} u_1 \\ u_2 \\ u_3 \\ u_4 \end{bmatrix} = \left(F^T E\right)^+ M'(\theta)\upsilon - \left(F^T E\right)^+ \left(\left[F^T I_2\right] Z(\theta) f^R - \left(C'(\dot{\theta},\theta) + N'\right) \begin{bmatrix} x_3 \\ x_4 \end{bmatrix} \right) \qquad (5.33)$$

with υ to be designed as in equation (5.29). Once K_p and K_v is selected, the synthetic input υ is available, and once υ is available the actual control torques to be applied by the actuators is generated according to the law given above by

equation (5.33). That is to say, by simply selecting K_p and K_v the nonlinear robotic system can controlled so as to make it follow the desired trajectory, unless the singularity state is encountered during the locomotion.

The above control strategy can be thought of as two interlacing control loops interacting through coordinate transformation. The overall control architecture of the robot is given in Figure 5.2

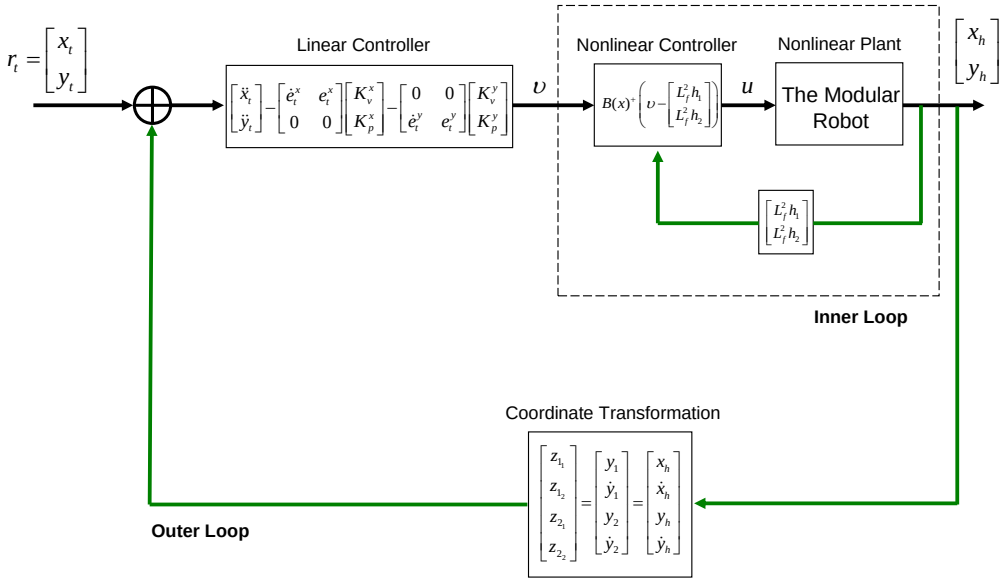


Figure 5.2: Overall control architecture of the modular robot

In Figure 5.2, the inner loop represents the nonlinear dynamics of the system which includes the nonlinear controller derived from equation (5.32) and the nonlinear plant. The plant is the nonlinear equations of motion of the robot which were given by equation (4.25). The outer loop of the above control architecture is basically the linearized (transformed) system dynamics and consists of the coordinate transformation, reference input and the linear controller. In this control

block diagram, for reasons of simplicity, output channels of the transformation blocks represents directly the expressions denoted in the blocks themselves. That is to say, for example, instead of interpreting u as the multiplication of the synthetic input υ by the block denoted by *Nonlinear Controller* in Figure 5.2, it should be understood that u actually equals to the value of the *Nonlinear Controller* itself. As expected, the inner and outer loops have a common output which is the head position of the robot.

As have been mentioned earlier by referring to [43] and [54], the feedback linearization method simply reduces to the computed torque control method in the case of rigid manipulators. However, it should be noted that by applying the feedback linearization method an interesting feature in the system dynamics is observed. In equation (5.24), for an explicit expression of actual control input u it was necessary to obtain the left pseudo-inverse of the decoupling matrix $B(x)$ which does not exist. Hence, u is designed to be as in equation (5.32) such that equation (5.24) holds. This implies that the control law that has been found is not unique. This arises from the fact that although through the application of nonholonomic constraints the system coordinates have been reduced to two, there are still four independent control inputs to apply just like in the original system equations. However, it is a quite natural choice and resembles the fashion in which the control input is selected such that the time derivative of the Lyapunov function is negative definite in application of the Lyapunov based nonlinear control. By merely applying the computed torque method this fact could not be noticed. Also based on these results, one can attempt to analyze the possible non rigid dynamics cases in further studies.

5.4 Stability and Controllability

Stability and controllability of a dynamic control system is of uttermost importance since there would be no point in analyzing the system dynamics and

synthesizing a controller for the overall system if one, somehow, shows that the plant is unstable and/or uncontrollable in the region of interest.

Although, physically, it may seem that the robotic system is obviously stable, it is nevertheless necessary to show that it indeed is, for the purposes of rigor and completeness. Stability of a control system implies that all of its states remain bounded during its operation. The states of the system is defined to be $x = \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{bmatrix} = \begin{bmatrix} x_h \\ y_y \\ \dot{x}_h \\ \dot{y}_y \end{bmatrix}$. Hence, through the application of control torques dictated by equation (5.33), if it is managed to make the head position of the robot bounded, the states x_1 and x_2 also become bounded. Also, from equation (5.5) it is observed that the states x_3 and x_4 will also be bounded provided that the control input u and the robot head position are stable (bounded). This is because the reduced inertia matrix $M'(\theta) \in \mathbb{R}^{2\times 2}$ is positive definite, invertible and bounded. Thus to summarize, in order to have a stable system the following conditions must hold

1) $|x_h(t)| < \delta \quad |y_h(t)| < \delta \quad \forall t;\ \delta \in \mathbb{R}^+,\ t \geq 0$

2) $|u(t)| < \varepsilon \quad \forall t;\ \varepsilon \in \mathbb{R}^+,\ t \geq 0$

If it can be shown that the head position of the robot follows a bounded reference trajectory with bounded tracking error, it will be sufficient to ensure that the first condition holds. In order that the second condition for stability holds, from equation (5.33) it is seen that $(F^T E)^+$, which is the right pseudo-inverse of $F^T E$, must exist. For that, $F^T E$ needs to have a full rank. Noting that $F^T E$ is the transpose of the manipulator Jacobian given in equation (3.25), it is concluded that the second condition automatically reduces to the avoidance of singularity condition explained section 3.3. In order to show that the tracking error is bounded, consider the following proposition.

Proposition: If $K_p = \begin{bmatrix} K_p^x & 0 \\ 0 & K_p^y \end{bmatrix}$, $K_v = \begin{bmatrix} K_v^x & 0 \\ 0 & K_v^y \end{bmatrix} \in \mathbb{R}^{2\times 2}$ are symmetric, positive definite matrices, then the control law given by equation (5.33) applied to the system given by equation (5.1) leads to exponential tracking [43].

The proof of the above proposition is straight forward and is given in [43] which can be found also in Appendix A. It states that, by a proper selection of position and velocity gains, it can be guaranteed that error dynamics given by equation (5.31) converges to zero in the steady state. Hence, as long as proper gains for error dynamics are selected, the first condition for stability holds. Although it is only required that the tracking error is contained in a bounded region (marginal stability), according to the above preposition, a suitable selection of gains leads to absolute stability.

If the nonlinear system given by equation (5.11) can be transformed to the linearized system given by equation (5.27) through feedback linearization, then the system is controllable. This is rather obvious, because the linearized system given by equation (5.27) is the augmentation of two *controllable canonical* systems given by equation (5.14). Thus, the transformed linearized system given by equation (5.27) automatically satisfies Kalman's necessary and sufficient condition for controllability. Through straight forward application of conditions to our system under which the feedback linearization can be realized, controllability can be proven. However, without checking these mentioned conditions, input-output linearization to the system has been directly applied and the linearized system is successfully obtained. Recall from equation (5.33) that, in order to cancel nonlinearities (i.e. to achieve feedback linearization) existence of $(F^T E)^+$ is again a necessity. Hence, avoidance of the singularity condition turns out to be a required condition for both stabilizability and controllability. This shows us again that avoidance of singularity is of prime importance.

5.5 Convergence to Singularity

As have been stated before, a proper selection of the design parameters K_p and K_v guarantees that the controlled robotic system is stable and controllable during its course tracking the given task trajectory as long as avoidance of the singularity configuration is attained. It is necessary to prevent the modules from converging to the singular configuration which occurs when $\varphi_1 = \varphi_2 = \varphi_3 = \varphi_4 \pmod{\pi}$. Consider now the following lemma:

Lemma: When $\dot{r} = constant$, $\varphi_1 = \varphi_2 = \ldots = \varphi_{m-1} = 0 \pmod{\pi}$ *becomes a stable equilibrium of* $\dot{\theta} = F(\theta)\dot{r}$ [24].

The above lemma means that when it is attempted to control the velocity of the head of the robot to be constant, the robot becomes straight as depicted in Figure 3.3 a). Based on the equations of motion of the robot given by equations (5.3), (5.4) and (5.5), a Matlab Simulink model has been built. The control law given by equation (5.33) is applied to this model in order to track a given reference input r_t . The schematics of the Matlab Simulink model and related m-files are given Appendix B. In this section, the simulation of the robot will be dealt with when it is given an ordinary path to follow. Then, the general tracking behavior and the normal friction forces f_i^N, $i = 1,\ldots,5$ applied by the ground to the individual modules during the simulation will be observed. As it shall be recalled f_i^N are the constraint forces that prevent the modules from side slipping. In order to find these forces, it is necessary to obtain the Lagrangian multipliers $\lambda(t)$ during the simulation time. Observing equation (4.10) and referring to the expression and derivation to obtain $\lambda(t)$ for robotic manipulators in [43]. it is seen that the following equation gives $\lambda(t)$ at any simulation time in our case:

$$\lambda = -\left(AM^{-1}A^{T}\right)^{-1}\left(AM^{-1}\left(Q+\begin{bmatrix} Eu \\ 0 \\ 0 \end{bmatrix}-(C+N)\dot{q}\right)+\dot{A}\dot{q}\right) \tag{5.34}$$

where A is given by equation (3.23) and Q is given by equation (4.19). Thus the Lagrangian multipliers can be calculated as a function of the configuration variable q, its time derivative $\dot{q}$, applied control torques u and the tangential friction coefficient μ_R. Using state space equations of the motion given by equations (5.3), (5.4) and (5.5) together with the nonholonomic constraint equation given by (3.22) (and also with a given initial condition $q = q_0$) the configuration variable q and $\dot{q}$ can be obtained as a function of time. Thus, then they can be substituted into equation (5.34) to obtain the Lagrangian multipliers λ. Using the principle of virtual work and referring to the result in [25], the vector of normal forces applied by the ground to modules f^N given by equation (4.3) is the obtained as follows:

$$f^N = \left(F_A{}^T\right)^{-1}\lambda \tag{5.35}$$

where F_A is given by equation (3.20). If equation (5.34) is observed, it is recognized that part of the constraint force originates from the control torques u and the rest is due to the velocity terms $\dot{q}$. During a singularity condition the norm of the constraint force becomes very large (diverges) and the locomotion ceases. Through our Matlab Simulink model it is illustrated how the robot converges to the singular configuration when the head is given a definite path with a constant velocity as covered in the above Lemma. Firstly, the physical parameters of the robot are defined as follows:

m (mass of a single module) = 0.792 kg, l = 0.1 m, J = 0.00269 kg.m^2, $\mu_R = 0.035$, $d_\varphi = 0.01$.

Also from equation (5.31), it is seen that K_p^x, K_v^x and K_p^y, K_v^y determine the error dynamics e_t^x and e_t^y, respectively. That is, selection of K_p^x, K_v^x and K_p^y, K_v^y determines whether the error dynamics will be underdamped, critically damped or overdamped. Generally, a sluggish responding system is not desired but at the same time overshoot is not something desirable when the objective is to follow a trajectory when obstacles are around the way. Hence, designing a critically damped system is generally a good and simple engineering practice. That is because, when perturbed, critically damped systems will reach the steady-state value in minimum time without any overshoot. So, setting the damping coefficients of the linear second order system given by equation (5.31) to unity, i.e. $\zeta^x = \zeta^y = 1$, equation (5.31) leads to

$$\begin{aligned} K_p^x &= \omega_x{}^2, K_v^x = 2\omega_x \\ K_p^y &= \omega_y{}^2, K_v^y = 2\omega_y \end{aligned} \tag{5.36}$$

where ω_x and ω_y are the respective natural frequencies of the respective error dynamics which will determine their rate of decay. Equivalently, ω_x and ω_y determine the speed of response to the location perturbation of the head position from its desired value. Then there are double eigenvalues placed at $-\omega_x$ for e_t^x and double eigenvalues placed at $-\omega_y$ for e_t^y. In the first simulation the natural frequencies are $\omega_x = \omega_y = 2\ rad/s$, meaning that equal weights are put on tracking in the x and y directions. Initially the modular robot is at rest, i.e. $\dot{x}_h(0) = \dot{y}_h(0) = \dot{\theta}_1(0) = \dot{\theta}_2(0) = \dot{\theta}_3(0) = \dot{\theta}_4(0) = \dot{\theta}_5(0) = 0$ as the initial condition of the configuration variables. This will be the case for all of the following simulation cases in this study. Also, the initial conditions of the configuration variables of the robot are $x_h(0) = 5$, $y_h(0) = 10$ and $\theta_1(0) = 10^\circ$, $\theta_2(0) = -10^\circ$, $\theta_3(0) = 20^\circ$, $\theta_4(0) = -15^\circ$, $\theta_5(0) = 5^\circ$. It is required that the head of the robot follows the task trajectory given by $x_t(t) = -0.05t + 4.9$ and $y_t(t) = 10.1$, where the coordinates are

in meters and *t* denotes time[6] in seconds. That is to say, it is required that the robot goes straight to the left with a velocity of 5 cm/s and with some initial position offset of 0.1 m. Simulation is performed for 25 seconds using the developed Matlab Simulink model based on our previously constructed equations of motion and control law.

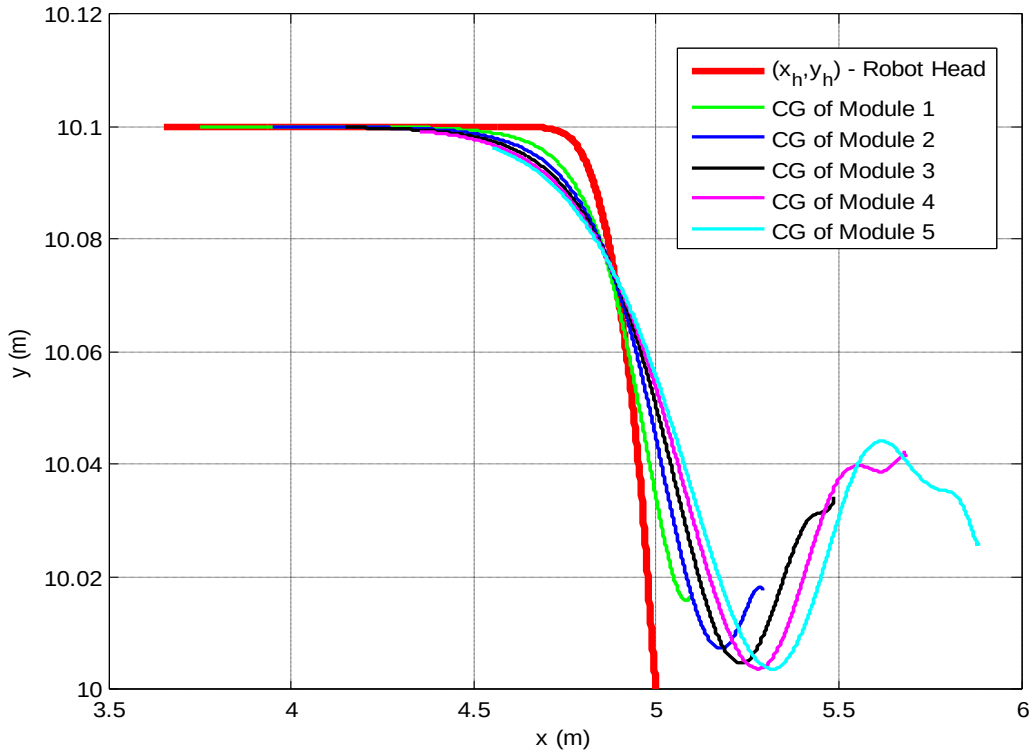


Figure 5.3: Trajectory of the robot head and modules following a straight line with constant velocity reference

In Figure 5.3, trajectories of the robot head (x_h, y_h) and of the center of gravities of five modules $(x_1, x_2, x_3, x_4, x_5)$ are plotted starting from the initial condition for 25 seconds.

[6] Unless otherwise stated explicitly, the variable *t* in expressions $r_t(t)$, $x_t(t)$, $y_t(t)$, $\dot{x}_t(t)$ and $\dot{y}_t(t)$ (which are related with the parametric equation of the task trajectory) does not denote time, throughout this study. Instead, it denotes a generic parameter.

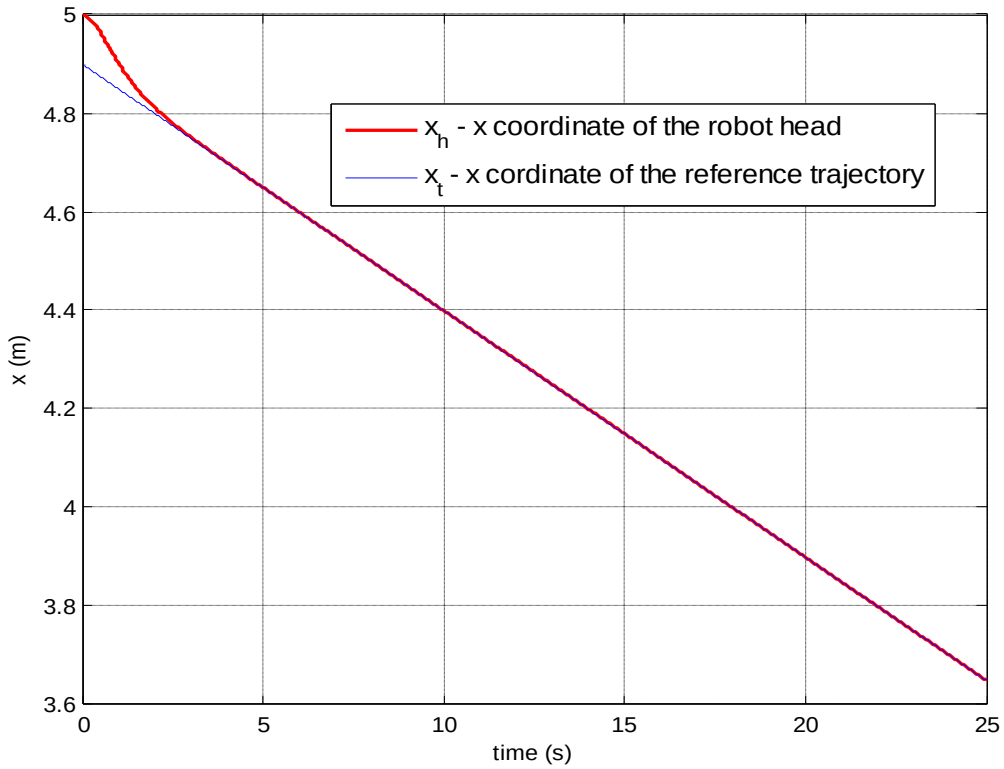


Figure 5.4: Plot of x coordinates of robot head trajectory and reference trajectory

In Figure 5.4, the plot of the x coordinate of the robot head trajectory x_h is superimposed on the plot of the x coordinate of the reference task trajectory x_t for all of the simulation time of 25 seconds. Also, similarly in Figure 5.5, the plot of y coordinates of the head y_h and the reference trajectory y_t are superimposed. In Figure 5.6, plot of tracking errors in x direction e_t^x and in y direction e_t^y are superimposed for about 8 seconds which is enough to capture the overall dynamics. When all of the four figures, Figure 5.3, Figure 5.4, Figure 5.5 and Figure 5.6 are observed, it *seems* that the robot accomplishes its given task to follow the reference line given by $x_t(t) = -0.05t + 4.9$ and $y_t(t) = 10.1$. However, simply looking at Figure 5.3 is enough to see that, by following the given straight at steady state all of the joint angles φ_i, $i = 1,\ldots,4$ converges to the same value of

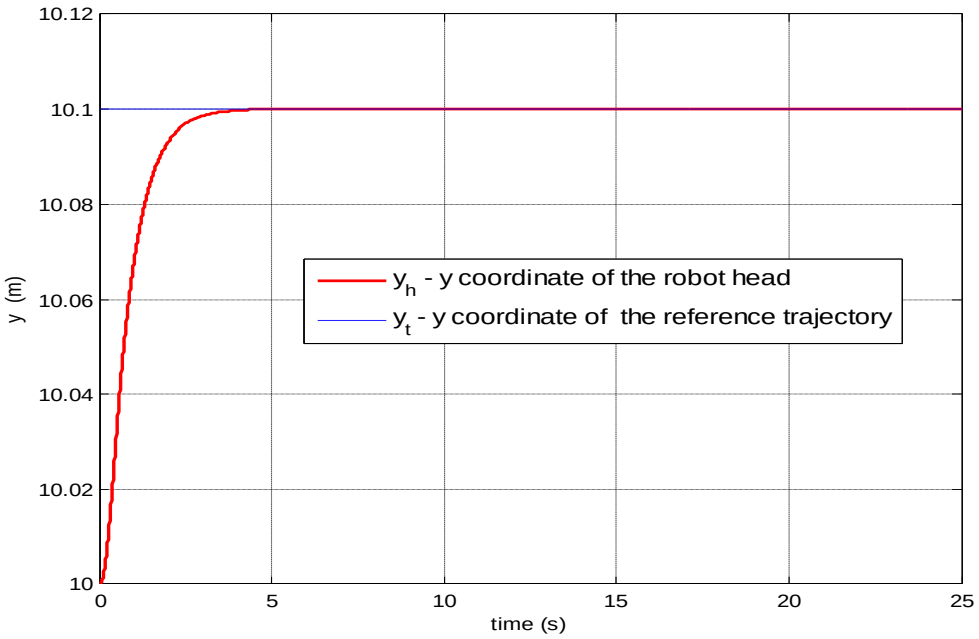


Figure 5.5: Plot of y coordinates of robot head trajectory and reference trajectory

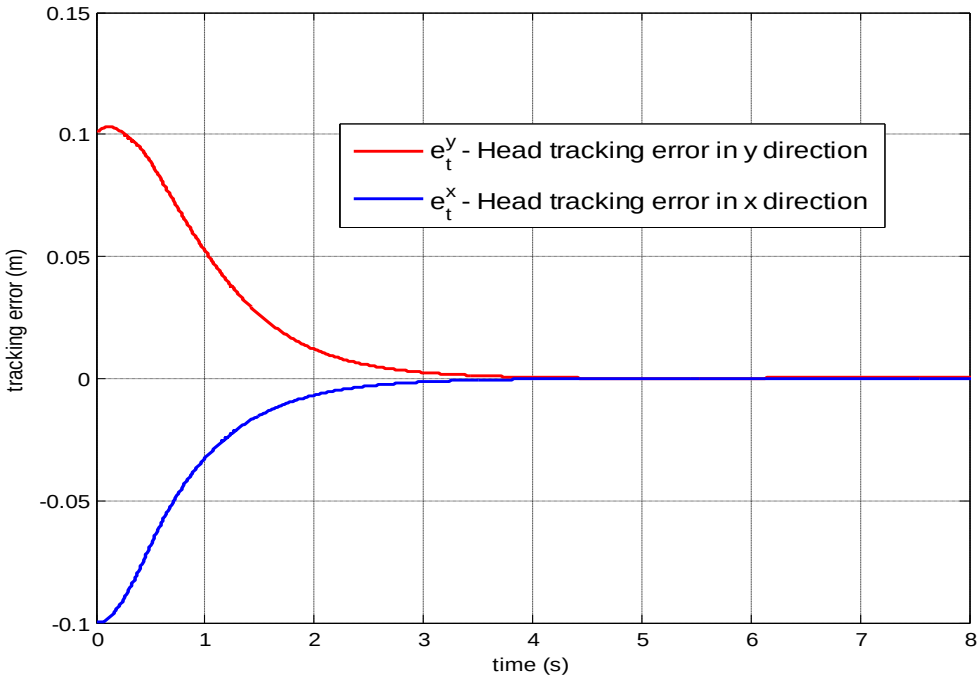


Figure 5.6: Plot of tracking errors in x direction e_t^x and in y direction e_t^y

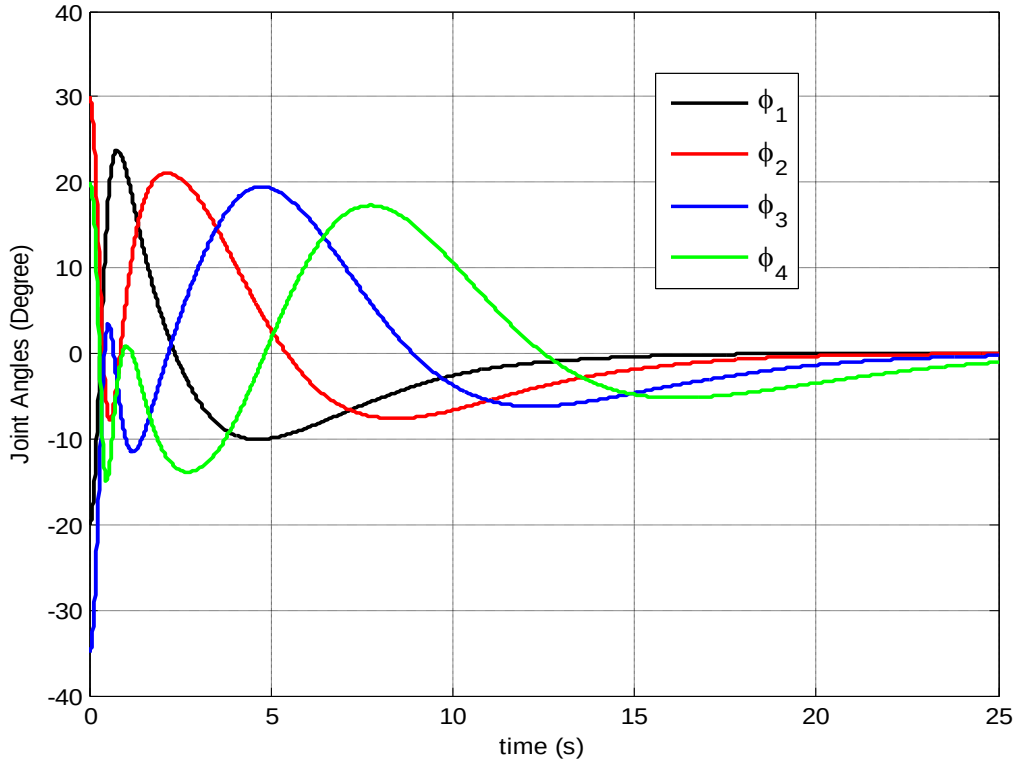


Figure 5.7: Plot of the joint angles φ_i, $i = 1,\ldots,4$ of the robot during a simulation time of 25 seconds

zero degrees. This fact is illustrated by Figure 5.7 where it is seen that φ_i, $i = 1,\ldots,4$ converges to zero with different rates. The convergence rate to the steady state value (desired value) is fastest for ϕ_1 and slowest for ϕ_4 since the robot locomotes forwards, the heads leads and the other modules follow it with a certain phase lag, as expected. Furthermore occurrence of the singularity condition is further justified by Figure 5.8 and Figure 5.9. In Figure 5.8, the lateral forces $f_1^N,\ldots, f_5^N$ applied by the ground to the modules through the wheels during the locomotion are plotted against time. In Figure 5.9, control torques $u_1,\ldots,u_4$ applied by the motors to the joints are plotted. Clearly, it is observed that both $u_1,\ldots,u_4$ and $f_1^N,\ldots, f_5^N$ diverge unboundedly after the singularity condition is approached.

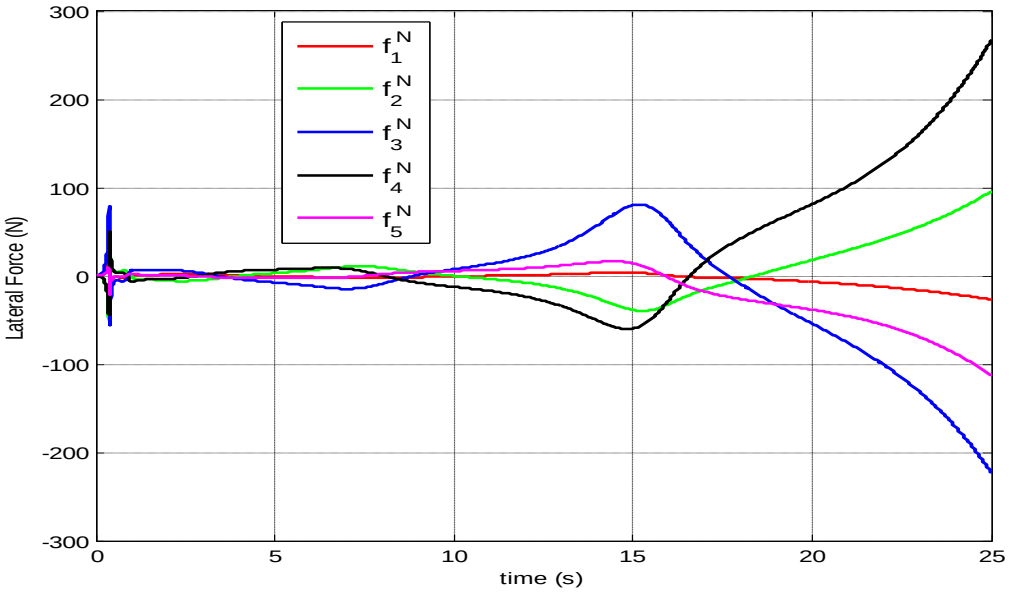


Figure 5.8: Lateral forces $f_1^N,\ldots, f_5^N$ applied by the ground to the modules through the wheels during the locomotion

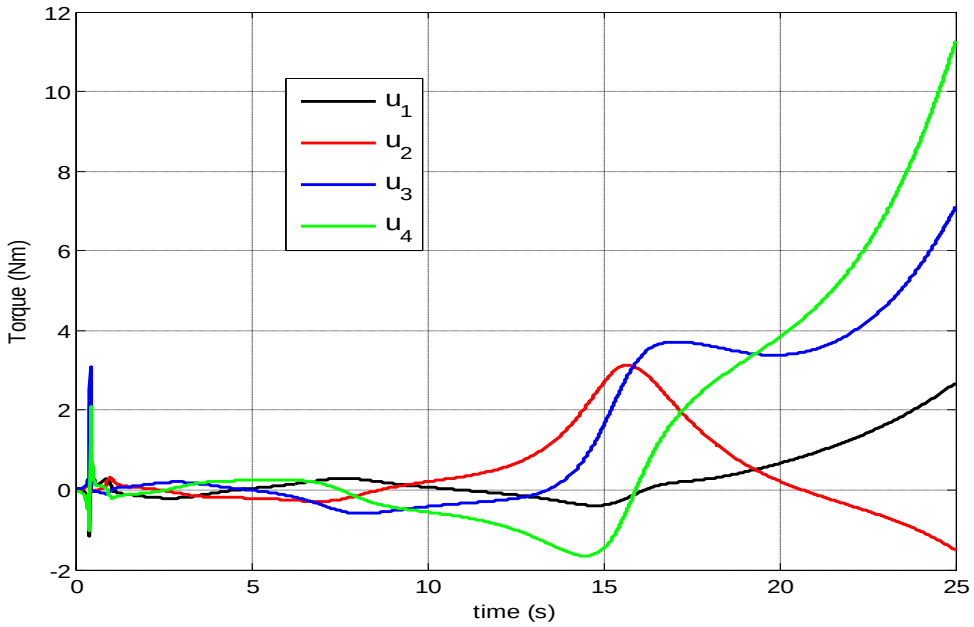


Figure 5.9: Torques $u_1,\ldots,u_4$ applied by the actuators to the robot joints during the locomotion

That is, when the singularity is neared, in order to continue the oscillations and locomotion the actuators work harder without any help due to the reasons stated before. While the actuators apply gradually bigger torques, the ground applies bigger lateral forces to the modules in order to satisfy the nonholonomic no slip condition. This loop becomes inevitable and consequently both $u_1,\ldots,u_4$ and $f_1^N,\ldots, f_5^N$ diverge. Actually the torques *applied* by the actuators and the forces *applied* by the ground refer to the hypothetical values that should be applied. However, as usual, there are limits both for the actuators and for the ground. These limits are set by the capacity of the motors and the friction coefficient of the ground in the lateral (normal) direction. Hence, singularity means that the locomotion will inevitably stop. Also in Figure 5.10, the total power *P(t)* applied by the all of actuators combined to the system during the locomotion is plotted. The total power applied is simply given by the equation

$$P(t) = \sum_{i=1}^{4} \dot{\phi}_i(t) u_i(t) \tag{5.37}$$

Observing equation (5.37) it can be seen that all of the actuators (motors) are assumed to be regenerative and this assumption is used throughout this study. As it can be seen from Figure 5.10, some variable amount of power is applied to the system until the robot body aligns with the reference line and until the initial offset error of the head position is vanished. Afterwards a constant amount of power is applied to overcome the tangential friction applied by the ground. To conclude, it is necessary to provide the robot with a reference trajectory with non-constant velocity to avoid singularity and yet this trajectory must be applicable and reasonable.

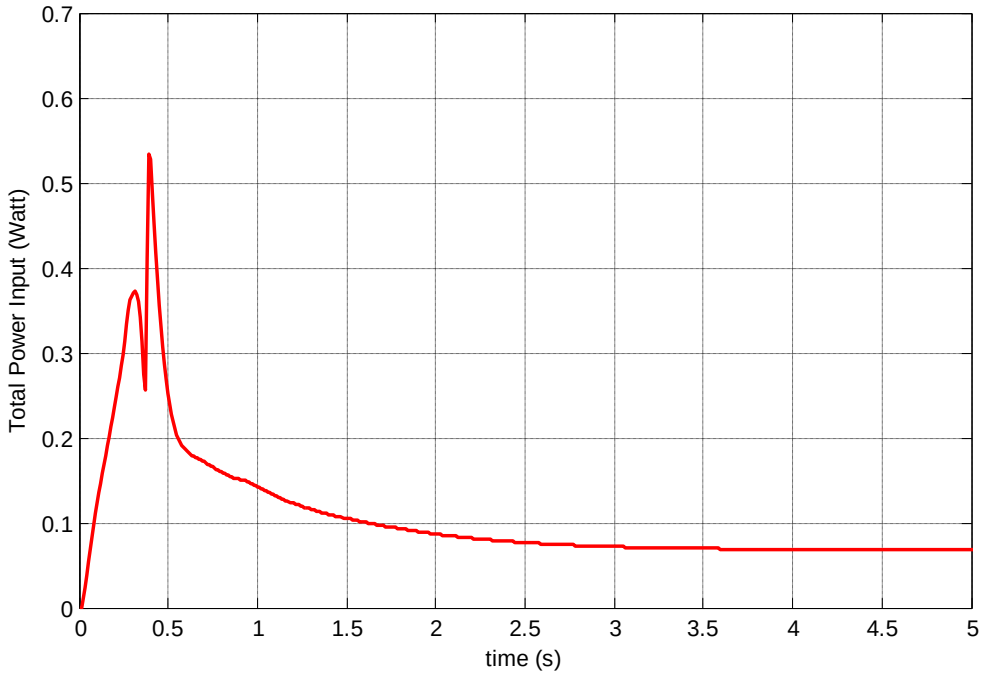


Figure 5.10: Total power input by the all of the actuators to the system during the locomotion

CHAPTER 6

MODIFIED SERPENOID CURVE

In this chapter, the *serpenoid curve* developed by Hirose will be briefly introduced. Then, parameters defining the overall geometry of the serpenoid curve will be described. Afterwards, a novel *modified serpenoid curve* will be introduced whose aim is to map the task reference trajectory to a serpenoid curve. By mapping the reference task trajectory to a serpenoid curve, the controller is supplied with a *tuned* reference so that tracking it will not lead to the singular configuration. Lastly, the robot behavior will be simulated when it tracks the proposed new reference track and it will be observed that it avoids trapping into the singularity.

6.1 Serpenoid Curve

The robotic researcher Shigeo Hirose has thoroughly studied the locomotion patterns of living snakes and acquired biological inspirations from his investigations. In his seminal work [17], Hirose introduced the so called serpenoid curve which he demonstrates real snakes in nature closely follow during their most common locomotion patterns of lateral undulation. In his studies, close agreement has been found between the empirical data obtained from the gaits of natural snakes and the serpenoid curve on constant friction surfaces. He concludes that the serpentine motion follows a kind of serpenoid curve where the curvature along the snake body varies sinusoidally. The curvature of the serpenoid curve is given in

[17] as

$$\kappa(s) = -\alpha \frac{2K_n\pi}{L} \sin(\frac{2K_n\pi}{L} s) \tag{6.1}$$

where, referring to Figure 6.1, s is the body length along the body curve of the robot, α is the initial winding angle (the value of the tangential angle ψ when s equals to zero), K_n is the number of S-shapes in the curve and L is total effective body length of the robot which is $10l$ in our case which means L = 1m. As explained in [17] in detail, this curvature is related to the muscular force a snake needs to form a certain body shape which justifies that the serpenoid curve is a natural selection for snake like movement simulation. This curve allows a smooth and continuous forward motion for the serpentine locomotion. By integrating equation (6.1) along the curve, the tangential angle at s can be found to be

$$\psi(s) = \alpha \cos(\frac{2K_n\pi}{L} s) \tag{6.2}$$

The equations of the serpenoid curve are then given as

$$\begin{aligned} x(s) &= \int_0^s \cos(\psi(\sigma))d\sigma \\ y(s) &= \int_0^s \sin(\psi(\sigma))d\sigma \end{aligned} \tag{6.3}$$

where *x(s)* and *y(s)* stand for the displacements in the *x* and *y* directions along the body curve at s, respectively. Geometrical representation of the above mentioned parameters on the serpenoid curve is given in Figure 6.1. Based on his investigations, Hirose proposed that a modular vehicle composed of simple and repetitive parts which follows his serpenoid curve could accomplish locomotion through the application of internal torques to its modules imitating the natural

snakes which contract their muscles to generate creeping movement. Therefore, utilizing the serpenoid curve for snake like robotic applications has been a

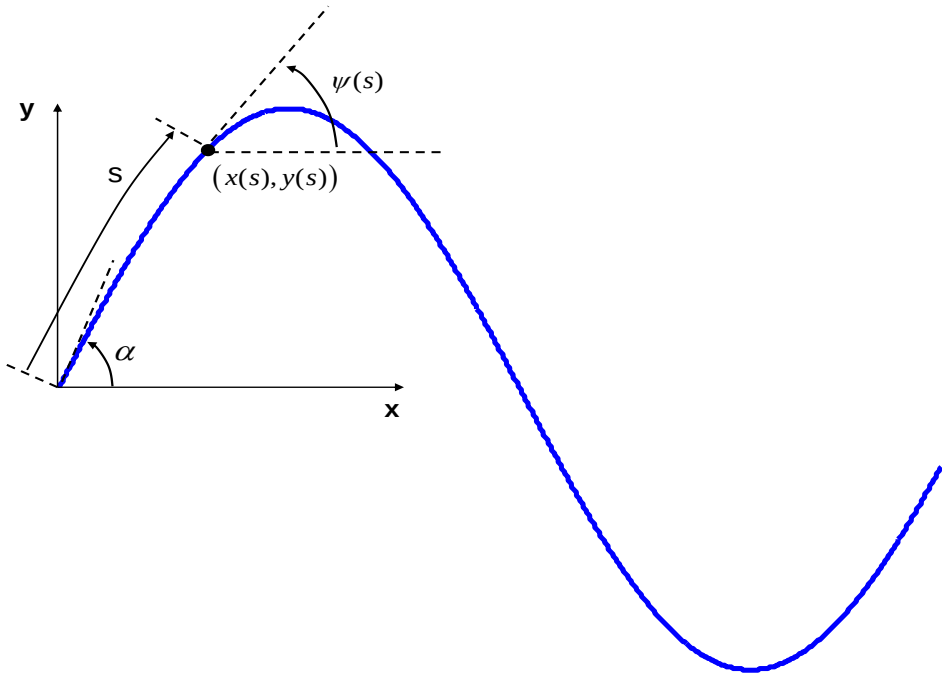


Figure 6.1: Serpenoid curve proposed by Hirose and its parameters

common practice in the literature. That is because it is actually a form of gait of lateral undulation for snakes that they have developed for millions of years through evolution. Experimental studies also indicate that applied torques at joints are smooth, continuous with minimal energy consumption rates observed both in biological snakes and robotic ones.

6.2 Parameters of Serpenoid Curve

The above mentioned parameters determine the main geometrical features of the

Hirose's serpenoid curve. When Figure 6.1 is observed, it is seen that the curve is actually oscillating symmetrically around its *line of symmetry* (x axis). Although it looks like a simple sinusoidal curve it is actually not. Obviously, if a robotic snake is made to follow the generic serpenoid curve given in Figure 6.1, at the *macro scale* it would simply follow the x axis. In order to also change the direction (at a macro scale) of the robot, the method of symmetrical line modulation has been proposed by Hirose. It simply modifies equation (6.1) by adding a constant *c* for turning motion as

$$\kappa(s) = -\alpha \frac{2K_n\pi}{L} \sin(\frac{2K_n\pi}{L} s) + c \tag{6.4}$$

which implies that equation (6.2) takes the form

$$\psi(s) = \alpha \cos(\frac{2K_n\pi}{L} s) + cs \tag{6.5}$$

By adding the constant *c* to the curvature, the robot is made to follow a circular curve at the macro scale. The effect of parameter α, K_n and *c* on the serpenoid curve is illustrated in Figure 6.2, Figure 6.3 and Figure 6.4. *L* is taken as 1m (total body length of the robot) throughout all the following simulations and figures. As can be seen from the figures, the parameter α determines the severity of undulation, the parameter K_n determines the number of S-shapes (number of periods) in unit length while *c* determines the radius of the circle at the macro scale. Apart from the method of line of symmetry modification first introduced in [17], other attempts have been made to modify the usual serpenoid curve so that the robotic snake can be given a definite direction to follow in a high level. Some of those attempts are the methods illustrated in [74]. In their work starting from equation (6.2), they obtain the relative angles ϕ between the modules (joint angle). The joint angles ϕ_i are obtained by discretization of equation (6.2) in the form of

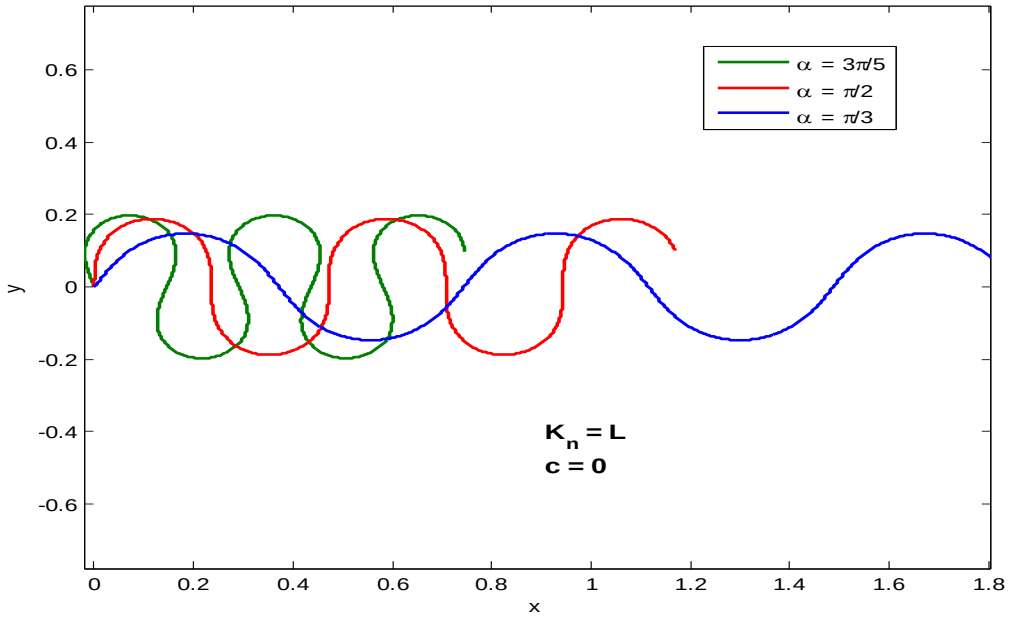


Figure 6.2: Effect of winding angle α on serpenoid curve

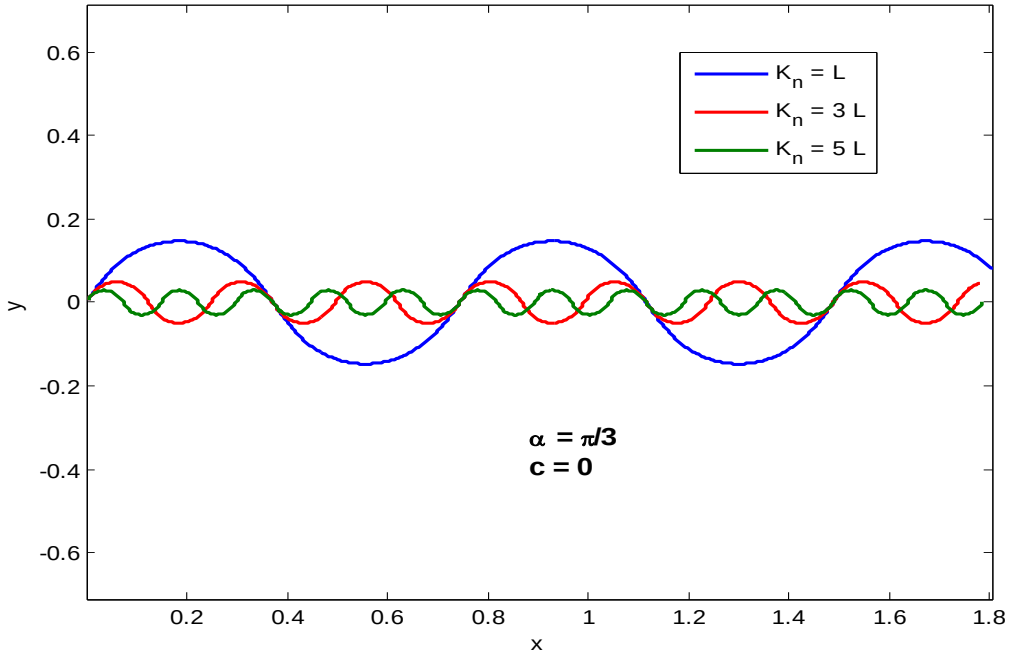


Figure 6.3: Effect of parameter K_n on serpenoid curve

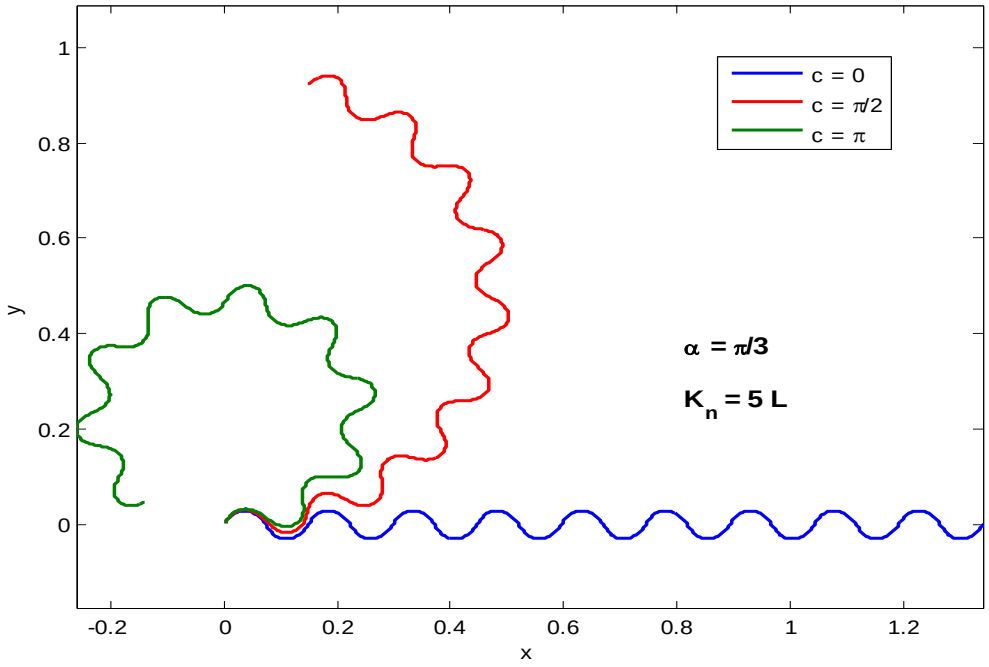


Figure 6.4: Effect of parameter *c* on serpenoid curve

$$\phi_i = A\sin(\omega t + (i-1)\beta) \tag{6.6}$$

Using their methods, such as amplitude modulation, phase modulation and side movement modulation on the joint angles ϕ_i, the serpenoid curve is modified in a desired manner. Finally this modified form of the curve forms the actual reference trajectory to track. There are very few studies in the literature concentrating on the turning motion of the robotic snake when it is made to follow the serpenoid curve. The usual approach is after defining all the joint angles ϕ_i and modifying them such that the desired macroscopic shape of the serpenoid curve is obtained, the control torques are applied such that the actual joint angles closely follows the reference ones.

Hence, by making the joint angles track the reference ones, the overall locomotion is made to follow the serpenoid curve. Also, by proper modification of ϕ as

explained above, the serpenoid curve is given a certain direction at a macro scale. However, this is quite an implicit way to make the robot to follow a desired trajectory. That is because, the relation between the actual reference path to follow and the parameters used to modify the joint angles ϕ_i inferred from equation (6.2) is always implicit and indirect. This means that, using these methods, exact tracking of reference trajectories is not possible. Thus a new method of modification of the serpenoid curve is proposed such that it will be able to follow any feasible task trajectory *exactly* at a macro scale in an explicit way.

6.3 Proposition of the Modified Serpenoid Curve

The motivation behind our modified serpenoid curve is to be able to generate a serpenoid curve in an explicit and quantitative manner such that it can represent any feasible task trajectory at the macro scale.

First it is observed that, for example, to generate a serpenoid curve that undulates around (represents) the line which makes an angle of γ degrees with the positive x axis it is necessary to modify equation (6.2) as follows

$$\psi(s) = \alpha \cos(\frac{2K_n\pi}{L} s) + \gamma \qquad (6.7)$$

Equation (6.7) basically means that an angle of γ is deliberately added to the tangential angle $\psi(s)$ of the serpenoid curve so that it performs its oscillations around the objective line (the one that makes an angle of γ degrees with the positive x axis). Substituting equation (6.7) into equation (6.3) the new form of the serpenoid curve on the Cartesian coordinates is obtained as follows

$$x(s) = \int_0^s \cos(\alpha \cos(\frac{2K_n\pi}{L}\sigma) + \gamma) d\sigma$$
$$y(s) = \int_0^s \sin(\alpha \cos(\frac{2K_n\pi}{L}\sigma) + \gamma) d\sigma \qquad (6.8)$$

Equation (6.8) can be solved using numerical integration methods only [17] when the initial conditions x_0 and y_0 are given. To illustrate a sample case, equation (6.8) has been integrated numerically by the built-in ODE solver of Matlab (*ode45*) for a total curve length of 2.4 meters (L = 2.4 m) with the initial conditions $x_0 = x(0) = y_0 = y(0) = 0$. The angle γ, which will be called *compensation angle* for the modified serpenoid curve, is selected to be 60° so that it "follows" the line given by the equation $y = \sqrt{3}x$. The other parameters are also assigned certain values. The result of the numerical integration is plotted in Figure 6.5 with line given by the equation $y = \sqrt{3}x$ as the objective line.

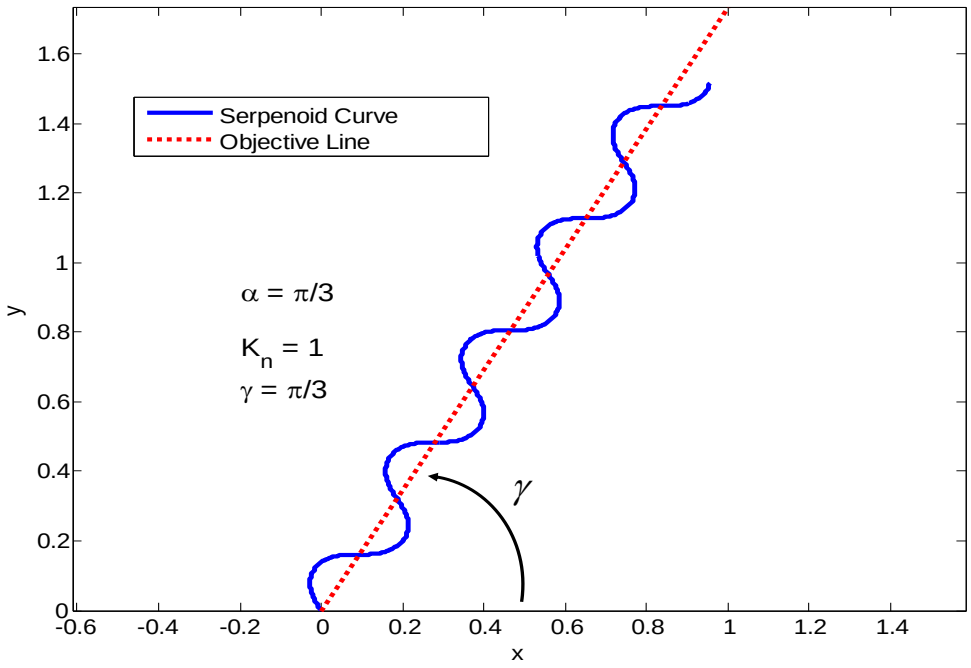


Figure 6.5: Modified serpenoid curve is made to follow an objective line

As can be seen in Figure 6.5, it oscillates around its objective line at the macro scale, starting from the origin. Hence, by modifying the initial conditions x_0 and y_0 with the compensation angle γ any given line on the Cartesian plane can be followed. While the parameter γ determines the overall propagation direction of the the curve at a macro scale, the parameters K_n and α are more about the internal geometrical shape of the curve. In another point of view, by modifying γ where the locomotion will proceed in the Cartesian plane can be determined; while by modifying K_n and α the *pattern* in which the locomotion will proceed can be determined. However, so far the serpenoid curve can only be made to follow straight lines. Obviously, it is required to make it follow any feasable curve on the plane so that a *tuned* reference trajectory for the head of the robot can be generated. By doing that, it will be possible to *fit* a serpenoid curve around the given task trajectory which will serve as *the objective curve* instead of the objective line.

Thus, once the robot head is made to follow this fitted serpenoid curve instead of the original given task trajectory, the singularity conditions will be avoided. That is because, due to the oscillatory patterns of the serpenoid curve the robot head will never have constant velocity reference trajectory. Hence, the joint angles φ will not be equal to each other in modulus π. Inspiring by Figure 6.5, and observing that in order to follow a curve with constant slope (which is a line) a constant compenstation angle γ is added, we propose that in order to follow a generic curve, the companstation angle γ should vary with the slope of the curve. Specifically, it is required that at point *s* on the *to-be-fitted-serpenoid-curve*, the compensation angle should be equal to the tangent angle at point s_t which is the intersection of the normal drawn to the objective curve from the point *s* with the objective curve itself. Geometrical underpinnings of the modified serpenoid curve is illustrated in Figure 6.6. Then it is noted that the compensation angle $\gamma(s_t)$ now depends on the objective curve itself and hence the location s on the serpenoid

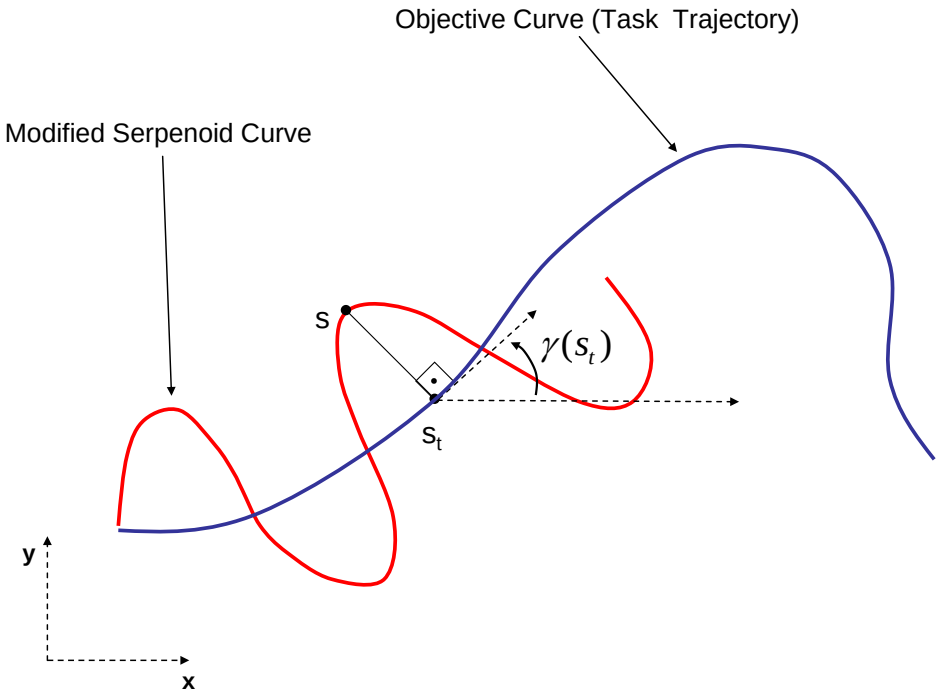


Figure 6.6: Construction of the Modified Serpenoid Curve

curve. Here s denotes the path length along the modified serpenoid curve and s_t is simply a parameter to define the objective curve parametrically (not necessarily the path length along the the objective curve). Let $x(s)$ and $y(s)$ denote the x and y coordinates of the modified serpenoid curve at s. Also let $x_t(s_t)$ and $y_t(s_t)$ denote the x and y coordinates of the objective curve (task trajectory) at s_t. Then, one may obtain

$$\left(\frac{y(s)-y_t(s_t)}{x(s)-x_t(s_t)}\right)\frac{\dot{y}_t(s_t)}{\dot{x}_t(s_t)}+1=0 \qquad (6.9)$$

which simply states that, multiplication of the slope of the line connecting the point at *s* and the point at s_t with the slope of the objective curve at s_t is minus one. In equation (6.9) $\dot{y}_t(s_t) = \frac{dy_t(s_t)}{ds_t}$ and $\dot{x}_t(s_t) = \frac{dx_t(s_t)}{ds_t}$.

Also from Figure 6.6, the compenstation angle can be defined as follows

$$\gamma(s_t) = \text{atan2}\left(\dot{y}_t(s_t), \dot{x}_t(s_t)\right) \tag{6.10}$$

Combining equation (6.8) and (6.10) the following two differential equations are obtained which define the modified serpenoid curve as

$$\frac{dx(s)}{ds} = \cos\left[\alpha\cos(\frac{2K_n\pi}{L}s) + \text{atan2}\left(\dot{y}_t(s_t), \dot{x}_t(s_t)\right)\right] \tag{6.11}$$

$$\frac{dy(s)}{ds} = \sin\left[\alpha\cos(\frac{2K_n\pi}{L}s) + \text{atan2}\left(\dot{y}_t(s_t), \dot{x}_t(s_t)\right)\right] \tag{6.12}$$

Equations (6.11) and (6.12) are simply the result of substituting equation (6.10) into equation (6.8) and then differentiating with respect to *s*. The core form of the third governing differential equation is equation (6.9). However, care must taken when dealing with this equation. This is because; there are two conditions when equation (6.9) diverges. These conditions are

1) $x(s) - x_t(s_t) \to 0$

2) $\frac{\dot{y}_t(s_t)}{\dot{x}_t(s_t)} \to \infty$

It is vital to avoid these singularity conditions as they would create problems during the numerical solution of the differential equations of the modified serpenoid curve. However, the above mentioned conditions are mutually exclusive, i.e. when the first one occurs the second does not and vice versa. This mutual exclusivity results from the manner in which the modified serpenoid curve is defined. The first condition corresponds to the case when the mentioned normal in Figure 6.6 is perpendicular to the global x axis and that is when the slope of the objective curve vanishes, i.e. $\frac{\dot{y}_t(s_t)}{\dot{x}_t(s_t)} = 0$. The second condition corresponds to the case when the mentioned normal in Figure 6.6 is parallel to the global x axis and when $\frac{\dot{y}_t(s_t)}{\dot{x}_t(s_t)} \to \infty$. Equation (6.9) is simply multiplied by $\left(x(s) - x_t(s_t)\right)$ so that the singularity case can be cancelled which occurs when $x(s) - x_t(s_t) = 0$ and finally

$$\left(y(s) - y_t(s_t)\right)\frac{\dot{y}_t(s_t)}{\dot{x}_t(s_t)} + x(s) - x_t(s_t) = 0 \qquad (6.13)$$

Thus, when $x(s) - x_t(s_t) = 0$, equation (6.9) is actually multiplied by zero. For that reason, the validity of equation (6.13) must be checked when $x(s) - x_t(s_t) = 0$. Observing again Figure 6.6, it is noted that when $x(s) - x_t(s_t) \to 0$ the slope of the objective curve also vanishes, i.e. $\frac{\dot{y}_t(s_t)}{\dot{x}_t(s_t)} \to 0$ as $x(s) - x_t(s_t) \to 0$. Hence, it can be concluded that equation (6.13) is valid and equivalent to equation (6.9) even when $x(s) - x_t(s_t) = 0$. In order to cancel the second singularity condition which will occur during the numerical solution of the differential equations, equation (6.13) is now multiplied by $\frac{\dot{x}_t(s_t)}{\dot{y}_t(s_t)}$ and gives

$$y(s)-y_t(s_t)+\left(x(s)-x_t(s_t)\right)\frac{\dot{x}_t(s_t)}{\dot{y}_t(s_t)}=0 \tag{6.14}$$

Following the same logic and noting that $\frac{\dot{y}_t(s_t)}{\dot{x}_t(s_t)}\to\infty$ implies $\frac{\dot{x}_t(s_t)}{\dot{y}_t(s_t)}\to 0$, equation (6.13) is actually multiplied by zero when the second singularity condition occurs.

However, since $\frac{\dot{x}_t(s_t)}{\dot{y}_t(s_t)}\to 0$ implies $y(s)-y_t(s_t)\to 0$, equation (6.14) is valid and equivalent to equation (6.13) even when $\frac{\dot{x}_t(s_t)}{\dot{y}_t(s_t)}\to 0$. Hence, it is concluded that during the solution of the differential equations of the modified serpenoid curve represented by equations (6.9), (6.11) and (6.12); instead of equation (6.9), equation (6.13) must be used around the first singularity condition and equation (6.14) must be used around the second singularity condition. At any point other than these two mentioned singularity points, either equation (6.13) or equation (6.14) can be used safely. To conclude, the complete set of differential equations of the proposed modified serpenoid curve are obtained as follows

$$\begin{aligned}
\frac{dx(s)}{ds}&=\cos\left[\alpha\cos(\frac{2K_n\pi}{L}s)+\operatorname{atan2}\left(\dot{y}_t(s_t),\dot{x}_t(s_t)\right)\right]\\
\frac{dy(s)}{ds}&=\sin\left[\alpha\cos(\frac{2K_n\pi}{L}s)+\operatorname{atan2}\left(\dot{y}_t(s_t),\dot{x}_t(s_t)\right)\right]
\end{aligned} \tag{6.15}$$

$$\begin{cases}
y(s)-y_t(s_t)+\left(x(s)-x_t(s_t)\right)\dfrac{\dot{x}_t(s_t)}{\dot{y}_t(s_t)}=0 & \text{if} \quad \dfrac{\dot{y}_t(s_t)}{\dot{x}_t(s_t)}\to\infty\\
\left(y(s)-y_t(s_t)\right)\dfrac{\dot{y}_t(s_t)}{\dot{x}_t(s_t)}+x(s)-x_t(s_t)=0 & \text{otherwise}
\end{cases}$$

When the equation set (6.15) is examined, it is seen that it is implicit i.e. the highest order derivatives cannot be collected at one side of each individual equation. For this special case, a special built-in differential equation solver of Matlab, called *ode15i* is utilized. Implementation of equation (6.15) for *ode15i* and the complete Matlab code for generation of the modified serpenoid curve that fits around any given parametric curve on a plane is given in Appendix B. To illustrate the generation of the modified serpenoid curve, two sample reference trajectory is given parametrically to be $r_t(s_t) = x_t\hat{i} + y_t\hat{j} = (s_t + 5)\hat{i} + (s_t{}^2 + 10)\hat{j}$ for task trajectories are given in Figure 6.7 and Figure 6.8. In Figure 6.7, the reference $0 \le s_t < 5.5$ which corresponds to the parabolic curve given in Cartesian coordinates as $y = x^2 - 10x + 35$ for $0 \le x < 10.5$. The solution starts from the initial condition $(x_0, y_0) = (5,10)$ which, for example, may represent the initial head position of the robot. It is preferred to represent the task trajectories parametrically

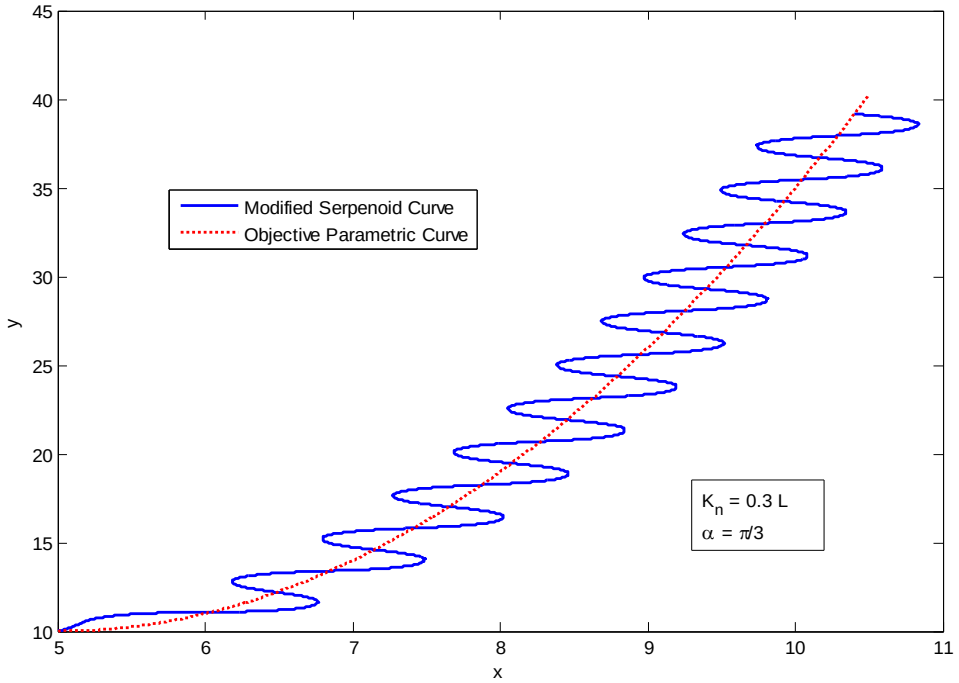


Figure 6.7: Fitting of the Modified serpenoid curve to the objective curve $r_t(t) = (t + 5)\hat{i} + (t^2 + 10)\hat{j}$ for $0 \le t < 5.5$ from the initial point $(x_0, y_0) = (5,10)$

instead of the Cartesian coordinates since parametric representation is more flexible, compact and complete for numerical solution of equation (6.15). Also in Figure 6.7 and Figure 6.8, parameter *t* is substituted for s_t for simplicity. In Figure 6.8 the serpenoid curve is fitted to the parametric objective curve

$r_t(s_t) = (-4\sin(s_t \frac{\pi}{2}) + 5)\hat{i} + (8\cos(s_t \frac{\pi}{4}) + 2)\hat{j}$ for $0 \le s_t < 8$ which forms a sort of figure "eight" with initial point $(x_0, y_0) = (5,10)$, again.

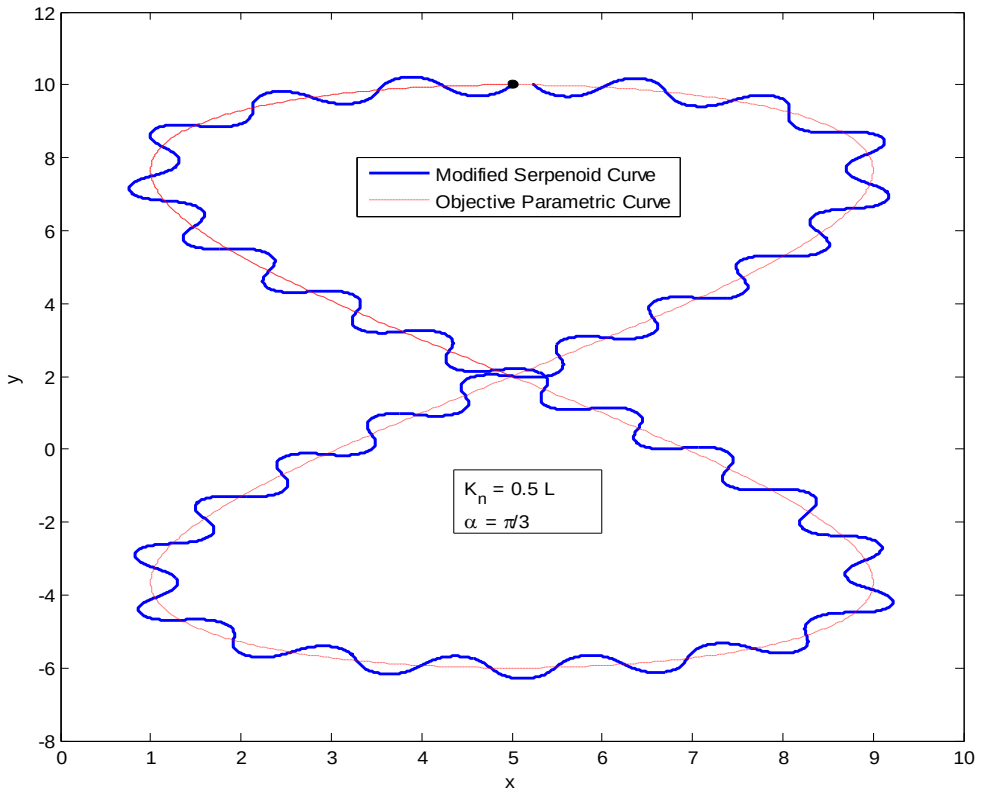


Figure 6.8 : Fitting of the Modified serpenoid curve to the objective curve $r_t(t) = (-4\sin(t \frac{\pi}{2}) + 5)\hat{i} + (8\sin(t \frac{\pi}{4}) + 2)\hat{j}$ starting from the initial point $(x_0, y_0) = (5,10)$

If this curve were to be represented in Cartesian coordinates four different equations and associated end points would be needed. When Figure 6.7 and Figure 6.8 are examined it is noted that the modified serpenoid curve smoothly undulates around the reference task trajectory and exactly follows it at a macro scale. Thus, instead of the original task trajectory the controller is supplied with the modified serpenoid curve. This leads to the modification of the overall control architecture given in Figure 5.2. Accordingly, the new overall control architecture of the modular robot is given in Figure 6.9. It is constructed by supplying the modified serpenoid curve fitted to the task trajectory as reference. Thus, from now on, instead of $r_t = x_t\hat{i} + y_t\hat{j}$, $r_r = x_r\hat{i} + y_r\hat{j}$ will be used as the reference to the control system and it denotes the generated modified serpenoid curve. Also, in Figure 6.9 there is a block named Optimizer.

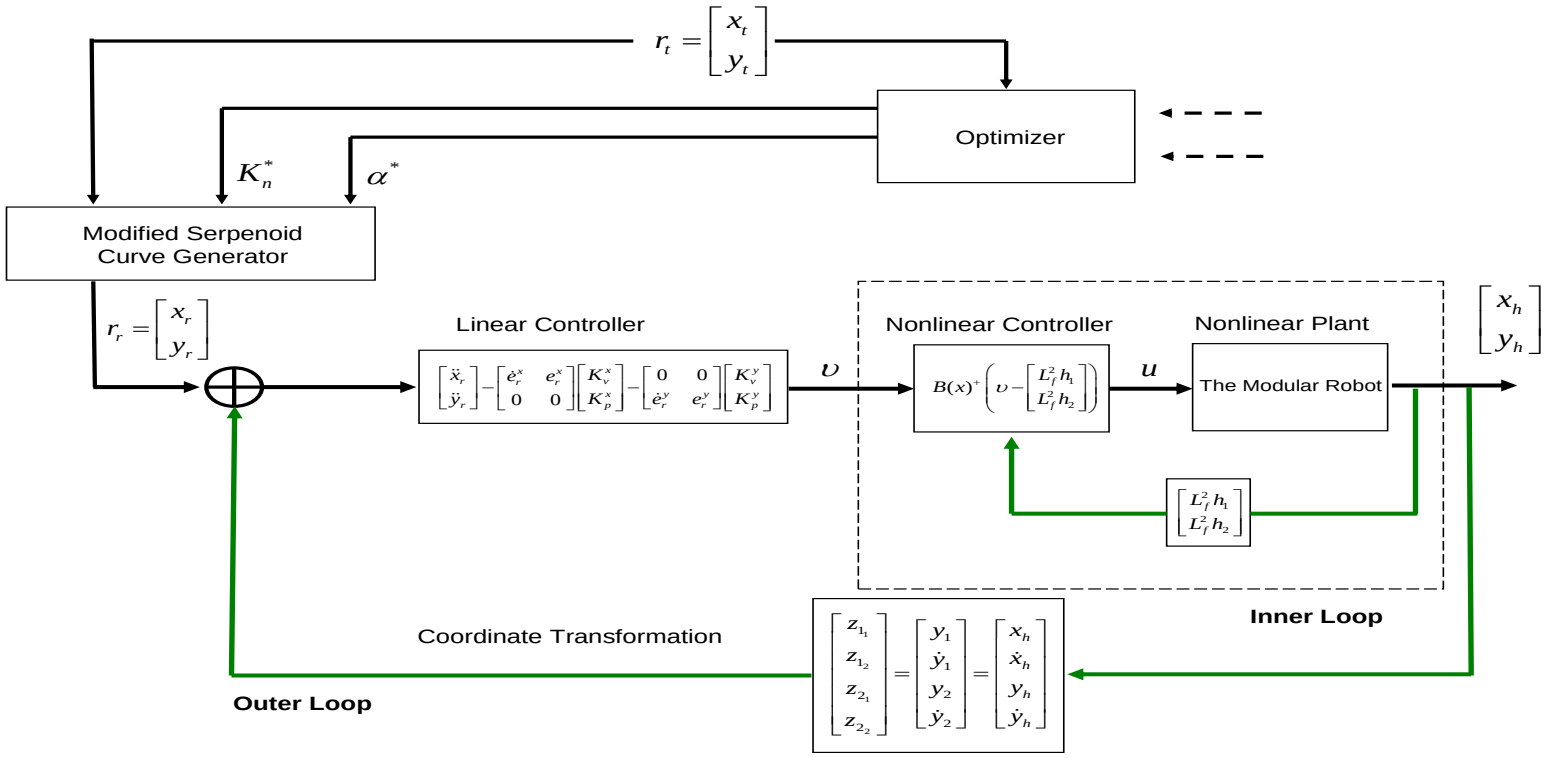


Figure 6.9: Overall control architecture with modified serpenoid curve as reference

This block provides the modified serpenoid curve generator with the optimal values of K_n and α which are denoted by K_n^* and α^*, respectively. The way in which K_n^* and α^* are determined will be explained in Chapter 7. Dashed arrows in Figure 6.9 show that there are inputs to the optimizer other than the task trajectory to determine K_n^* and α^*, which will be clarified in Chapter 7, again.

6.4 Tracking of the Modified Serpenoid Curve

As have been observed in section 5.5, when the robot head is made to follow a reference trajectory with a constant velocity, although it successfully tracks the reference it eventually converges to the singularity condition. After that it is no more able to continue locomotion. An Interesting observation about the simulation that has been performed in section 5.5 is that: when the tracking errors e_t^x and e_t^y are made to converge identically to zero, the robot converges also to the singularity conditions. Hence, the trick here is to make the tracking errors swirl around zero such that the robot tracks the reference at a macro scale. That was exactly the motivation behind our development of the modified serpenoid curve. Thus, instead of subscript "*t*" for the errors and the reference track, the subscript "*r*" will be used for them which will denote the modified serpenoid curve instead of the original task trajectory. Thus, that is the case given in Figure 6.9 which is constructed by using the modified serpenoid curve as the reference input. Here, the improvements made by utilizing the modified serpenoid curve $r_r(t)$ instead of the original task trajectory $r_t(t)$ will be demonstrated. For that, the same structural parameters of the robot as in the simulation made in section 5.5 will be used. Also, instead of giving an initial position offset for the head position, the reference trajectory will start exactly at (5,10). Thus, the task trajectory $r_t(t) = (-0.05t + 5)\hat{i} + 10\,\hat{j}$ for $0 \le t < 100$ will be used with $r_r(t) = x_r(t)\hat{i} + y_r(t)\,\hat{j}$

which is the numerical solution of the equation set given by (6.15). This is because, from now on, how the robot performs on a given trajectory will be focused on (instead of how it will correct the initial errors). One important issue to discuss here is the mapping between the numerical solution of the set of differential equations given by (6.15) and $r_r(t) = x_r(t)\hat{i} + y_r(t)\hat{j}$. In equation (6.15) , as have been stated before, the parameter *s* in $x(s)$ and in $y(s)$ denotes the path length along the serpenoid curve, while the parameter *t* in $r_r(t)$ denotes the actual simulation time. Hence, if it is simply defined

$$s = vt \tag{6.16}$$

where *v* is the speed of the head of the robot along the serpenoid curve, then, it follows

$$\begin{aligned} x_r(t) &= x(vt) \\ y_r(t) &= y(vt) \end{aligned} \tag{6.17}$$

It means that at time *t,* $x(vt)$ and $y(vt)$ are evaluated, since $x(s)$ and $y(s)$ are the known solutions of equation (6.15), and assigned to $x_r(t)$ and $y_r(t)$, respectively. Here, since analytical solution of equation (6.15) is not possible, the evaluation of $x(vt)$ and $y(vt)$ is to be performed by interpolation.

Let us set $v = 0.05$ m/s . Firstly, $x(s)$ and $y(s)$ are obtained by solving equation (6.15) for the selected serpenoid curve parameters $\alpha = \frac{\pi}{3}$, $K_n = 2L$. The solution is given in Figure 6.10. $\omega_x = \omega_y = 4$ rad/s is set for the linear controller, anticipating that an oscillatory reference will be tracked now instead of a linear one. The simulation is run for 35 seconds and the results are summarized in figures from Figure 6.11 to Figure 6.17. The constant -0.05 in the task trajectory $r_t(t) = (-0.05t + 5)\hat{i} + 10\hat{j}$ has absolutely no significance as it could be any negative number.

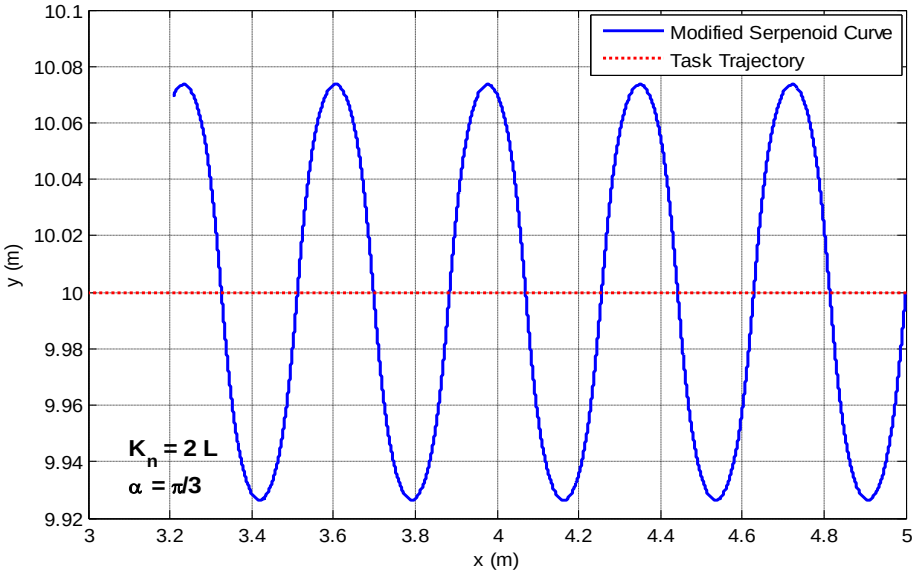


Figure 6.10: Superposition of the task trajectory $r_t(t) = (-0.05t + 5)\hat{i} + 10\,\hat{j}$ to its fitted serpenoid curve $r_r(t) = x_r(t)\hat{i} + y_r(t)\,\hat{j}$

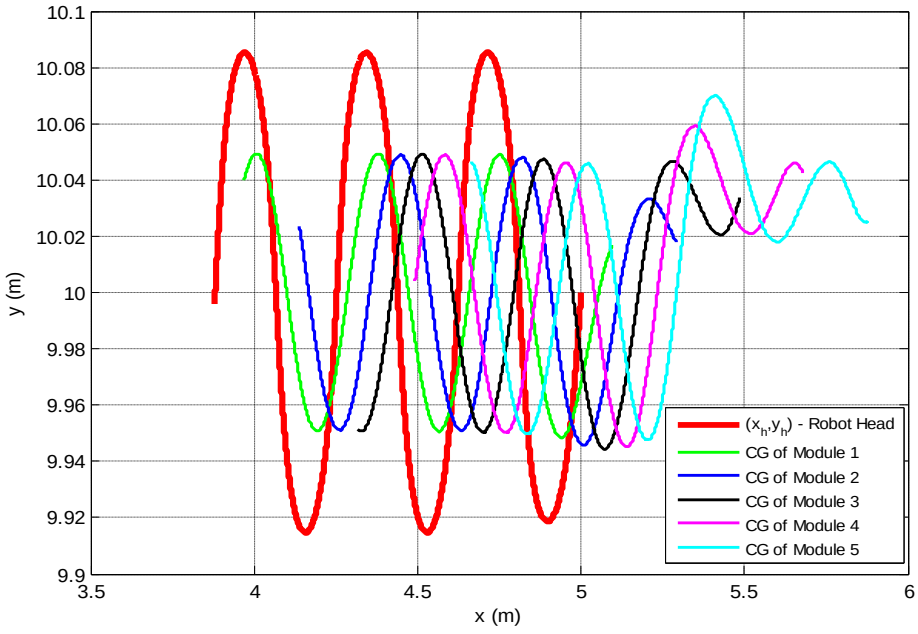


Figure 6.11: Trajectory of the robot head and modules following the reference $r_r(t) = x_r(t)\hat{i} + y_r(t)\,\hat{j}$

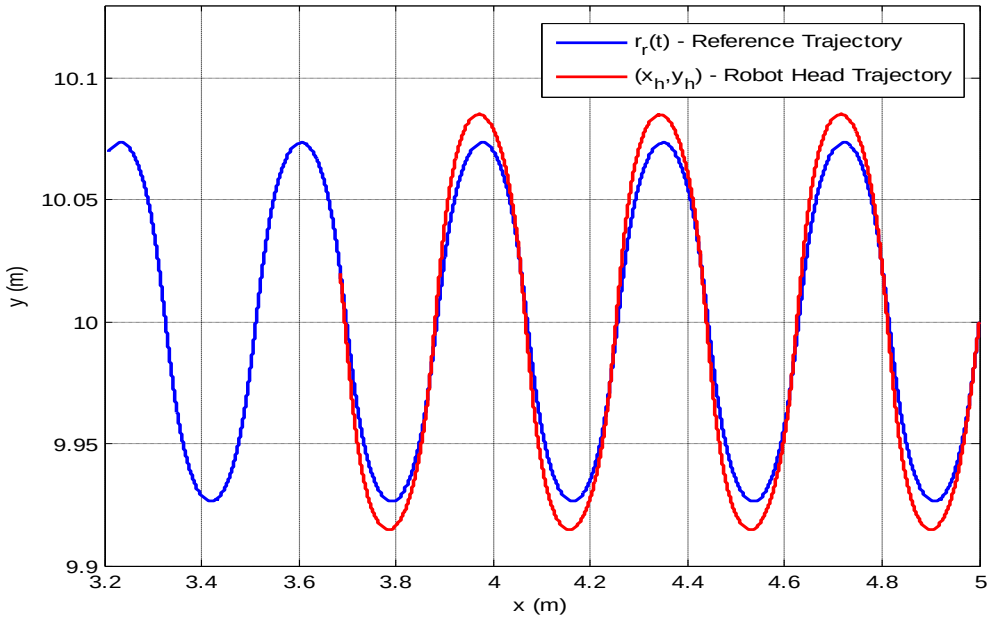


Figure 6.12: Trajectory of the robot head superimposed on the reference $r_r(t) = x_r(t)\hat{i} + y_r(t)\hat{j}$

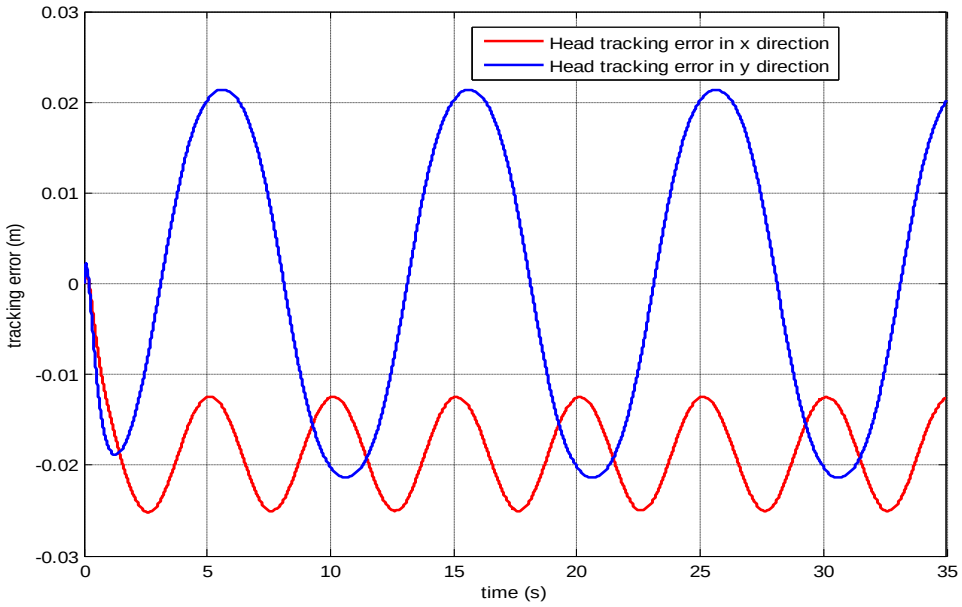


Figure 6.13: Robot head tracking errors e_r^x and e_r^y in x and y directions, respectively.

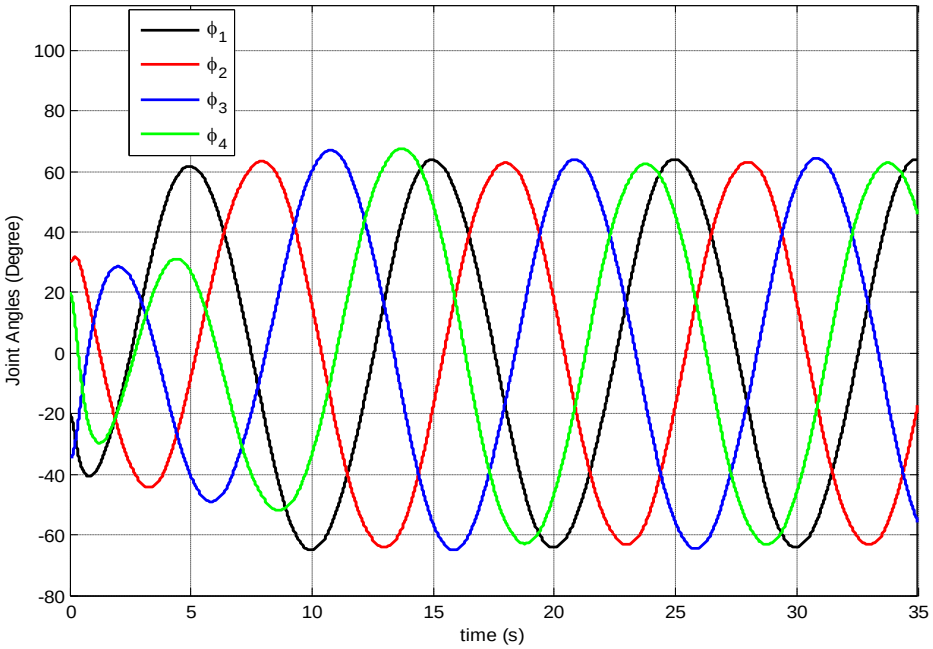


Figure 6.14: Joint angles φ_i, $i = 1,\ldots,4$ of the robot during the locomotion

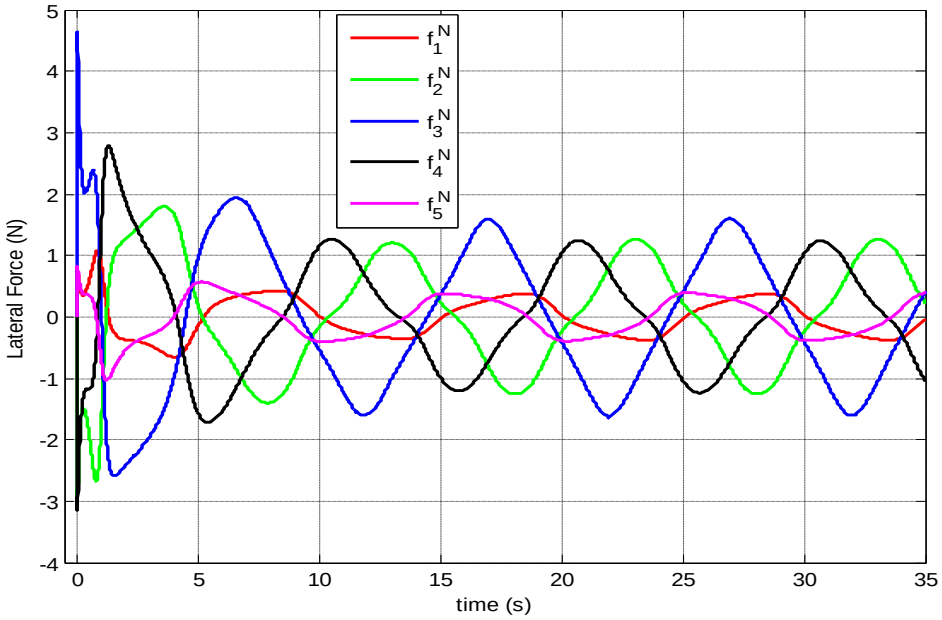


Figure 6.15: Lateral forces $f_1^N,\ldots, f_5^N$ applied by the ground to the modules through the wheels during the locomotion

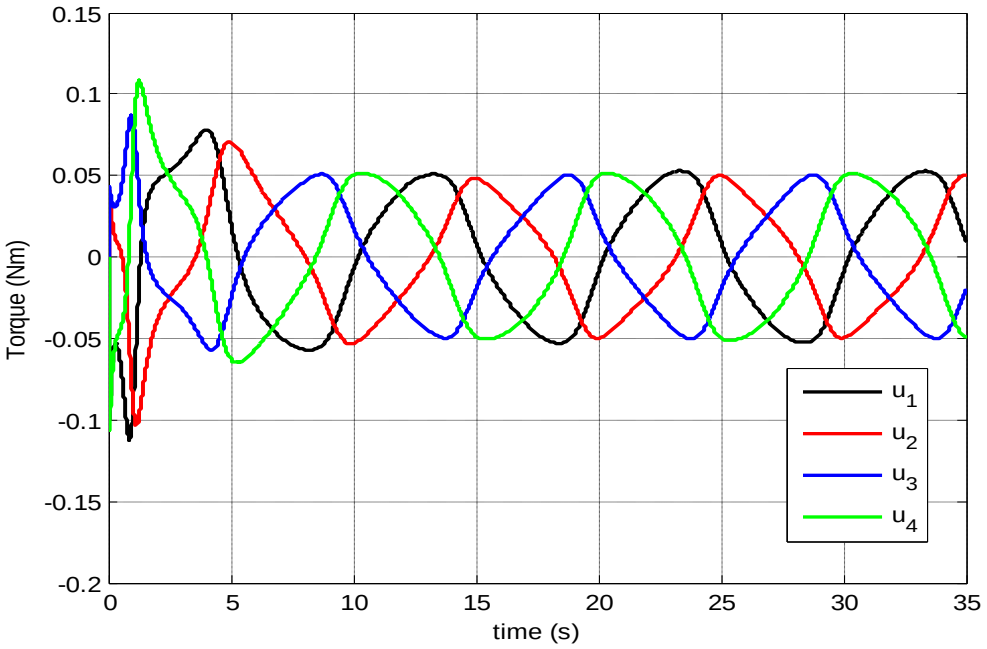


Figure 6.16: Torques $u_1,\ldots,u_4$ applied by the actuators to the robot joints during the locomotion

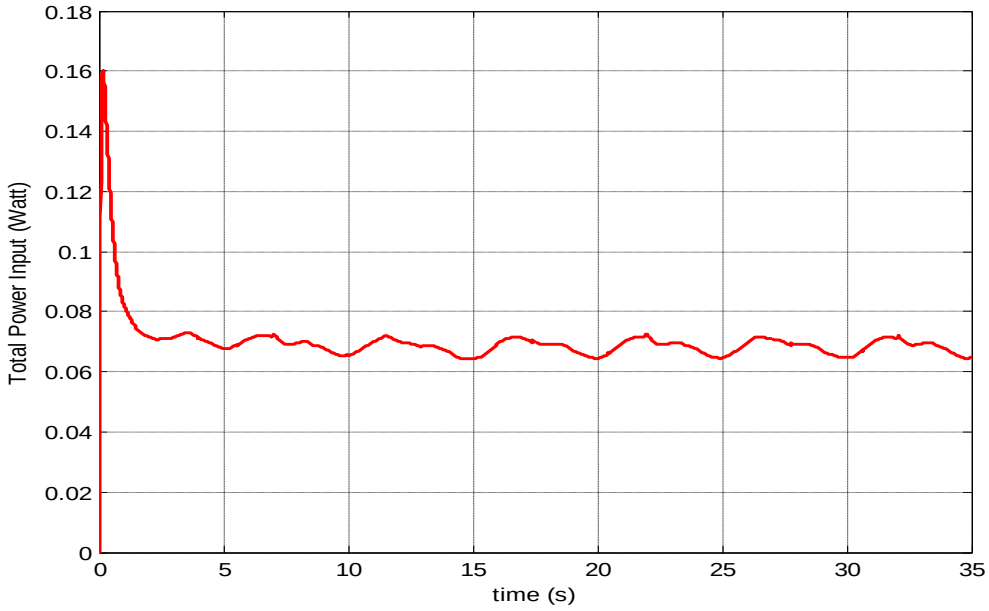


Figure 6.17: Total power input by the all of the actuators to the system during the locomotion

That is because it no more sets the velocity, but only serves for the purpose of defining the geometry of the path to be followed at the macro scale. Here the parameter *v* defines how fast the robot locomotes, instead. In Figure 6.11, it is observed how the robot performs the sinusoidal locomotion where the head leads and other modules follow. In Figure 6.12 and Figure 6.13 it can be seen how the robot head follows the generated serpenoid curve. Especially in Figure 6.13, it is seen that the error in the y direction of the head position oscillates around zero while the error in the x direction oscillates around -0.018 m, approximately. This fundamental difference arises from the very nature of the requirement imposed on the robot: to trace the serpenoid curve which is fitted around a task trajectory propagating in the negative x direction, with a certain velocity *v*. Oscillating around a negative value means that the robot always lags in the x direction along which the locomotion propagates. Hence, having a lag in the direction of locomotion is due to the critically damped linear controller that has been designed. Consequently, one can simply increase ω_x in order to decrease the lag in the x direction when the robot follows a line. Finally, examining Figure 6.14, Figure 6.15 and Figure 6.16, it is observed that what has been aimed has been accomplished i.e., singularity has been successfully avoided with a compromise of oscillating about 0.08 m around the actual trajectory. Indeed, it is not even a real compromise considering the fact that the robot is almost imitating how the real biological snakes locomote and 0.08 m is not even half length of a single module. In Figure 6.14 it is seen that the joint angles of the robot, in the steady state, oscillates with a common amplitude and a common frequency only differing with a constant phase difference consecutively from the head to the tail (from the first module to the fifth). This shows us that the snake robot moves in the form of a travelling wave just like the real snakes. Since the joint angles do not converge to a common value, singularity does not occur. In Figure 6.15 and Figure 6.16, the lateral forces $f_1^N,\ldots,f_5^N$ and actuator torques $u_1,\ldots,u_4$ not only stay bounded (as opposed to the ones in Figure 5.8 and Figure 5.9), but their magnitudes are also

significantly reduced. In Figure 6.17, total power input to the system by the actuators is given. Even during the steady state, total power input is oscillatory as it is now the multiplication of two oscillatory quantities (joint velocity and torque input).

What has been simulated in this case study was actually the locomotion of the robot following a straight line. It could have been done by using the classical serpenoid curve equation (6.3), developed by Hirose, instead of solving equation (6.15). Now, the locomotion of the robot on a complicated path will be simulated which can be realized only using our proposed modified serpenoid curve (if serpenoid curve method is to be used). As a more complicated path, the eight shaped path given in Figure 6.8 will be taken, with different dimensions appropriate for the simulation purposes. The simulation will be started with the same initial conditions and parameters. The only difference is that the robot will track the serpenoid curve fitted to the aforementioned eight shaped parametric curve. As have been mentioned earlier the modified serpenoid curve can be followed as closely as desired simply by increasing the position and velocity gains, or the natural frequencies ω_x and ω_y. Again, $\omega_x = \omega_y = 8$ rad/s are set for this simulation. Also v = 0.20 m/s is set, which means that the robot will locomote 4 times faster than before. In order not to deal with how the overall system will respond to the initial position offset error, the initial position error is set to zero. Observing Figure 6.15, it is seen that starting from the rest, to increase the speed of the head to the given v, the actuators work harder which give rise to abrupt jumps in the lateral forces. In order to make this transient period milder, the desired v is modified as given in Figure 6.18, instead of a simple step one. In Figure 6.18, v is defined as $v = 0.2\frac{t^2}{9}$ for $0 \le \mathrm{t} < 3$ and $v = 0.2$ for $\mathrm{t} \ge 3$, where t is the simulation time. Considering a long duration of operation of the snake robot for a given task in a real world environment, using a head speed profile as in Figure 6.18 will not affect the overall task.

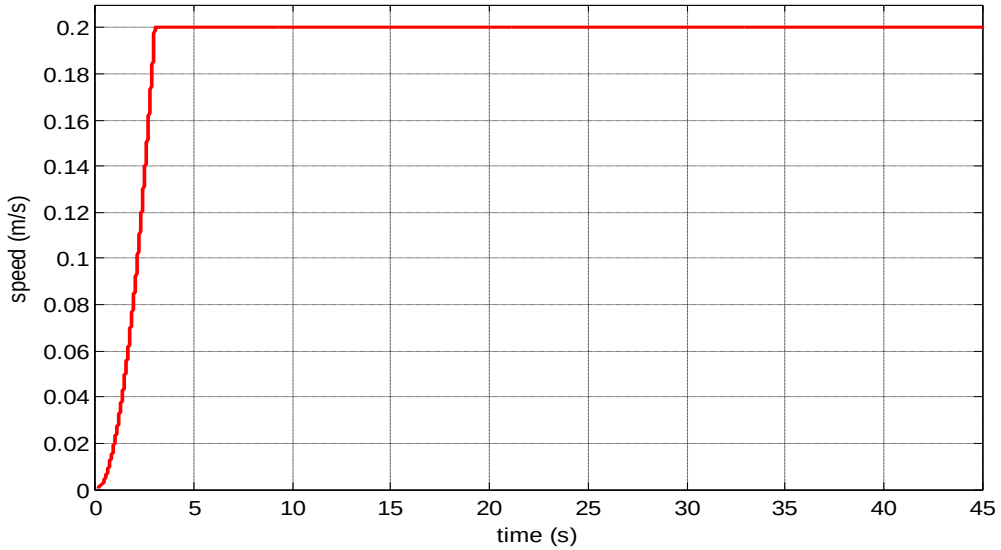


Figure 6.18: Desired speed of the robot head along the modified serpenoid curve

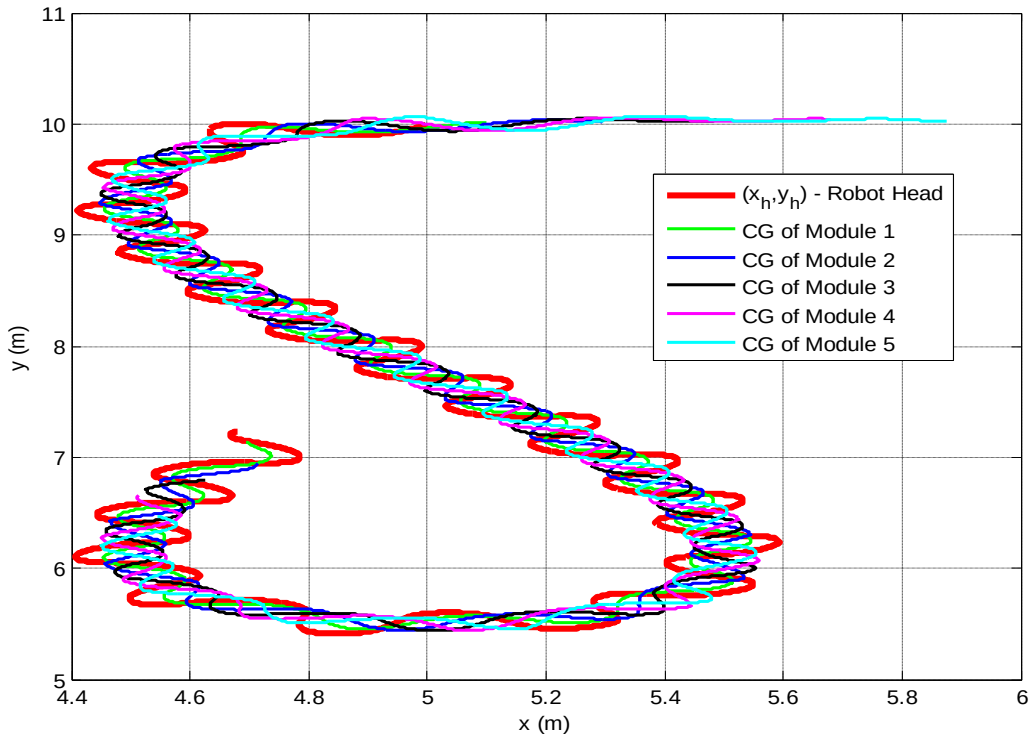


Figure 6.19: Trajectory of the robot head and the modules following the reference $r_r(t) = x_r(t)\hat{i} + y_r(t)\hat{j}$

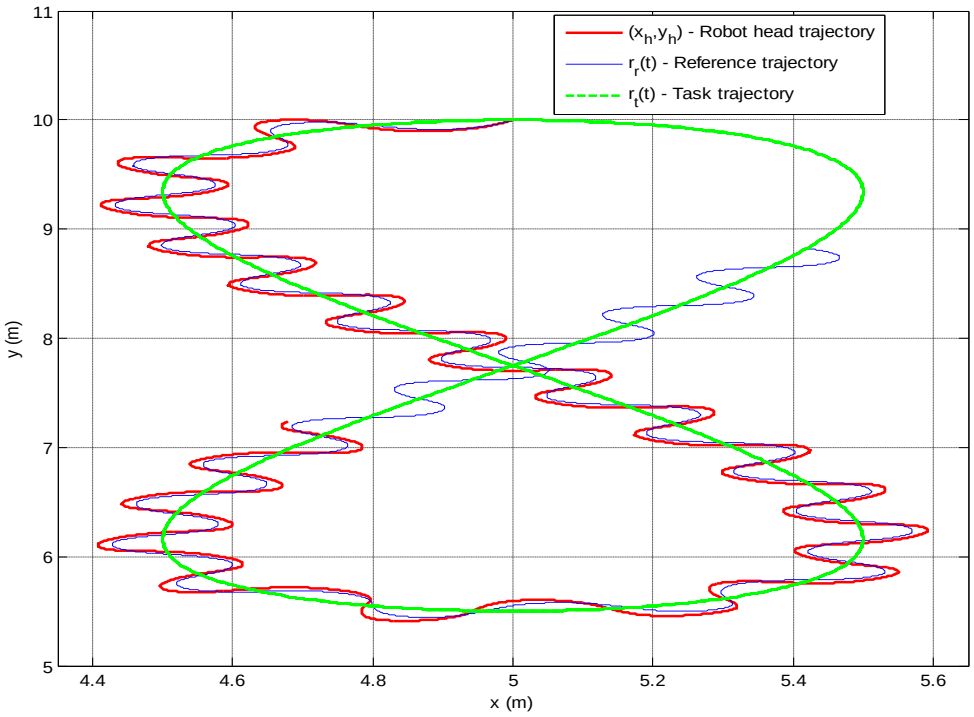


Figure 6.20: (x_h, y_h) with trajectories of reference $r_r(t)$ and task $r_t(t)$

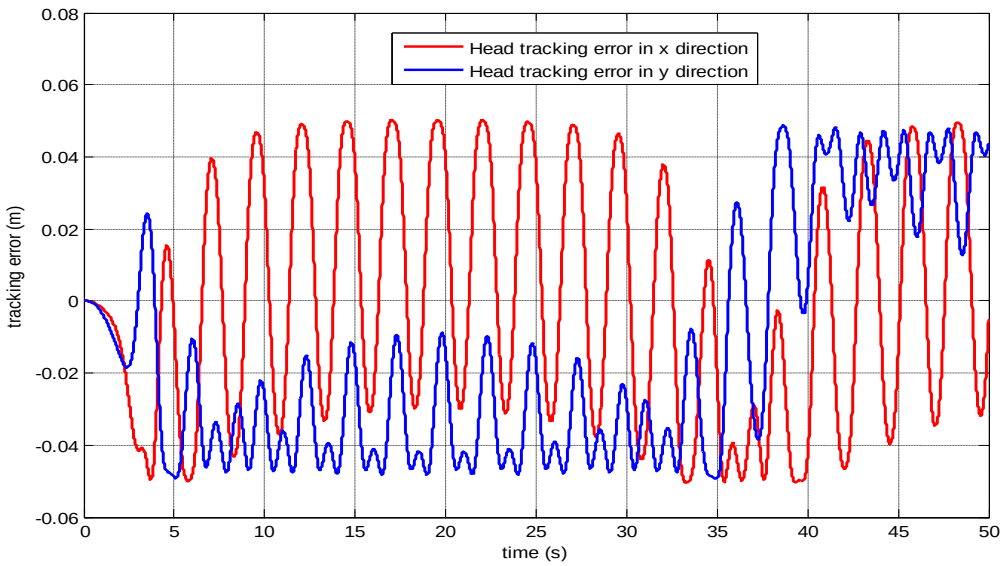


Figure 6.21: Robot head tracking errors e_r^x and e_r^y

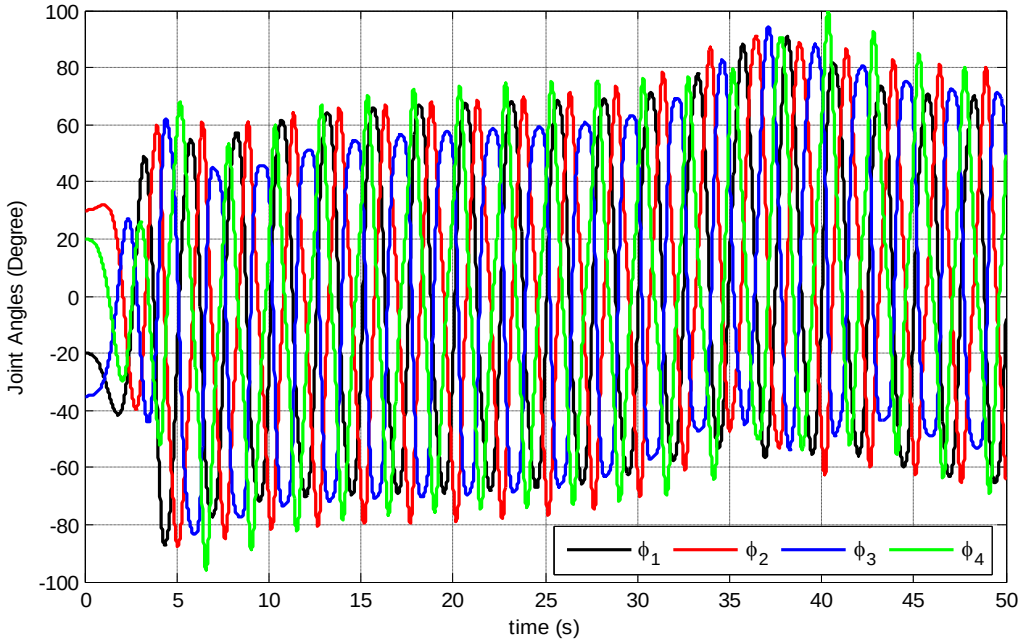


Figure 6.22: Joint angles φ_i, $i = 1,\ldots,4$ of the robot during the locomotion

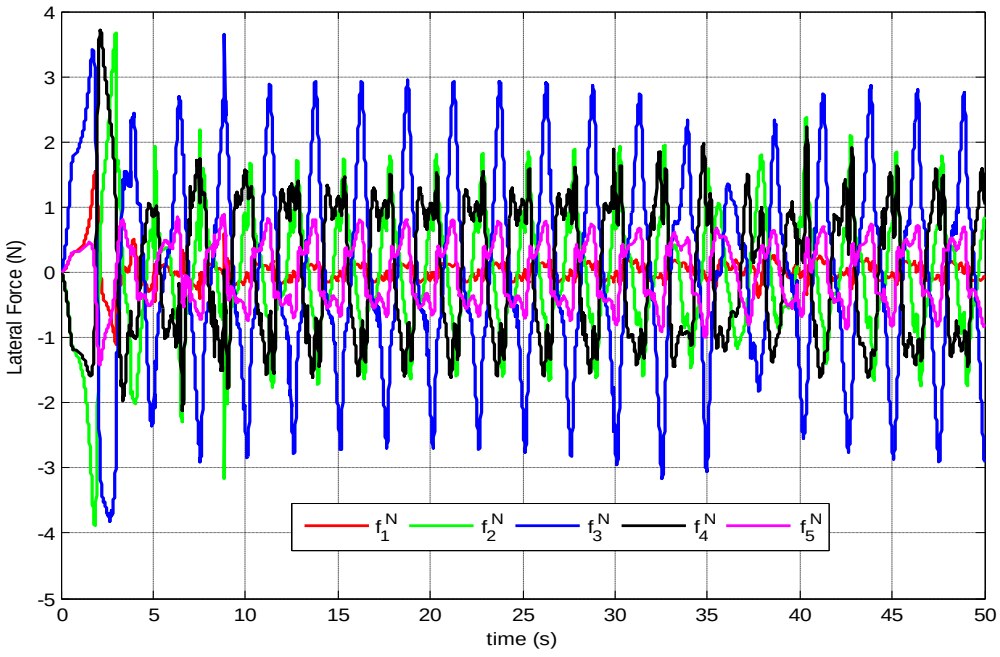


Figure 6.23: Lateral forces $f_1^N,\ldots, f_5^N$ applied by the ground to the modules

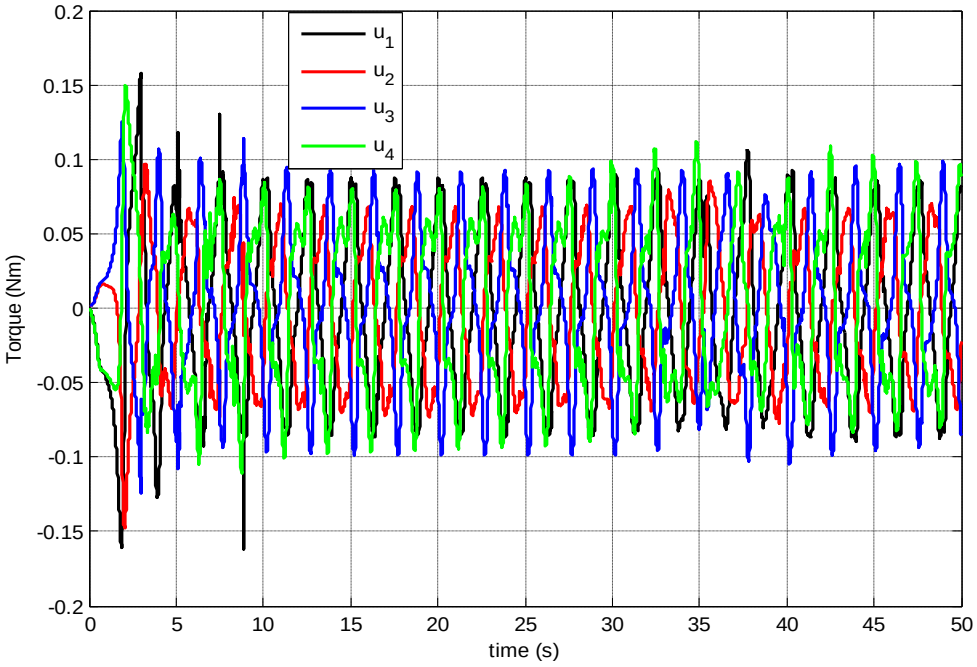


Figure 6.24: Torques $u_1,\ldots,u_4$ applied by the actuators to the robot joints

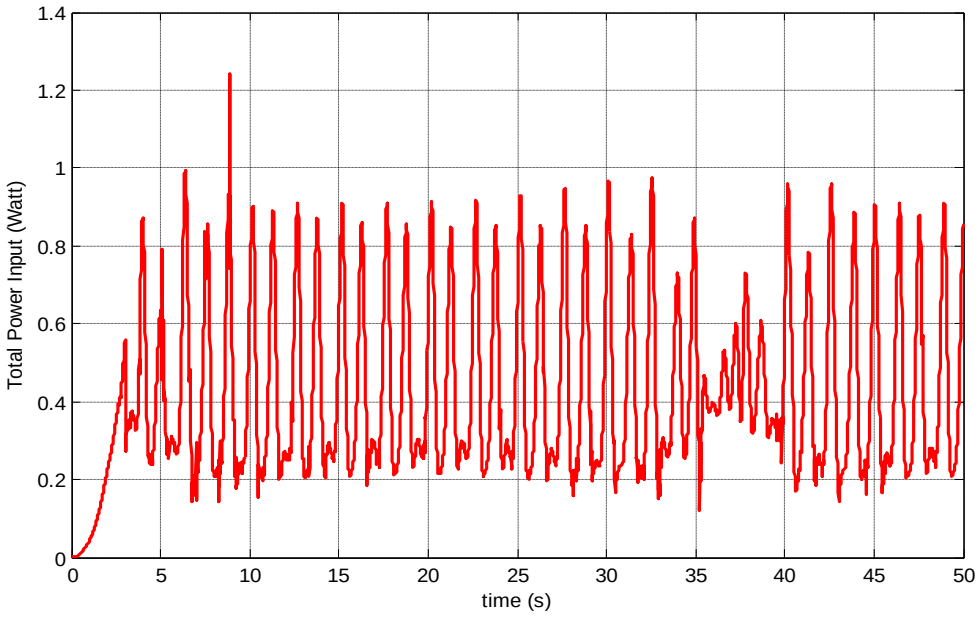


Figure 6.25: Total power input by all of the actuators to the system

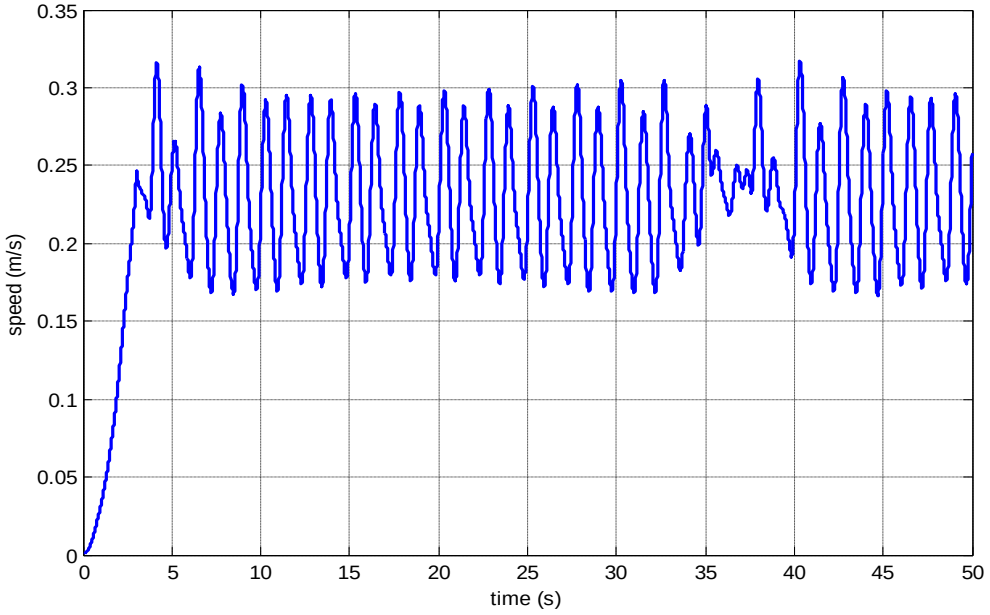
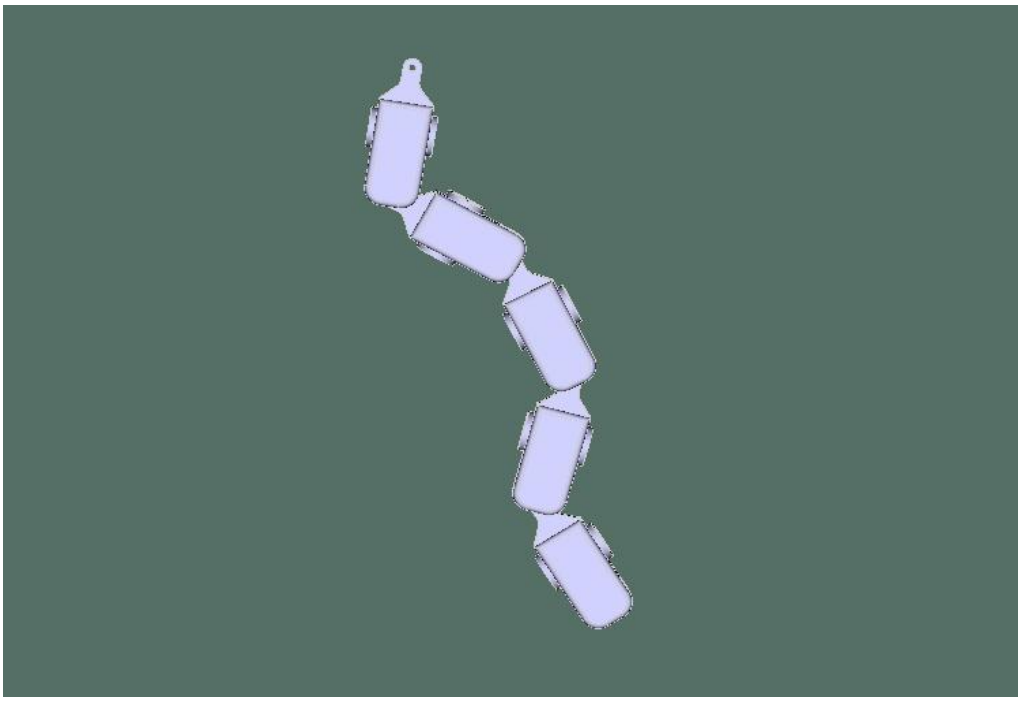


Figure 6.26: Actual speed *v* of the robot head along the trajectory

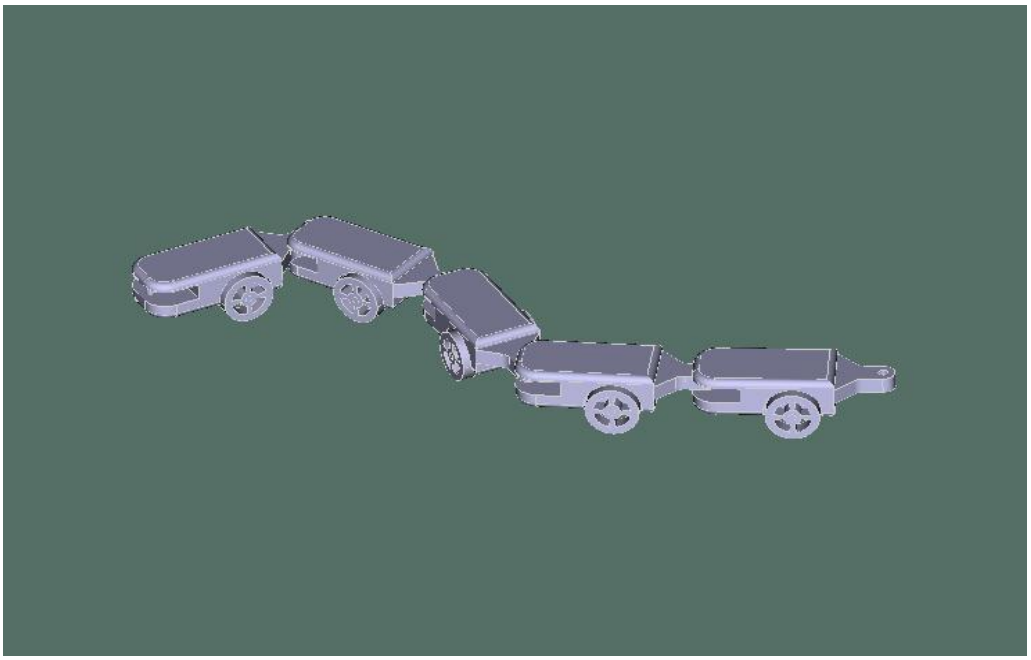
This time the simulation is ran for 50 seconds with $K_n = 2L$ and $\alpha = \pi/3$ again. The task trajectory for this simulation is the eight shaped curve given by the parametric equation $r_t(t) = (-\frac{1}{2}\sin(t\frac{\pi}{2}) + 5)\hat{i} + (2.25\sin(t\frac{\pi}{4}) + 7.75)\hat{j}$ for $0 \le t \le 8$. The results of this simulation are thoroughly summarized by Figure 6.19 to Figure 6.26. Tracking performance of the robot can be clearly seen in Figure 6.19, Figure 6.20 and Figure 6.21. It should be noted that, the tracking performance of the robot is directly related with the natural frequencies ω_x and ω_y, path speed *v*, task trajectory $r_t(t)$ and serpenoid curve parameters K_n and α. When ω_x and ω_y are increased surely the tracking errors will decrease. Tracking performance will decrease if the desired speed of the head along the trajectory *v* is increased. As have been discussed in section 6.2, when K_n and α are increased, the serpenoid curve becomes more wavy, that is, the frequency of undulations along the serpenoid curve increases. This increase implies that the rate at which the

curvature of the curve changes increases. Hence, larger K_n and α brings about a deterioration in tracking performance. On the same basis, task trajectories with a high rate of change of curvature are difficult track, by intuition. For example, tracking errors are minimal if the given task trajectories are linear or perfect circles, since their curvatures are always constant. That is why natural frequencies ω_x and ω_y are increased when the task trajectory is switched from a linear one to a more complicated eight shaped curve given in Figure 6.20. Interestingly, if K_n is decreased, deviation from the given trajectory increases which is illustrated in Figure 6.3. When Figure 6.22, Figure 6.23 and Figure 6.24 are examined it can be seen that the locomotion is performed smoothly and stably. That is to say, no matter how abruptly the task trajectory changes its curvature, no radical jumps on the time histories of $\varphi_1,\ldots,\varphi_4$, $f_1^N,\ldots,f_5^N$ and $u_1,\ldots,u_4$ are observed. That is mainly because our modified serpenoid curve behaves like a buffer between the task trajectory and the snake robot, smoothing out sharp turns and twists. It is like; the differential equation solver (ode 15i) receives $r_t(t)$ as input and interprets it in a form which is suitable for the snake robot to "understand". In Figure 6.25, total power consumption during the locomotion is plotted, while in Figure 6.26 the actual tracing speed *v* is plotted. Total energy consumption is simply the area under the total power input curve (Figure 6.25) and it turns out to be *21.217 Joules* for this simulation. Striking similarities between the trends in these figures are not unexpected since power consumption is directly proportional to speed. In Figure 6.26 it should be noted that the actual *v* deviates from the desired one given in Figure 6.18. This deviation is directly related with the tracking errors and correlates with the trends given in Figure 6.19, Figure 6.20 and Figure 6.21. Graphical representation of the dynamic simulation is constructed in Matlab virtual reality toolbox and is attached to Matlab Simulink in real time.



a) t = 12 seconds

b) t = 41 seconds



c) t = 43 seconds

Figure 6.27: Snapshots from the Matlab VRML during the locomotion simulation

A few snapshots recorded at selected time instances during the simulation are given in Figure 6.27. The graphical model of the robot is built according to the mentioned specifications and is connected to the related configuration variables of the system environment. These configuration variables are the result of the simulation which is run online. Simply reducing the parameter K_n , with everything else kept the same, results in an increase in tracking performance. To demonstrate, K_n is decreased from 2L to L and the simulation is ran again. The simulation results for $K_n = 2L$ are summarized by the figures from Figure 6.28 to Figure 6.34. In Figure 6.28, Figure 6.29 and Figure 6.30, it is seen that even though maximum values of the tracking errors remained roughly the same, the tracking performance is improved, nevertheless. That is because the errors became less oscillatory. Also, since it is a derivative term, the maximum error in tracing

speed *v* decreased which can be seen if Figure 6.29 and Figure 6.26 are compared. When Figure 6.32 and Figure 6.23 are compared to each other it is observed that by decreasing K_n from 2L to L, the lateral forces acting on the modules decreased although there has been an increase in the initial response. However, when Figure 6.33 is examined, where torques applied by the actuators are given, and compared with Figure 6.24, it is notable that by decreasing K_n from 2L to L, the actuator torques are decreased. This decrease in the applied actuator torques is the general trend during the locomotion of the robot. In order to explain the increase in the lateral forces, equation (5.34) must be understood. There are two main constituents of the Lagrangian multipliers and hence the lateral forces: the ones caused by the input torques and the ones caused by the centrifugal (velocity) terms. If the ones caused by the input torques dominate, then the same change trends exist for lateral forces and the input torques, and if the ones caused by the velocity terms $\dot{q}$ dominate then the opposite change trends exist. Thus in our case it can be

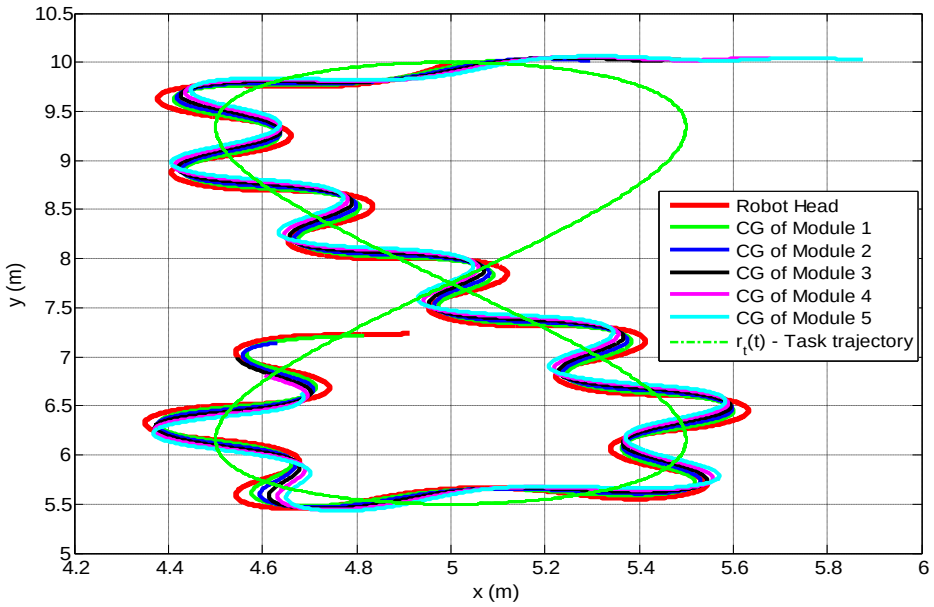


Figure 6.28: Trajectory of the robot head and modules following the reference superimposed on $r_t(t)$

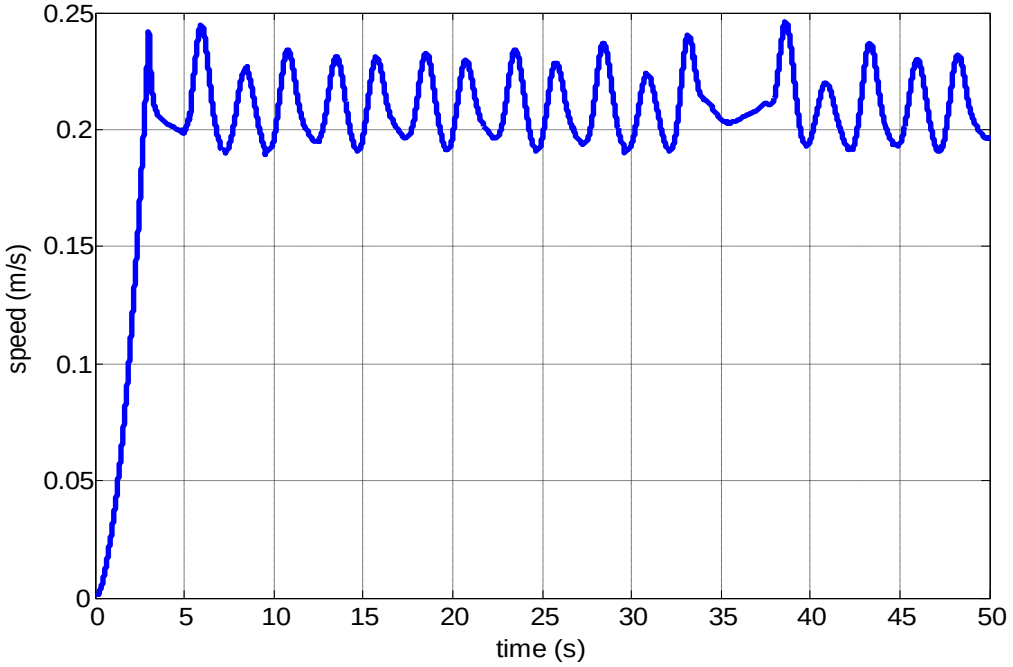


Figure 6.29: Actual speed v of the robot head along the trajectory

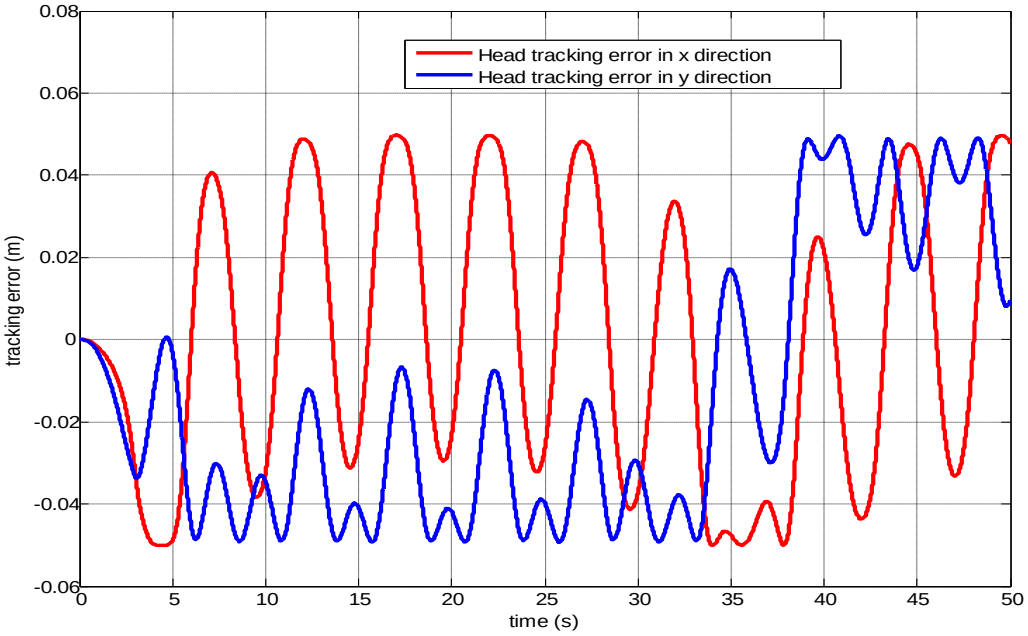


Figure 6.30: Robot head tracking errors e_r^x and e_r^y

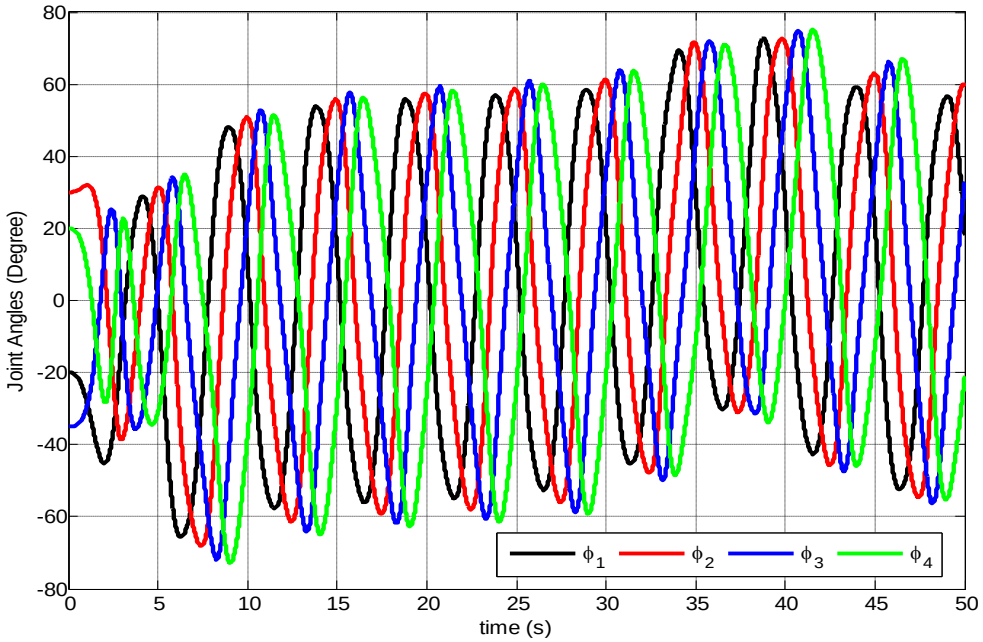


Figure 6.31: Joint angles φ_i, $i = 1,\ldots,4$ of the robot during the locomotion

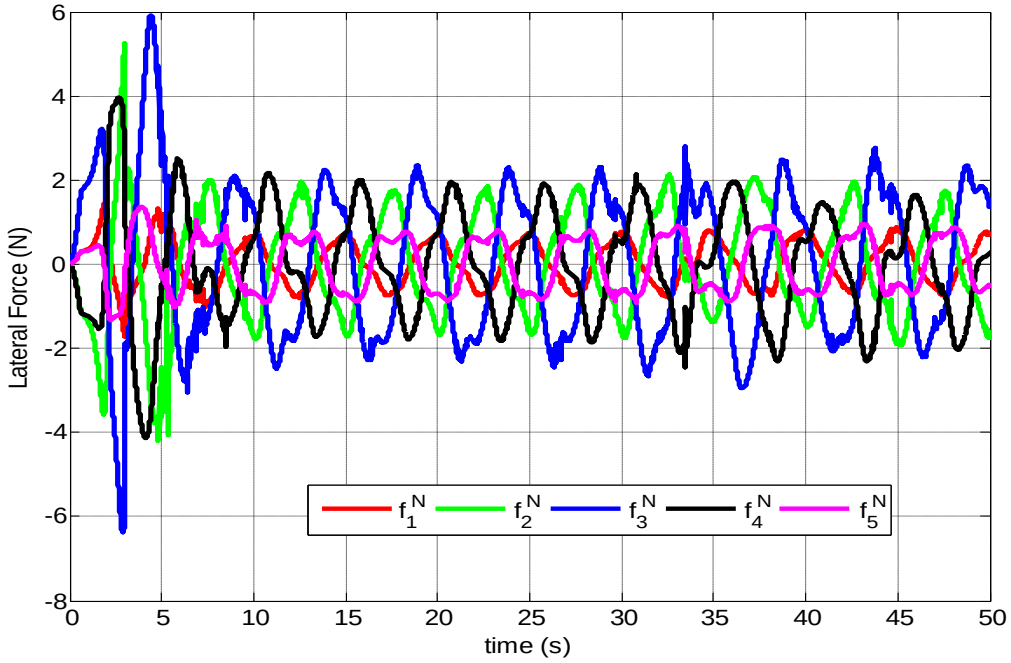


Figure 6.32: Lateral forces $f_1^N,\ldots, f_5^N$ applied by the ground to the modules

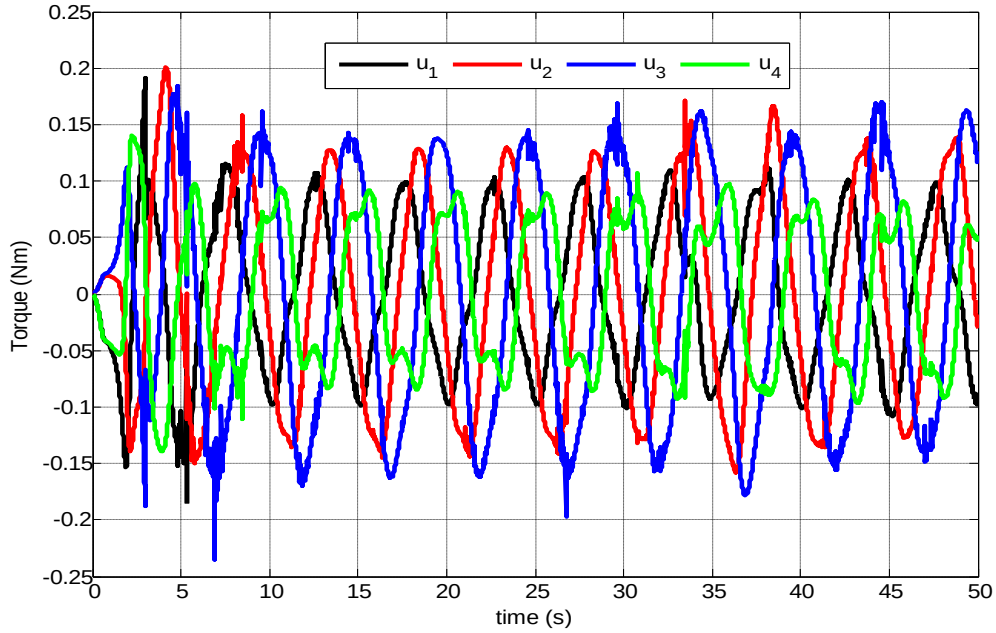


Figure 6.33: Torques $u_1,\ldots,u_4$ applied by the actuators to the robot joints

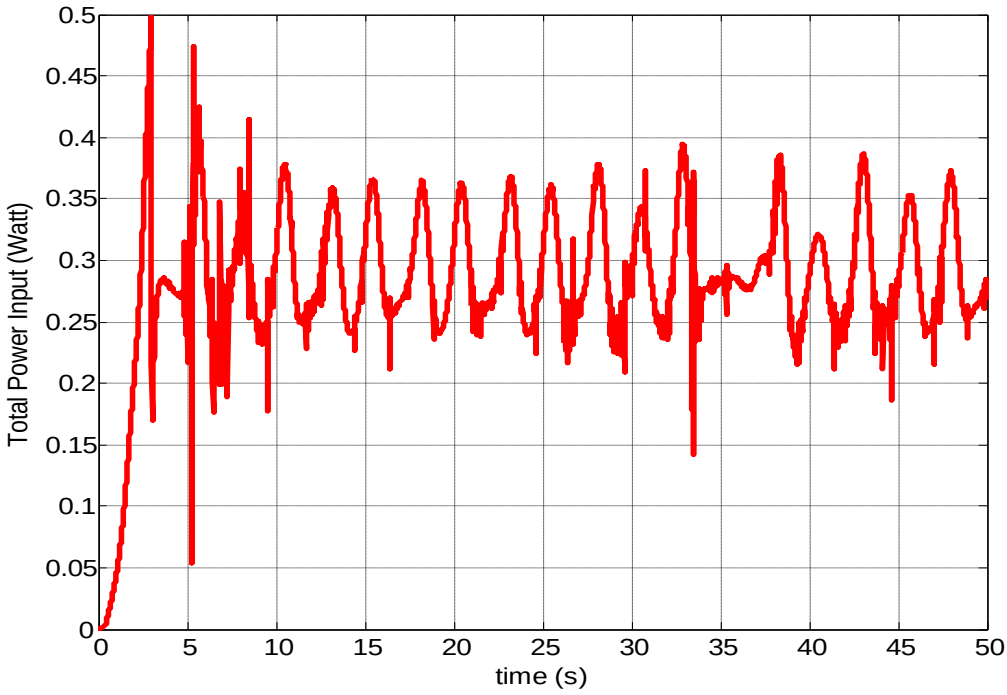


Figure 6.34: Total power input by all of the actuators to the system

concluded that if K_n is increased from *L* to *2L*, although torques applied by the actuators decrease, increase in the contribution of the velocity terms dominates. It is intuitive to conclude that the contribution of the velocity terms in equation (5.34) increase with an increase in K_n, α, tracing speed *v* and with a decrease in radius of curvature of task trajectory $r_t(t)$. Total power input *P(t)* to the system by all of the actuators is given in Figure 6.34, which again correlates with the plot of the tracing speed *v* in Figure 6.29. Total energy consumed during a locomotion period of 50 seconds (which corresponds to a 10 m of total path length traveled) is calculated to be *14.169 Joules*, from the area under the curve in Figure 6.34. That means, the energy consumption of the snake robot decreased about 33.2 % (from *21.217 Joules* to *14.169 Joules*) simply by decreasing the parameter K_n from 2L to L, while the speed at which it has locomoted remained the same. It is interesting to note that although torques applied by the actuators increased total energy consumption decreased. This is clearly due the fact that the rates at which the joint angles φ_i, $i=1,\ldots,4$ changed, i.e., joint velocities $\dot{\varphi}_i$, $i=1,\ldots,4$, remarkably decreased. This can be observed if Figure 6.22 is compared with Figure 6.31, where the time histories of joint angles are given for the case $K_n = L$. Thus, it can be concluded that, there are some favorable values for K_n and α for a given path with a given tracing speed *v* such that the robot consumes less energy or minimizes some (to be selected) performance measure. Clearly, it would be quite meaningful to attempt to determine them and this will be exactly the scope of the next chapter, Chapter 7.

CHAPTER 7

OPTIMALLY EFFICIENT LOCOMOTION

In this chapter, the conditions under which the locomotion of the modular snake robot becomes optimally efficient will be investigated. As in any optimization problem, a performance measure will defined. Afterwards, simulations of the robot to obtain optimal parameters will be performed. Then the results of the simulations will be presented and the obtained results will be discussed. Finally, how the selection of eigenvalues of the linear controller (i.e. selection of natural frequencies ω_x and ω_y) and different tangential friction coefficients affect the locomotion performance and the optimal parameters will be elaborated on.

7.1 Determination of the Objective Function

Firstly, it should be noted that our locomotion, and hence to be found optimally efficient locomotion, is confined to our modified serpenoid curve introduced before. That is to say, it is assumed that what is best for the snake robot in terms of locomotion is our modified serpenoid curve and it is attempted to determine its parameters which optimize the performance measure or the objective function. It is assumed that a task trajectory $r_t(t)$ is given to track, with a certain tracing speed *v*, and the environment of locomotion (μ_N and μ_R). The aim here is to grasp the overall picture of efficiency dynamics of the robot locomotion, rather than searching for the exact numerical values of the parameters that optimize the objective function, using some methods like simulated annealing, genetic

algorithms etc. For that reason, it is more vital to determine "relations" instead of determining of optimal points, which is the core interest of classical optimization problems. Whatever method is used for attaining the dynamics of the most efficient locomotion, it is necessary to know exactly and quantitatively what is sought for, that is, it is necessary to determine a sound performance measure simple, yet comprehensive enough for leading us to where actually desired. Hence, in order to determine a reasonable measure, it should be made clear what is "good" and what is "bad" for the locomotion in a universal and objective manner.

As a starting point, and as one of the most fundamental and reasonable measures in dynamics, it is required that the locomotion is performed by consuming low energy. Furthermore, it is deliberately required that the robot "moves" as the name locomotion implies. If solely the total energy consumption *E* is considered as a measure, certainly the most efficient action would be to stay at rest indefinitely as it requires zero amount of energy. Obviously, this is not something desired, at all. Referring to Figure 7.1, it is seen that while the robot undulates about the given task trajectory $r_t(t)$ depending on the initial winding angle α, the robot head goes sideways. That is to say, if α is increased, the deviation of the robot head from the given task trajectory also increases. Obviously, this deviation is minimal (zero deviation) when $\alpha = 0$ (it is the case when the modified serpenoid curve $r_r(t)$ coincides with the task trajectory $r_t(t)$) and maximal when $\alpha = 90^\circ$. It is seen that the displacement of the head to the sideways does not actually serve for locomotion. For that reason, instead of the curve length s_{act} of the actual path of the robot head $(x_h(t), y_h(t))$ between the starting point $(x_h(t_0), y_h(t_0))$ and the end point $(x_h(t_f), y_h(t_f))$ of the locomotion, the curve length p_{eff} is considered as the measure of the locomotion displacement. Here t_0 is the initial time (starting time) of the locomotion at which all of the system is at rest and always taken as zero, while t_f is the final time of the locomotion. Hence, p_{eff} is called the effective displacement of the locomotion and it is defined to be the curve length of $r_t(t)$

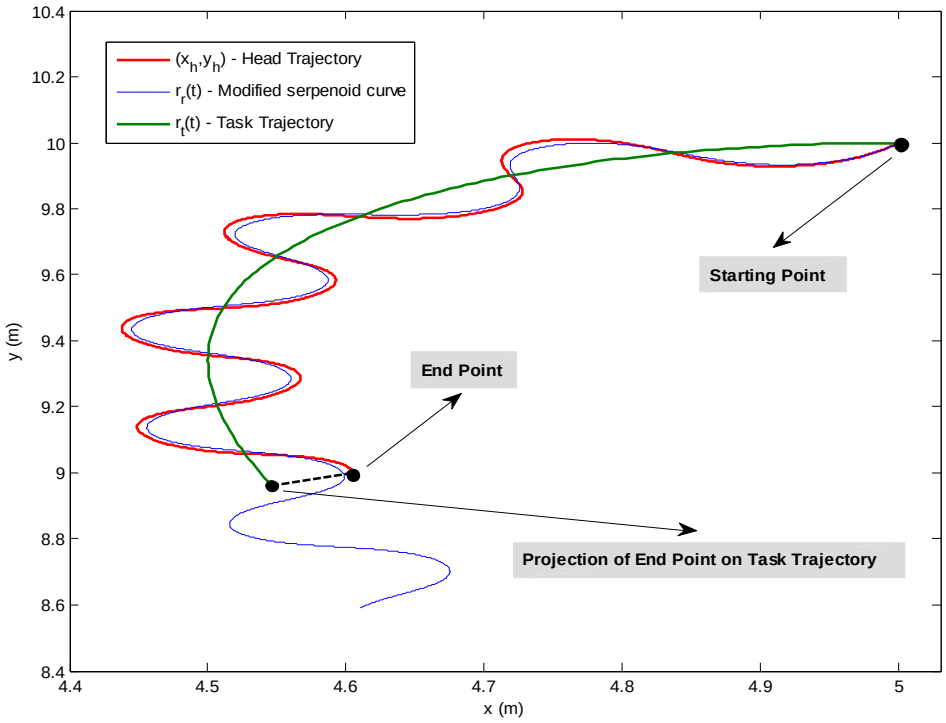


Figure 7.1: Illustration of effective locomotive displacement of the robot

between the starting point $(x_h(t_0), y_h(t_0))$ and the projection of $(x_h(t_f), y_h(t_f))$ on $r_t(t)$ itself. It should be noted here again that the parameter t in $(x_h(t), y_h(t))$ and in $r_t(t)$ are not to be taken as the same. That is, while the parameter t in $(x_h(t), y_h(t))$ is explicitly the locomotion time, the t in $r_t(t)$ is a generic one. Hence, if t' is let to be the independent generic parameter of $r_t(t')$, the equation for p_{eff} is given as follows

$$p_{eff} = \int_{t'=0}^{t'=t'_f} \sqrt{\dot{x}_t(t') + \dot{y}_t(t')} dt' \qquad (7.1)$$

where $t_f^{'}$ is the value of $t^{'}$ at which the distance from the point $(x_h(t_f), y_h(t_f))$ to the $r_t(t^{'})$ is minimal i.e., at $t_f^{'}$, $\left(\left[x_h(t_f)-x_t(t^{'})\right]^2+\left[y_h(t_f)-y_t(t^{'})\right]^2\right)$ is minimal. Thus, $t_f^{'}$ is the solution for $t^{'}$ of the equation given by

$$g(t^{'})=2\left[x_h(t_f)-x_t(t^{'})\right]\dot{x}_t(t^{'})+2\left[y_h(t_f)-y_t(t^{'})\right]\dot{y}_t(t^{'})=0 \qquad (7.2)$$

Since $r_t(t^{'})$ is completely and explicitly given, the root of equation (7.2) can readily be found numerically or analytically depending on $r_t(t^{'})$. Here, it is assumed that equation (7.2) always has a unique root. The conditions that violate this assumption include when the circle centered at the end point of locomotion $(x_h(t_f), y_h(t_f))$ with a radius of $\sqrt{\left[x_h(t_f)-x_t(t_f^{'})\right]^2+\left[y_h(t_f)-y_t(t_f^{'})\right]^2}$ intersects more than one point on curve $r_t(t^{'})$. Observing Figure 7.1, it can be concluded that these conditions correspond to the task trajectories having radius of curvatures smaller than the amplitude of the modified serpenoid curve (measured from corresponding symmetrical line) along the related section of the curve. These conditions constitute a subset of the previously mentioned reference task trajectories which are not feasable. Thus, it would be quite meaningful if the perfomance measure includes the term related to the total energy consumption of the robot per unit effective displacement of locomotion, which turns out to be $\frac{E}{p_{eff}}$. Observing the results obtained in the simulation performed in section 5.5 where the robot is made to follow a straight line "directly" without utilizing a serpenoid curve, low energy consumption rates at steady state are obtained. What is more, this required power input given in Figure 5.10 is to overcome the friction. The same simulation is repeated by setting the tangential friction coefficient $\mu_R=0$ and damping in the joints $d_\phi=0$ and as expected, the power consumption reduced to zero at steady state. That is because, once all of the modules's orientations are aligned with the straight line and once the specified speed is reached, no action is required from the

actuators anymore. However, if the task trajectory somehow deviates from a straight line at some point during the locomotion, actuators become useless, as the singularity will have already occurred by setting the same orientation for all of the modules. From this point on, actuators start to apply maximum amount of torque they could without any help or eventually make the modules side slip by causing the lateral forces $f_i^N, i = 1,\ldots,5$ exceed the maximum amount of friction force $f_{\max}^N = \mu_N mg$ that the ground can apply. This is the case even when tangential friction (rolling friction) is present as illustrated in the simulation performed in section 5.5. One might argue that when the tangential friction is present, even when all of the modules start to align to a long enough straight line (i.e., joint angles $\phi_i, i = 1,\ldots,4$ start to converge to zero) and even when the specified speed is reached, energy dissipation due to tangential friction would result in a decrease in the speed of the robot head and consequently, to regulate the speed, the actuators would inevitably apply some amount of torque. Hence, this constant application of torque by the actuators will keep the robot "alive" and prevent it from converging to the singularity. Unfortunately, the simulation results in section 5.5 tell us this is not the fact. This might be attributed to the fact that the regulatory actions of the joint actuators to overcome the friction do not serve for the purpose of preventing the modules from aligning with the straight line. This might be the physical reason for the mathematical lemma given in section 5.5. All of these lead to the point that the notion of minimizing the energy consumption of the robot is not enough to demote divergence or having large magnitudes (which might be the case even when the modified serpenoid curve is utilized) of actuator torques. It is concluded from the results obtained in Chapter 6 that, setting small values for K_n and α decreases the oscillation frequency and wave height of the serpenoid curve, respectively. Also, small values of α generates a modified serpenoid curve that looks similar to the original task trajectory[7] (or at least with small deviations).

[7] Indeed, setting α to zero generates a curve identical to the task trajectory.

When those cases coincide with the case when the task trajectory includes subsections with relatively low amount of rate of change of curvature, actuators apply relatively large magnitudes of torques. To sum up, decreasing K_n and α decreases the total energy consumption, increases the effective locomotion displacement p_{eff}, and decreases the deviation of the actual robot path from the nominal one (task trajectory r_t). However, at the same time, decreasing K_n and α leads to an increase in the torques applied by the actuators to the module joints. If that was not the case, $\alpha = 0$ would be set simply and there would be no need for a serpenoid curve or its proposed modified version. Hence, there well may be cases where, although the robot locomotes with low power requirement, large amount of input torques are required from the actuators, which would be impractical or even impossible. Even if the required torques are applicable, large torques may lead to large amount of lateral forces which in turn may lead to side slip or wear of module wheels in long times of operation of the robot. Thus, instead of considering applied joint torques to demote in the performance measure, directly to consider the lateral forces applied by the ground to the modules would be more meaningful. That is because, observing equation (5.34) and (5.35), there is a direct relation between the actuator torques and lateral forces. Thus, divergence of the actuator torques implies the divergence of lateral forces applied to the modules and vice versa (if the velocity of the configuration variables $\dot{q}$ stay bounded, which is always the case). Also, by considering the lateral forces applied, the centrifugal (velocity) terms in equation (5.34) are also covered, which means any abrupt twists and turns in actual path followed by the robot are also demoted. From Figure 7.1, it is seen that the actual path length followed by the robot head is the curve length of the robot head trajectory and is simply given by

$$s_{act} = \int_{t=0}^{t=t_f} \sqrt{\dot{x}_h(t) + \dot{y}_h(t)}dt \qquad (7.3)$$

The term $\frac{s_{act}}{p_{eff}}$ becomes unity when the modified serpenoid curve is not utilized at all and the robot head perfectly follows the task trajectory. In the light of our discussions about the locomotion of the robot, generally, the more the robot head follows a wavy path (the more the rate at which the radius of curvature of the path changes) constrained to our modified serpenoid curve, the less become the lateral forces experienced by the modules; however, the more becomes the term $\frac{s_{act}}{p_{eff}}$.

The term $\frac{s_{act}}{p_{eff}}$ implicitly quantifies the relative amount of time the modules are exposed to the lateral forces applied by the ground, in a compact and simple manner. In the ideal case (where the head exactly tracks the reference task trajectory $r_t(t)$) $\frac{s_{act}}{p_{eff}} = 1$, while $\frac{s_{act}}{p_{eff}} > 1$, otherwise. That means, although the magnitude of the lateral forces decreases, the amount of time that these forces are experienced by the modules increases, up to some certain value and is detrimental for structural integrity. Thus, trying to minimize the term $\left\| f_{rms}^{N} \right\| \frac{s_{act}}{p_{eff}}$, instead of simply the term $\left\| f_{rms}^{N} \right\|$, seems quite sound. Hence, determining a measure of the form

$$H = \frac{1}{\alpha_1 \frac{E}{p_{eff}} + \alpha_2 \left\| f_{rms}^{N} \right\| \frac{s_{act}}{p_{eff}}} \tag{7.4}$$

would be rather meaningful and our aim will be to maximize the performance measure *H*. Here the term $\left\| f_{rms}^{N} \right\|$ in equation (7.4) denotes the norm (Euclidean norm) of the root mean square-lateral force vector f_{rms}^{N} whose elements are simply the root mean squares of the elements of original lateral force vector f^{N}.

In other words, f_{rms}^{N} is defined as

$$f_{rms}^{N} = \begin{bmatrix} f_1^{rms} \\ f_2^{rms} \\ f_3^{rms} \\ f_4^{rms} \\ f_5^{rms} \end{bmatrix}, \text{ where } \quad f_i^{rms} = \sqrt{\frac{1}{t_f}\int_{t=0}^{t=t_f} \left[f_i(t) \right]^2 dt} \tag{7.5}$$

leading to

$$\left\| f_{rms}^{N} \right\| = \sqrt{\left[f_1^{rms} \right]^2 + \ldots + \left[f_5^{rms} \right]^2} \tag{7.6}$$

and where α_1 and α_2 are simply weighting coefficients. Also it should be noted that by including the lateral force vector f^N (instead of the actuator input vector u), the performance measure has been split into two independent components. One of them is *efficiency* and the second one is *sustainability*. Obviously, the term $\alpha_1 \frac{E}{p_{eff}}$ quantifies how effective the robot locomotes and the term $\alpha_2 \left\| f_{rms}^{N} \right\| \frac{s_{act}}{p_{eff}}$ quantifies the sustainability and both of them have the units of force, namely Newtons. Furthermore, to give them equal emphasis, weighting coefficients are intuitively set as $\alpha_1 = \alpha_2 = 1$ in equation (7.4). The sustainability term $\alpha_2 \left\| f_{rms}^{N} \right\| \frac{s_{act}}{p_{eff}}$, gives a measure about both the singularity condition and the severity of the wear on the modules. Inclusion of the both of the terms in the performance measure *H* is necessary for a compromise between the efficiency and the sustainability.

7.2 Simulation of the Dynamic System

As have been mentioned before, the snake robot is modeled in Matlab Simulink based on the derived equations of motion of the system given by equation (4.25) and equation (3.22) and synthesized controller given by the equation (5.33). A sample task trajectory is selected which is already given as $r_t(t) = (-\frac{1}{2}\sin(t\frac{\pi}{2}) + 5)\hat{i} + (2.25\sin(t\frac{\pi}{4}) + 7.75)\hat{j}$ for $0 \le t \le 8$ and plotted in Figure 6.20. This path has been selected to represent a general trajectory including both straight and curved parts. However, to handle repeated simulations and to decrease the total amount of simulation time, the simulation will be run for 25 seconds instead of 50 seconds. Although it roughly corresponds to the half the total path length spanned it turns out to be enough to represent the general task trajectory profile. A fixed step size of 0.01 seconds is used in Matlab Simulink and the built in ODE solver ode3 has been utilized. Detailed block diagram of the Matlab Simulink model is given in Appendix B. Also, as mentioned before, using Matlab Virtual Reality Modeling toolbox, all of the needed configuration variables of the simulation environment are mapped to the graphical representation of the robot model for online animation. This graphical representation has been created using a 3D solid modeling software and the model is converted to wrml format in order that Matlab Simulink can link it with the block diagrams. The general control architecture as implemented in Matlab Simulink is given in Figure 7.2. Here, each block diagram includes lower level block diagrams (subsystems) and the analytical equations of motions and the control law are implemented in m-files as much as possible. In Figure 7.2, the interpreted reference trajectory means the modified serpenoid curve $r_r(t)$ fitted on the task trajectory $r_t(t)$. All of the components given in Figure 7.2 are working simultaneously, online and in the real time. When it is compared with Figure 6.9 , it is seen that the components of the overall control architecture which are working offline are excluded in Figure 7.2. These components are the *Modified Serpenoid Curve Generator* and the *Optimizer*.

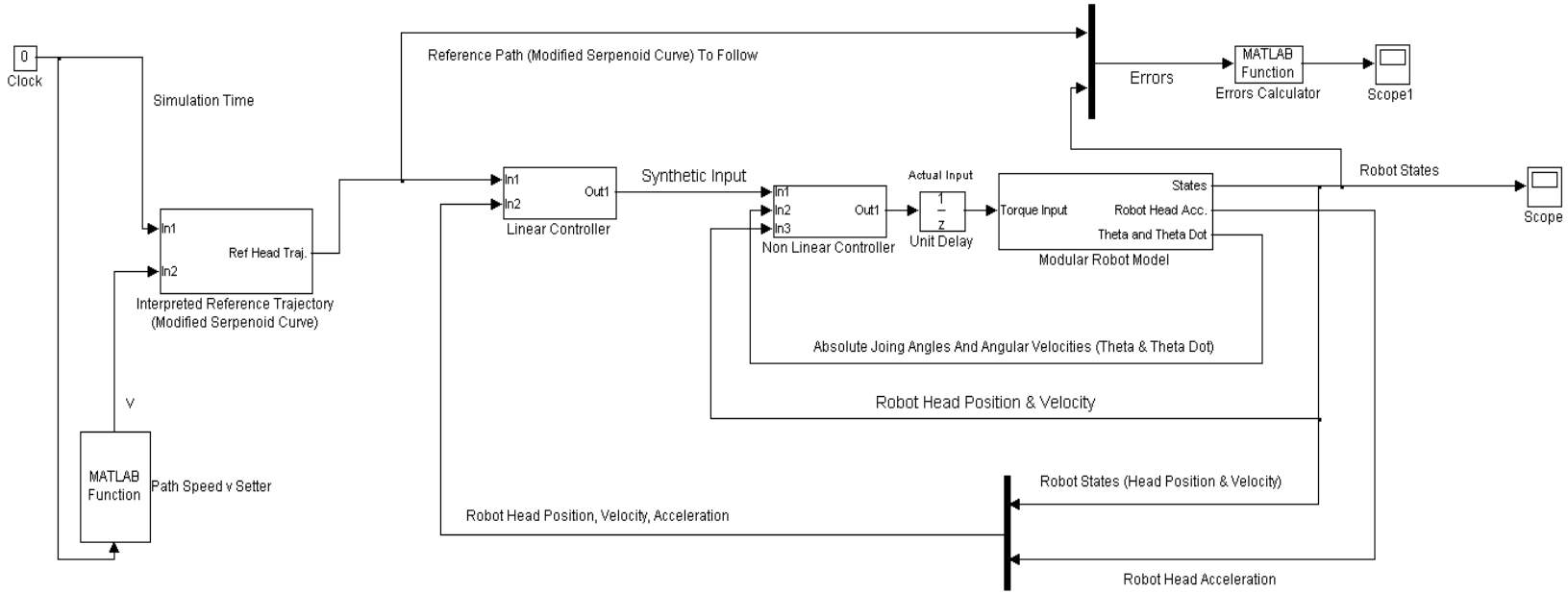


Figure 7.2: Implementation of modeling and control of the robot in Matlab Simulink

Modified Serpenoid Curve Generator, as its name implies, solves the set of differential equations given by equation (6.15) and provides the obtained solution for $r_r(t)$ to the mentioned block diagram in Figure 7.2, before the simulation starts. Optimizer determines the parameters K_n^* and α^* which causes the robot to locomote in the most efficient way. Online and offline layers of the

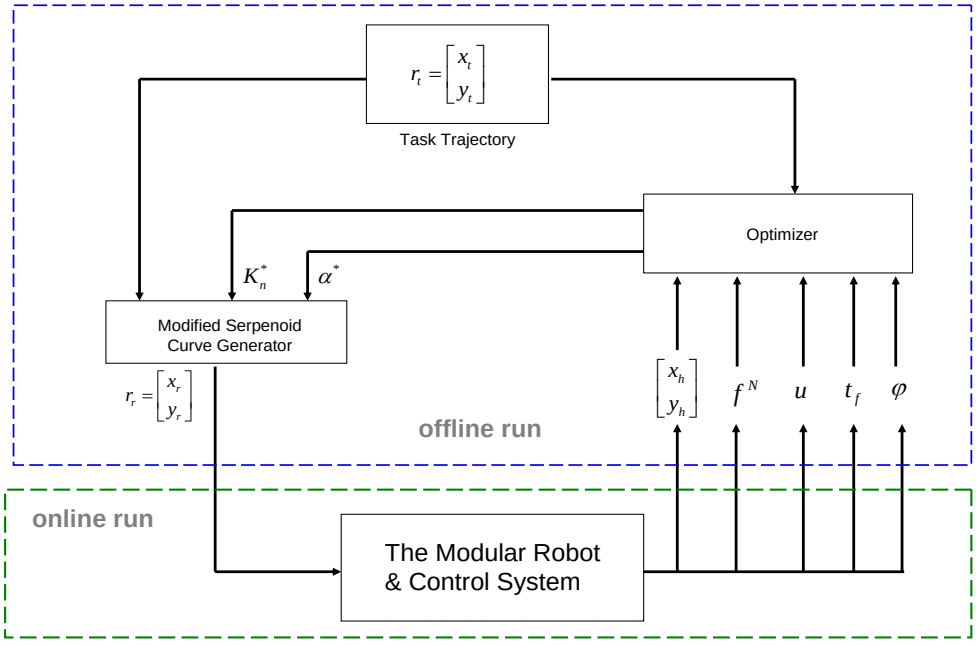


Figure 7.3: Online and offline layers of the overall control architecture

overall control architecture is depicted in Figure 7.3. The missing inputs to the optimizer denoted as dotted lines in Figure 6.9 are clearly given in Figure 7.3. These quantities are time histories of the robot head position $\begin{bmatrix} x_h(t) \\ y_h(t) \end{bmatrix}$, the vector of lateral forces (normal forces) $f^N = \begin{bmatrix} f_1^N & f_2^N & f_3^N & f_4^N & f_5^N \end{bmatrix}'$, the vector of actuator torques (control inputs) $u = \begin{bmatrix} u_1 & u_2 & u_3 & u_4 \end{bmatrix}'$, the final time of the simulation t_f and the vector of joint angles $\phi = \begin{bmatrix} \varphi_1 & \varphi_2 & \varphi_3 & \varphi_4 \end{bmatrix}'$. These

quantities are employed by the optimizer to maximize the performance measure *H* given by equation (7.4). What is meant by the online and offline run is that, the online part is run in the real time while the modified serpenoid curve generator of the offline part provides the online part with the reference trajectory, initially. After the online simulation is completed, the mentioned quantities are fedback to the optimizer in order to determine *H,* and K_n^* and α^* accordingly (after several online simulations).

7.3 Pareto Optimization of the Objective Function

Looking at the performance measure *H* given by equation (7.4) , it is a function of K_n , α, the structural parameters of the modular robot which are *L, J, m,* and the number of modules, the gravitational acceleration *g,* the tangential friction coefficient μ_R, damping constant d_ϕ of the revolute joints, the natural frequencies of the linear controller ω_x and ω_y , the speed at which the robot head traces the modified serpenoid curve *v* and even the task trajectory $r_t(t)$ itself. Considering all of these and assuming that *H* depends on a total of z-many parameters, in order to find the optimal ones it is necessary to span the $\mathbb{R}^z$ space or at least it is necessary to apply the search algorithms or the optimization methods to the $\mathbb{R}^z$ space, whatever they are. Obviously this would require enormous amounts of processing power and time and yet, most of the obtained results may serve for no practical purpose or might not be able to justify the allocated resources. Pareto efficiency is a concept originally from economics first introduced by Vilfredo Pareto, and has been successfully applied to engineering. In its broadest terms, pareto efficient situations are those in which an outcome (a measure in our case) cannot be increased further without hurting at least one player (with a game theoretic approach) [75]. This idea can be adapted to our case such that the performance measure *H* cannot be further increased without changing other

parameters which are assumed to be constant. On that account, considering that the *relations* are more of interest than the *points* and considering what can be *tuned* in a *given* situation, it is meaningful to determine the optimal parameters K_n^* and α^* for a given tracing speed *v* and then determine how K_n^* and α^* changes with the tracing speed *v*. The first series of simulations are run for the parameters given by Table 7.1 for *v* = 0.10 m/s

Table 7.1: Parameters used for the 1st series of simulations

	m (kg)	**J (kg.m^2)**	**L (m)**	μ_R	$\omega_x = \omega_y$ (rad/s)	d_ϕ (N.s/rad)	**v (m/s)**
Value	*0.792*	*0.00269*	*1*	*0.035*	*8*	*0.01*	**0.10**

Firstly the performance measure *H* is constructed on a grid. This grid (region) is spanned by $K_n \in [0.5L, 3L]$ and $\alpha \in [15^\circ, 85^\circ]$ and is composed of a total of 36 points, i.e. each of K_n and α have been partitioned to have 6 equidistant points. Simulation results of the first series of simulations are summarized by Figure 7.4 and Figure 7.5. In Figure 7.4, performance measure *H* has been generated as a surface by running repeated simulations on the mentioned grid points. In Figure 7.5, the contour plot of the performance measure surface is given. It is simply the collection of isolines along which the value of the performance measure is constant, i.e. *H* = c. It gives thorough information about the change trend of *H* as well as the optimal values K_n^* and α^*. It can be concluded that for the parameters given in Table 7.1, *H* has a maximum value of about 0.2030 and attains this maximum at the optimal parameters $K_n^* = 2.5L$ and $\alpha^* = 57^\circ$. Having said that, one point must be clarified: K_n^* and α^* illustrated in Figure 7.3 actually denotes the values after the optimal parameters are found and used for simulation.

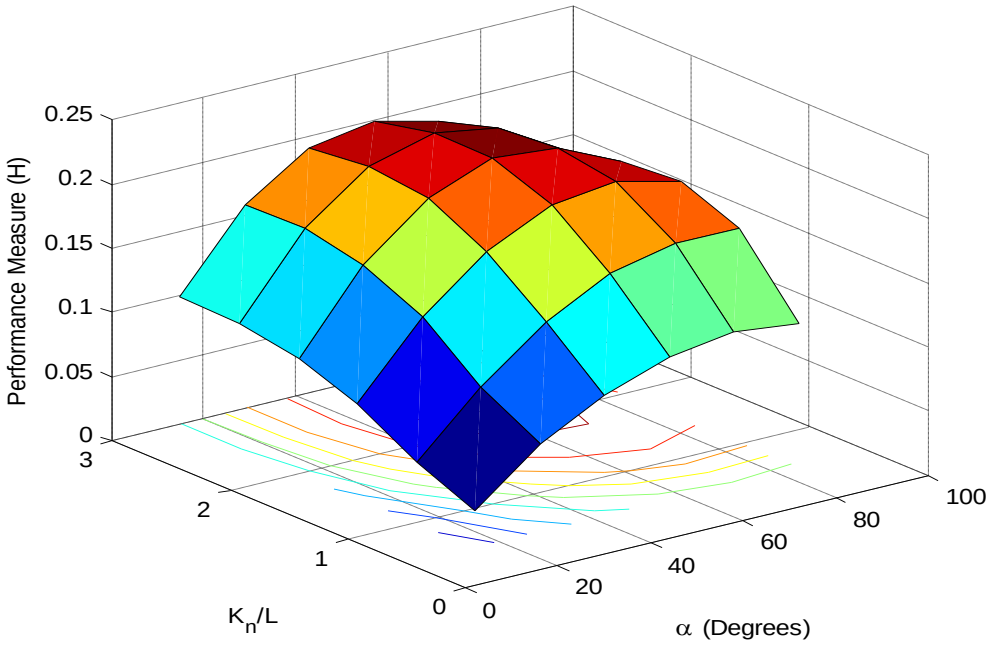


Figure 7.4: Surface of the performance measure H over the region spanned by parameters α and K_n / L for v = 0.10 m/s

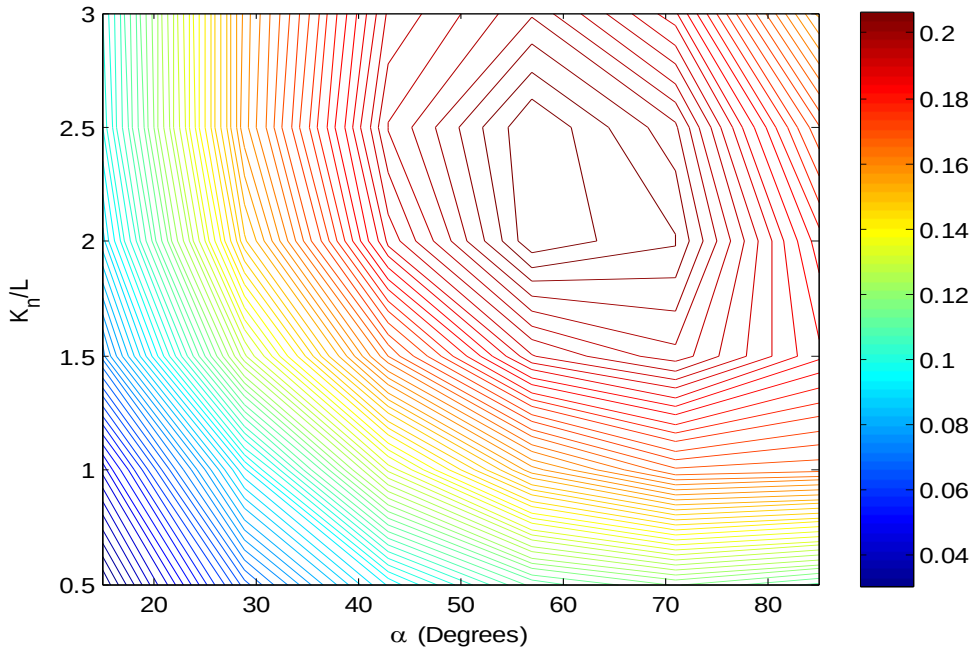


Figure 7.5: Contour plot of performance measure H for v = 0.10 m/s

If a usual cycle of optimization is to be considered, then * symbols should be dropped, i.e. they must be thought as simply K_n and α. Then, the simulation is repeated for the list of parameters given in Table 7.2, i.e. the tracing speed *v* is increased to 0.15 m/s while keeping the rest of the parameters unchanged. The result of this simulation is given by the figures Figure 7.6 and Figure 7.7. When they are examined, it is seen that the maximum value of *H* turns out to be about 0.1880 and it is attained at the optimal values $K_n^* = 2L$ and $\alpha^* = 57^\circ$. Comparing it with the first simulation it is seen that by increasing the tracing speed v from 0.10 m/s to 0.15 m/s, the maximum efficiency of the locomotion decreased from 0.2030 to 0.1880, K_n^* decreased from 2.5*L* to 2*L* while α^* remained the same. Continuing in the same manner, the third and the fourth set of simulations are performed with the parameters given in Table 7.3 and Table 7.4, respectively. In the third set of simulations, the tracing speed *v* is increased to 0.20 m/s and in the fourth to 0.25 m/s. Results of the third set of simulations are given in Figure 7.8 and Figure 7.9, while those of the fourth are given in Figure 7.10 and Figure 7.11. It is observed that by increasing *v* to 0.20 m/s from 0.15 m/s, the maximum of *H* decreased to 0.1630 and K_n^* decreased to 1.5*L*, while α^* remained the same at 57°, again.

Table 7.2: Parameters used for the 2nd series of simulations

	m (kg)	**J (kg.m^2)**	**L (m)**	μ_R	$\omega_x = \omega_y$ (rad/s)	d_ϕ (N.s/rad)	**v (m/s)**
Value	*0.792*	*0.00269*	*1*	*0.035*	*8*	*0.01*	**0.15**

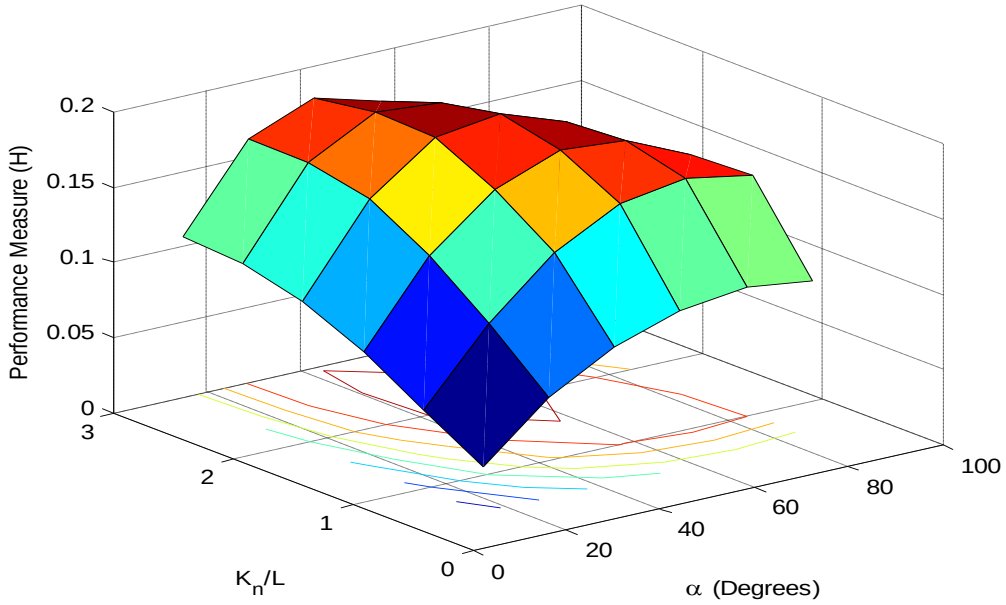


Figure 7.6: Surface of the performance measure *H* over the region spanned by parameters α and K_n / L for v = 0.15 m/s

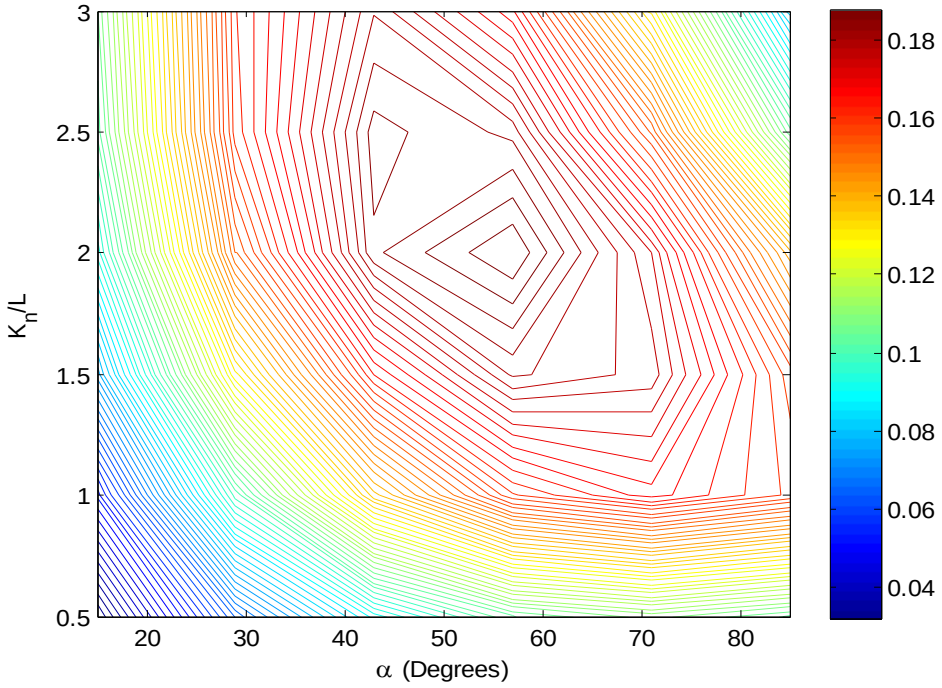


Figure 7.7: Contour plot of performance measure *H* for v =0.15 m/s

When *v* is increased to 0.25, it is notable that, the maximum value of *H* decreases again to about 0.1530, while K_n^* decreases to 1.5*L*, and no change occurs in α^* as before. After performing these four sets of simulations where the tracing speed *v* has been gradually increased, it should be noted that the performance degrades with increasing speed. This can be attributed to the nature of the performance measure *H* defined in equation (7.4). This is because, although there is no reason for the term $\alpha_1 \frac{E}{p_{eff}}$ to change with *v*, it is quite obvious that the term $\alpha_2 \left\| f_{rms}^N \right\| \frac{s_{act}}{p_{eff}}$ will increase with increasing *v*, since to go faster the actuators will apply bigger torques and the velocity terms in equation (5.34) will increase with increasing *v*. Combining the results of all the simulations performed, the pareto frontier is constructed between the tracing speed *v* and the parameter K_n / L and it is given by Figure 7.12. Pareto frontier is simply the set of choices which are pareto optimal with respect a given measure. Any of the points (choices) which lie on the pareto frontier dominates the ones which do not lie on the pareto frontier. However, none of the points on the pareto frontier is dominated by each other.

Table 7.3: Parameters used for the 3rd series of simulations

	m (kg)	**J (kg.m^2)**	**L (m)**	μ_R	$\omega_x = \omega_y$ (rad/s)	d_ϕ (N.s/rad)	**v (m/s)**
Value	*0.792*	*0.00269*	*1*	*0.035*	*8*	*0.01*	**0.20**

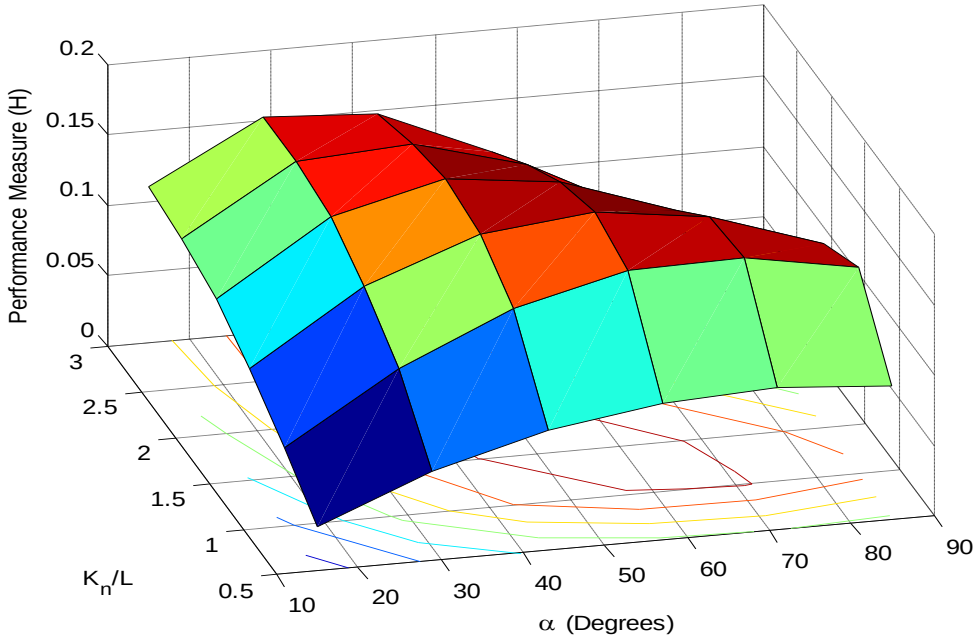


Figure 7.8: Surface of the performance measure *H* over the region spanned by parameters α and K_n / L for *v* = 0.20 m/s

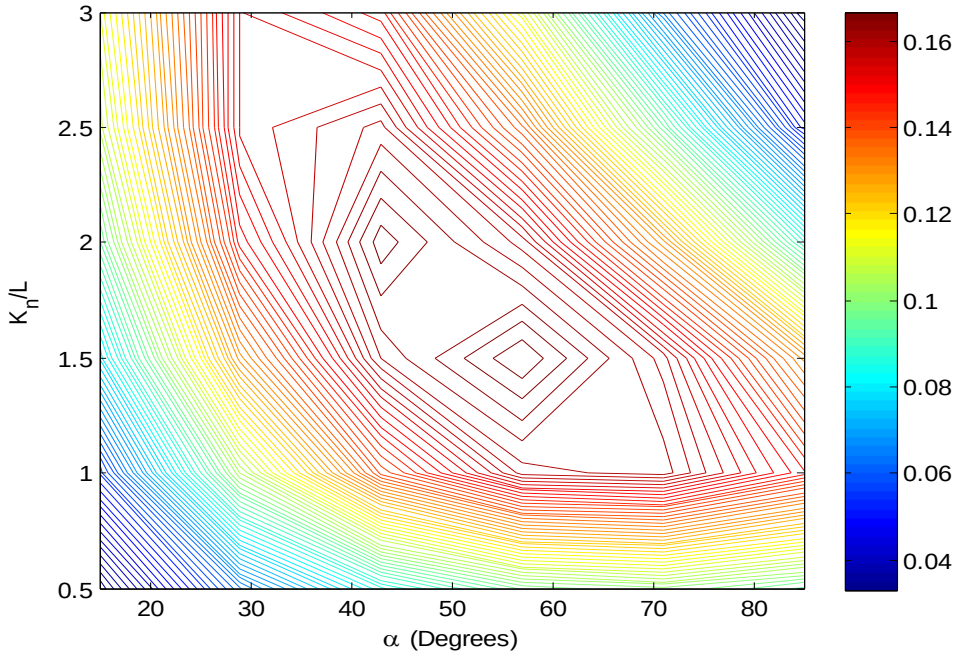


Figure 7.9: Contour plot of performance measure *H* for *v* =0.20 m/s

Obviously, the pareto frontier between the tracing speed v and α^* is the constant line $\alpha^* = 57^\circ$. When the obtained results are considered, it can be concluded that if the overall locomotion speed is increased by increasing the tracing speed v, K_n^* decreases. Recalling that K_n determines the undulation frequency of the modified serpenoid curve, it quite a meaningful result.

Table 7.4: Parameters used for the 4th series of simulations

	m (kg)	**J (kg.m^2)**	**L (m)**	μ_R	$\omega_x = \omega_y$ (rad/s)	d_ϕ (N.s/rad)	**v (m/s)**
Value	*0.792*	*0.00269*	*1*	*0.035*	*8*	*0.01*	**0.25**

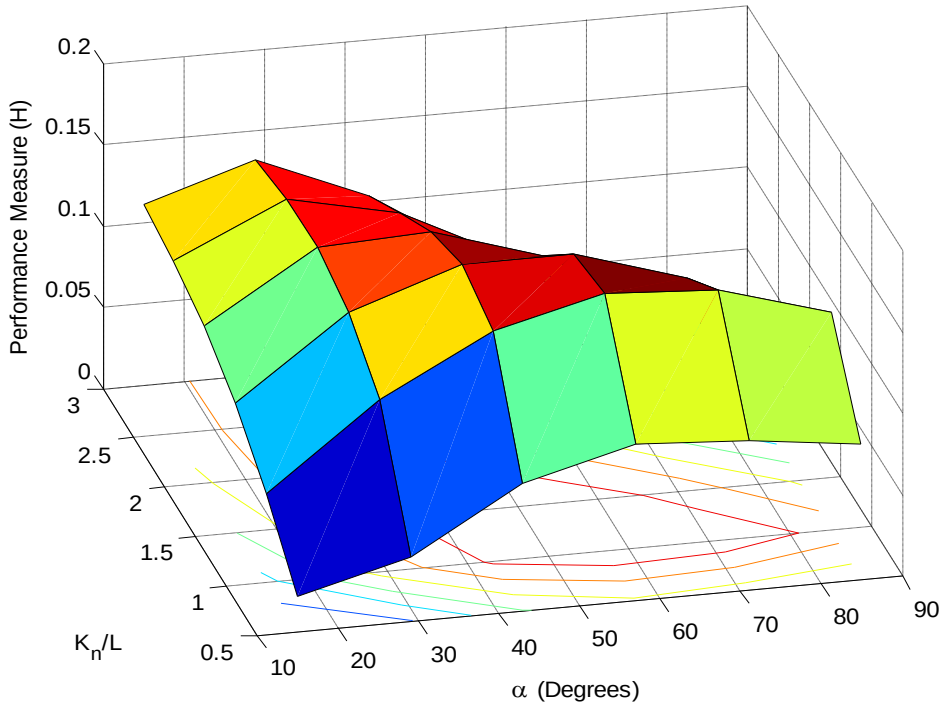


Figure 7.10: Surface of the performance measure H over the region spanned by parameters α and K_n / L for v = 0.25 m/s

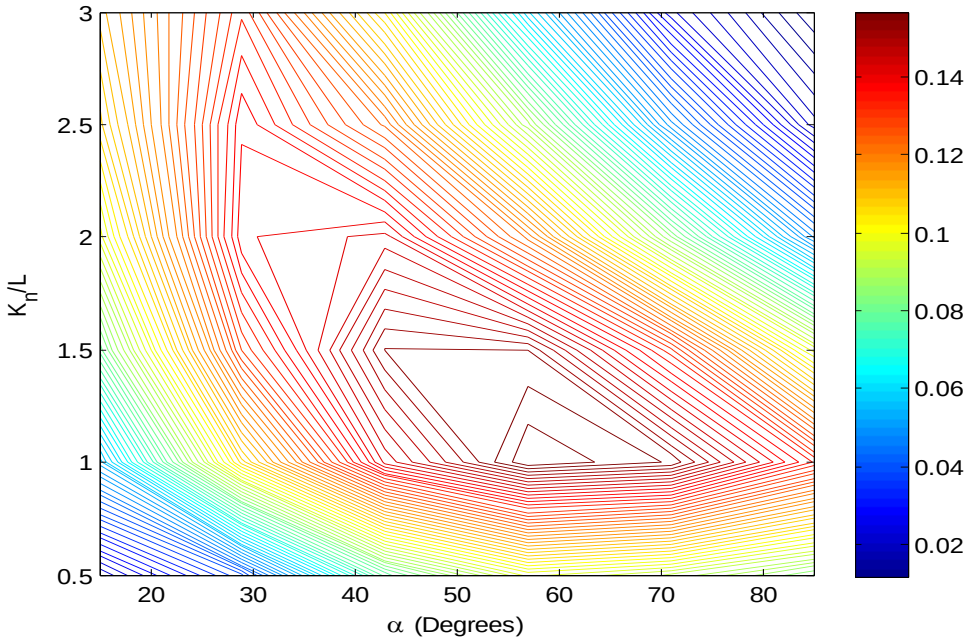


Figure 7.11: Contour plot of performance measure *H* for *v* =0.25 m/s

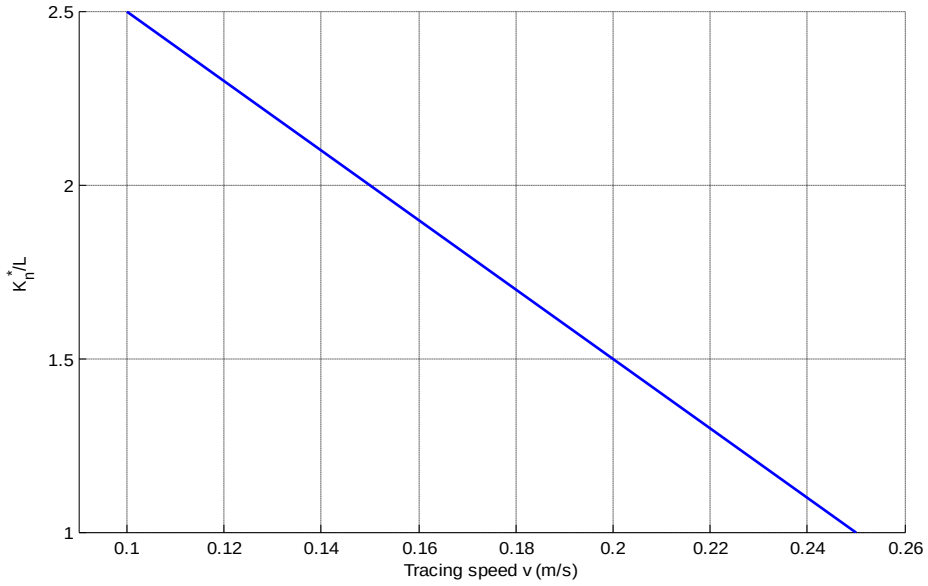


Figure 7.12: Pareto frontier for the performance measure *H* determined by K_n^* / L and tracing speed *v*

That can be interpreted in the following way: to compensate for the increased locomotion speed, the snake robot decreases its undulation frequency to locomote in an optimally efficient manner. What is interesting is that α^* is not affected by the locomotion speed.

7.4 Effect of Eigenvalues and the Friction Coefficient

The effect of eigenvalues (or the natural frequencies ω_x and ω_y) and the tangential friction coefficient μ_R on the optimal parameters K_n^* and α^* .will also be investigated. First tracing speed *v* is set to 0.15 m/s and the parameters are set as given in Table 7.5 where the natural frequencies are decreased from 8 rad/s to 4 rad/s. Then, using the same tracing speed, the natural frequencies are increased to 16 rad/s this time, i.e. the parameters given in Table 7.6 are used.

Results of the fifth set of simulations are summarized by Figure 7.13 and Figure 7.14, while the results of the sixth one are summarized by Figure 7.15 and Figure 7.16. When the results of the second set of simulation and the fifth one are compared with each other, it is seen that by decreasing the natural frequencies of the linear controller form 8 rad/s to 4 rad/s, α^* decreased from 57° to 43°, while K_n^* stayed at the same value of 2*L*.

Table 7.5: Parameters used for the 5th series of simulations

	m (kg)	**J (kg.m^2)**	**L (m)**	μ_R	$\omega_x = \omega_y$ (rad/s)	d_ϕ (N.s/rad)	**v (m/s)**
Value	*0.792*	*0.00269*	*1*	*0.035*	**4**	*0.01*	**0.15**

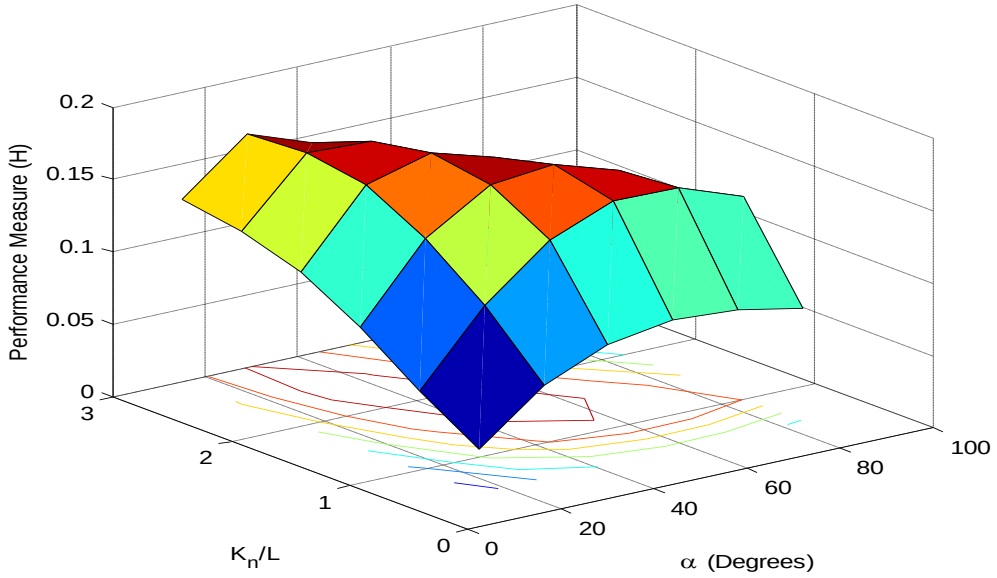


Figure 7.13: Surface of the performance measure *H* over the region spanned by parameters α and K_n / L for v = 0.15 m/s and $\omega_x = \omega_y = 4\ rad/s$

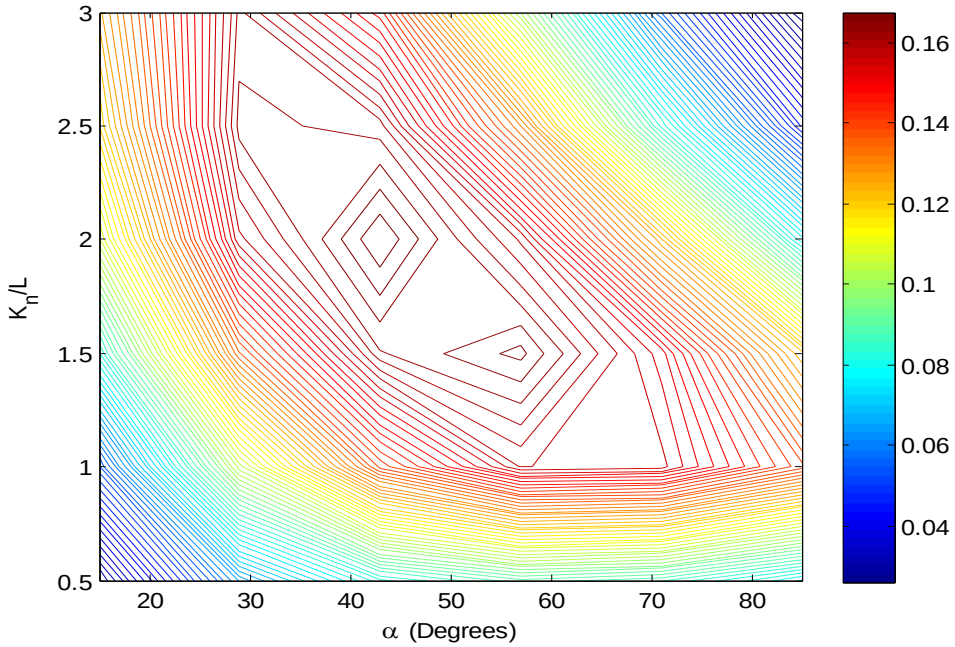


Figure 7.14: Contour plot of performance measure *H* for v =0.15 m/s and $\omega_x = \omega_y = 4\ rad/s$

It is known that eigenvalues of the linear controller and hence the natural frequencies ω_x and ω_y are directly related with the tracking performance of the robot head. It seems that reducing them to 4 rad/s degrades the tracking performance considerably and to reduce these tracking errors (deviation from the modified serpenoid curve) the robot chooses to locomote with a lower initial winding angle α. Thus, decreasing the deviation from the modified serpenoid curve leads to a more efficient locomotion. This can be easily confirmed by comparing Figure 7.7 and Figure 7.14, where at $\omega_x = \omega_y = 8\ rad/s$, H has a maximum value of 0.1880 and at $\omega_x = \omega_y = 4\ rad/s$, this value decreased to 0.1685. In the sixth set of simulation, the natural frequencies are simply increased to 16 rad/s. Observing Figure 7.15 and Figure 7.16, it is seen that increasing the natural frequencies from 8 rad/s to 16 rad/s did not change the optimal values $K_n^* = 2L$ and $\alpha^* = 57^\circ$. However, increasing the natural frequencies (and thus further pushing the eigenvalues to the left) increased the maximum value of H to 0.1930 from 0.1880 which can be seen if compare Figure 7.7 and Figure 7.16. Important conclusions can be made simply by analyzing the results of the fifth and the sixth set of simulations. Firstly, it seems that, up from certain values of natural frequencies, the specific form of the serpenoid curve (and hence the gait of the snake robot) for an optimally efficient locomotion does not change. That is because K_n and α sets this specific form of gait and they settle down to certain optimal values once the robot head follows the modified serpenoid curve, *well enough*. In order to further justify this conclusion and to obtain a more complete picture, simulations for the natural frequencies $\omega_x = \omega_y = 12\ rad/s$ and $\omega_x = \omega_y = 20\ rad/s$ have also been performed, without changing the rest of the parameters. The same optimal values $K_n^* = 2L$ and $\alpha^* = 57^\circ$ are obtained for both $\omega_x = \omega_y = 12\ rad/s$ and $\omega_x = \omega_y = 20\ rad/s$, as expected. For that reason, individual simulation results for the mentioned natural frequencies are omitted. However, maximum values of the performance measure H are obtained to be

Table 7.6: Parameters used for the 6th series of simulations

	m (kg)	**J (kg.m^2)**	**L (m)**	μ_R	$\omega_x = \omega_y$ (rad/s)	d_ϕ (N.s/rad)	**v (m/s)**
Value	*0.792*	*0.00269*	*1*	*0.035*	**16**	*0.01*	**0.15**

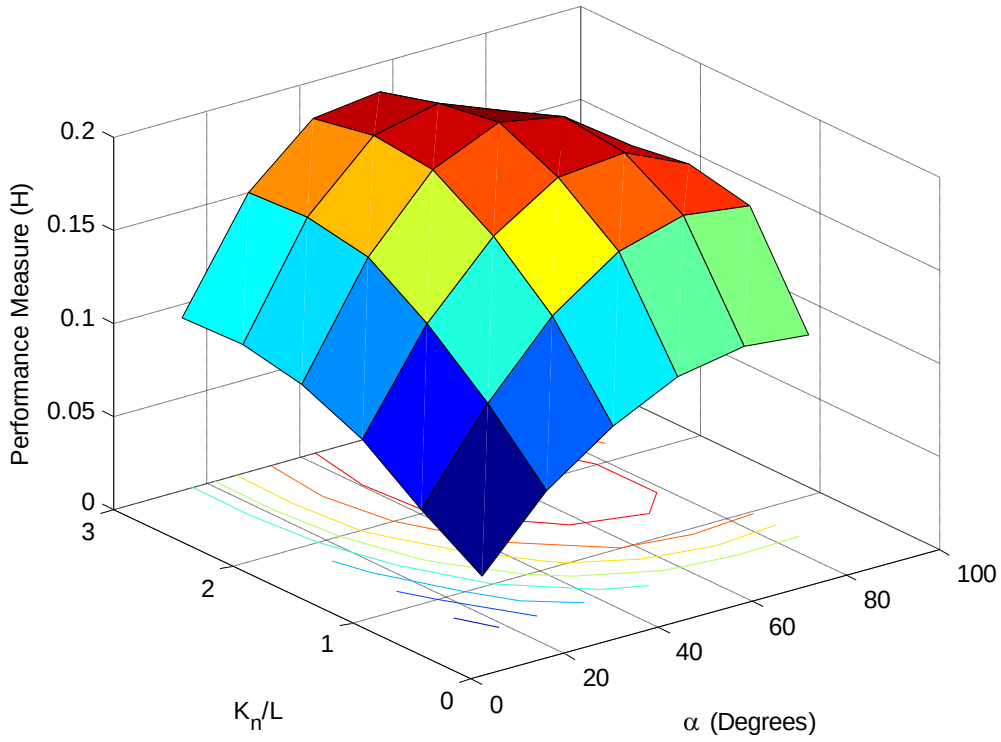


Figure 7.15: Surface of the performance measure *H* over the region spanned by parameters α and K_n / L for v = 0.15 m/s and $\omega_x = \omega_y = 16\ rad/s$

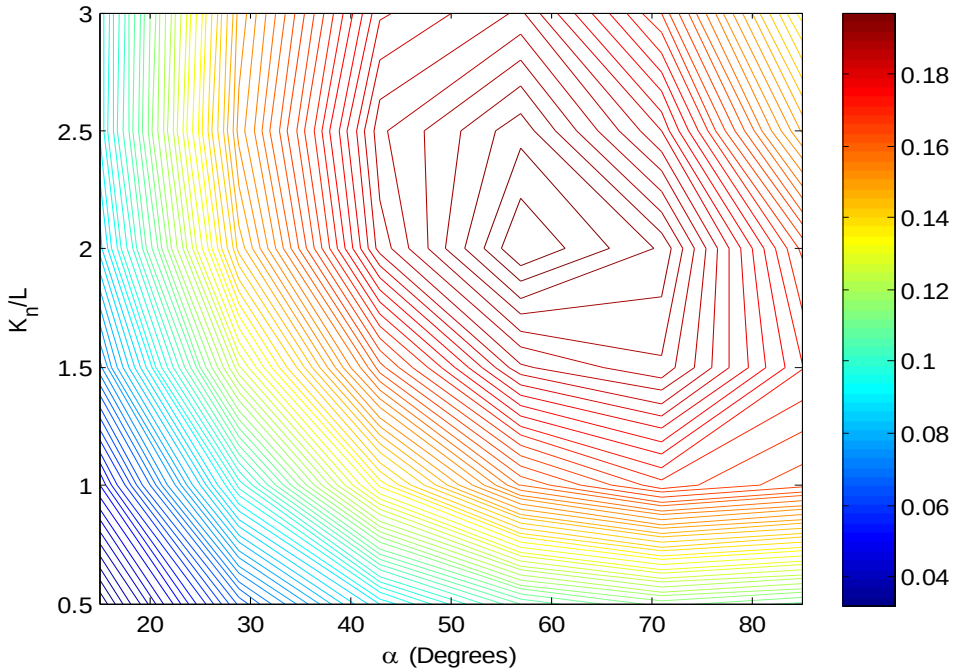


Figure 7.16: Contour plot of performance measure *H* for *v* =0.15 m/s and $\omega_x = \omega_y = 16\ rad/s$

0.1910 and 0.1945 respectively. How the locomotion performance (maximum value of the performance measure *H*) varies with eigenvalues (i.e., natural frequencies $\omega_x = \omega_y$) of the linear controller is given in Figure 7.17. Another valuable conclusion that can be made from these simulations is that: the fact that increasing the natural frequencies and making the robot track our modified serpenoid curve more closely makes the robot locomote more efficiently confirms and interconnects the followings:

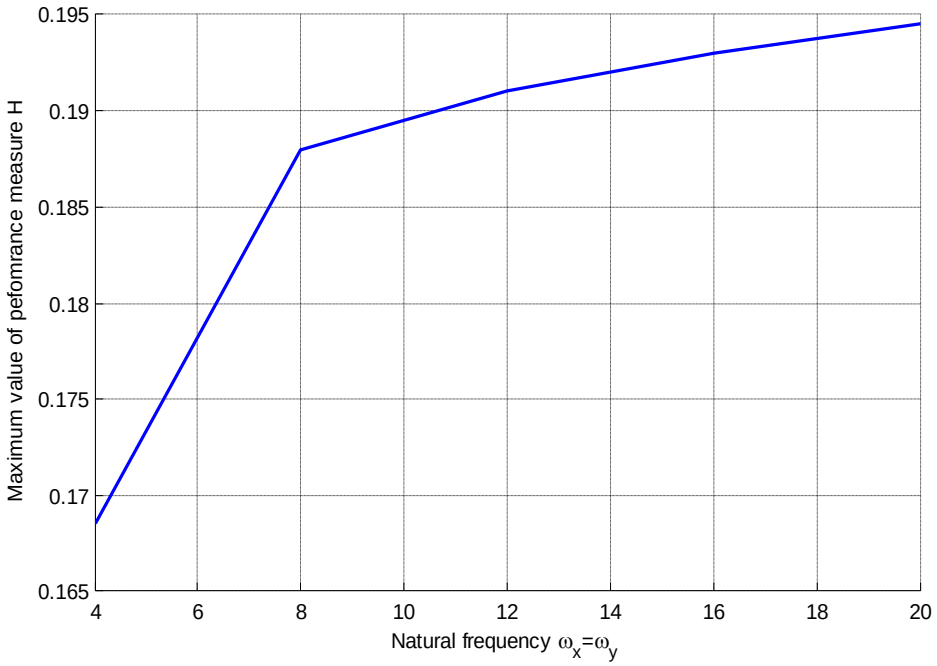


Figure 7.17: Variation of the locomotion performance with natural frequencies of the linear controller

- Validity of our proposed modified serpenoid curve to generate a smooth and efficient locomotion pattern for the modular snake robots.

- Validity and consistency of our proposed performance measure *H* to promote what is good and to demote what is bad in terms of efficient locomotion.

That is because, the more closely the modified serpenoid curve is tracked the higher the performance measure gets. Actually this is something quite unusual for regular control systems. In classical control systems, tracking a reference signal better implies implementing higher gains and this in turns implies consumption of more resources whether in the form of actuator force, torque or total energy consumption. Finally, in order to observe the effect of tangential friction on the robot locomotion seventh and eighth sets of simulations are performed using the

parameters given in Table 7.7 and Table 7.8, where μ_R is decreased ten times to 0.0035 and increased ten times to 0.35, respectively. Results for the seventh set of simulations are given by Figure 7.18 and Figure 7.19 and those for the eighth are given in Figure 7.20 and Figure 7.21. Observing Figure 7.18 and Figure 7.19 and comparing to the results of the second set of simulations, it is observed that decreasing the tangential friction coefficient decreased the optimal value K_n^* to L from $2L$ while α^* stayed at its usual optimal value of 57°. A decrease in μ_R resulted in a dramatic increase in the maximum value of H from 0.1880 to 1.020, which is an expected results since μ_R is the main cause of energy dissipation during the locomotion. Observing Figure 7.20 and Figure 7.21 and comparing the results again with those of the second set of simulations, increasing μ_R, increased K_n^* to $2.5L$ from L, and α^* to 71° from 57°. As expected, a dramatic decrease in the maximum value of H to 0.0235 from 0.1880 can be observed. Considering the results of these two last sets of simulations, it is noted that a decrease in μ_R allows the robot to locomote by less undulation (as K_n^* decreased to L from $2L$) while an increase in μ_R forces the robot to undulate and to go sideways as much as possible to overcome the friction. Actually, this result correlates with and explains the similar behavior observed also in natural snakes. On muddy and rough grounds they move in a more wavy and oscillatory manner and while on relatively smoother surfaces they perform the locomotion in a more linear way. Since it is known that the manner in which K_n^* and α^* changes with changing μ_R determines the manner in which the gait of the robot changes with μ_R, these results also confirm again the validity and consistency of the selected performance measure H.

Table 7.7: Parameters used for the 7th series of simulations

	m (kg)	**J (kg.m^2)**	**L (m)**	μ_R	$\omega_x = \omega_y$ (rad/s)	d_ϕ (N.s/rad)	**v (m/s)**
Value	*0.792*	*0.00269*	*1*	**0.0035**	*8*	*0.01*	*0.15*

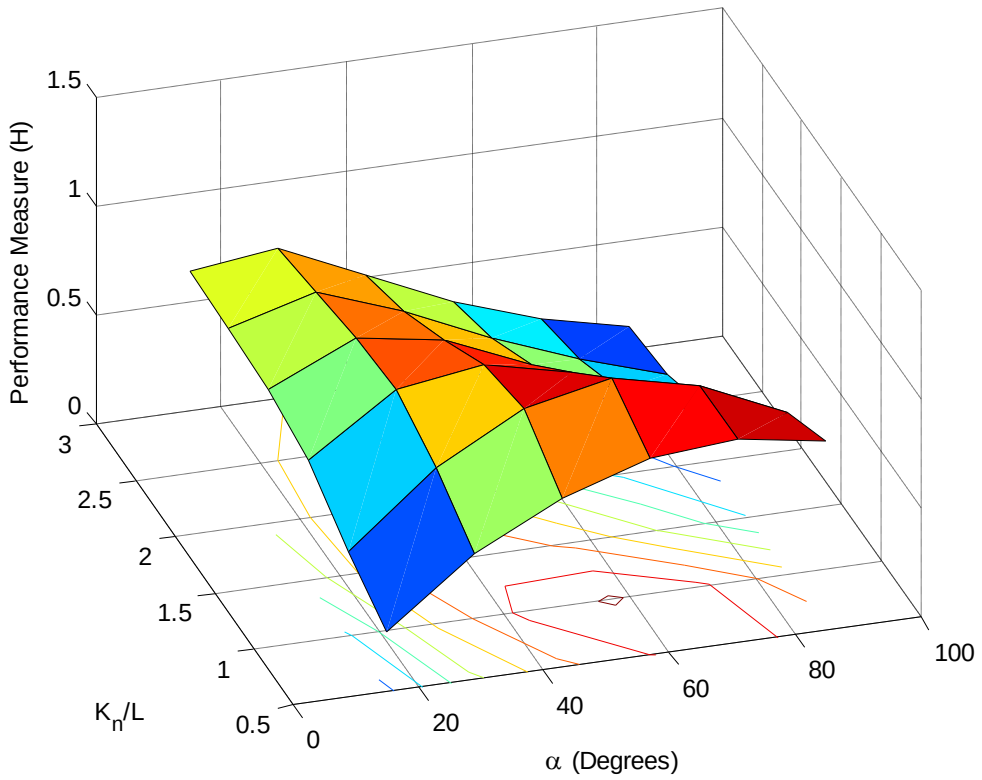


Figure 7.18: Surface of the performance measure H over the region spanned by parameters α and K_n / L for v = 0.15 m/s and $\mu_R = 0.0035$

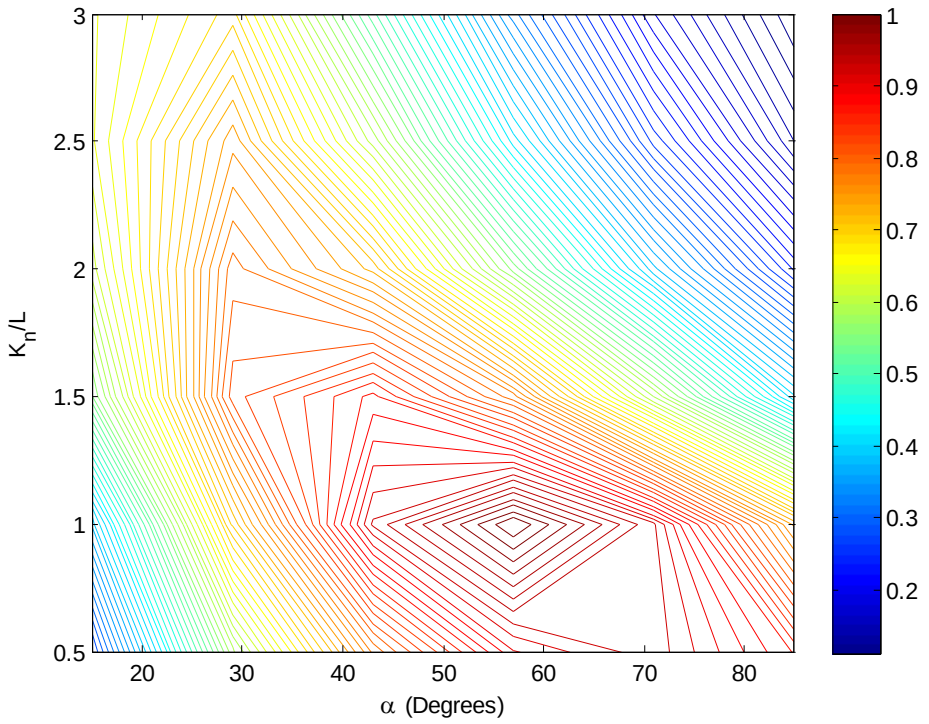


Figure 7.19: Contour plot of performance measure *H* for *v* =0.15 m/s and $\mu_R = 0.0035$

Table 7.8: Parameters used for the 8th series of simulations

	m (kg)	**J (kg.m^2)**	**L (m)**	μ_R	$\omega_x = \omega_y$ (rad/s)	d_ϕ (N.s/rad)	**v (m/s)**
Value	*0.792*	*0.00269*	*1*	**0.35**	*8*	*0.01*	*0.15*

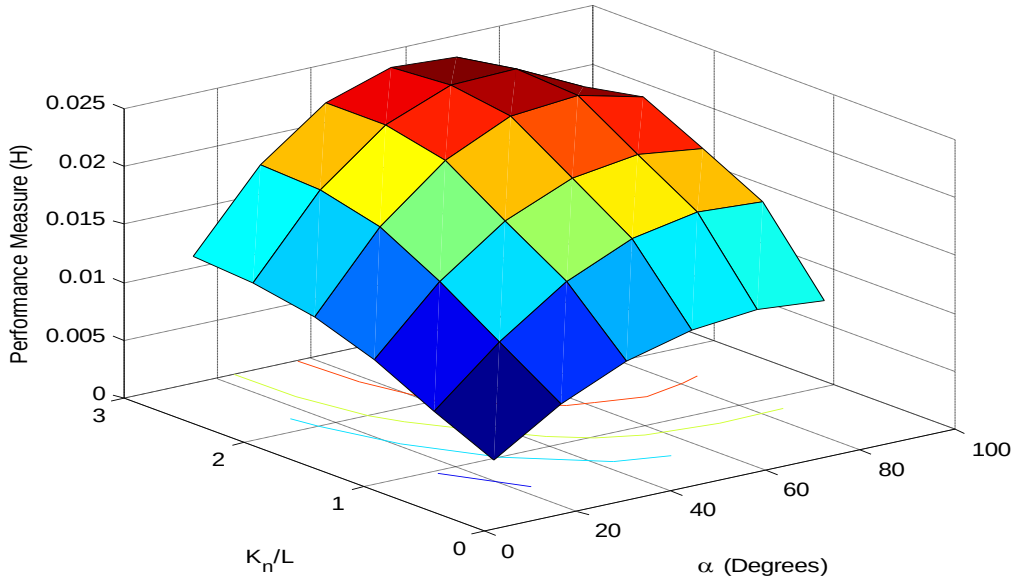


Figure 7.20: Surface of the performance measure *H* over the region spanned by parameters α and K_n / L for v = 0.15 m/s and $\mu_R = 0.35$

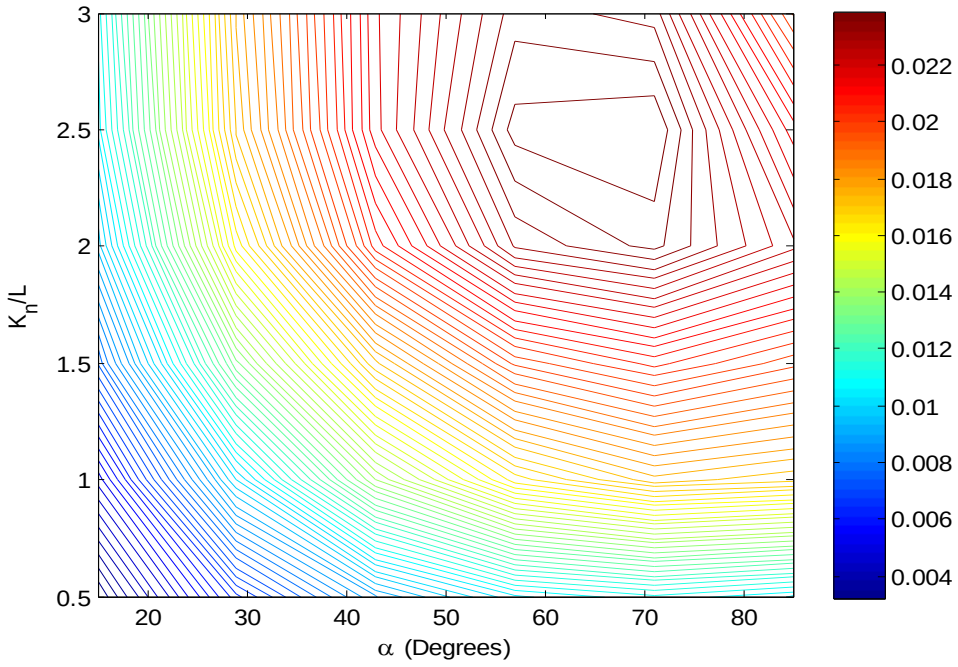


Figure 7.21: Contour plot of performance measure *H* for v =0.15 m/s and $\mu_R = 0.35$

CHAPTER 8

DISCUSSION OF RESULTS AND CONCLUSION

8.1 Discussion of Results

One could argue that since *purely* (without using our modified serpenoid curve) following the given eight-shape-like task trajectory would imply that the condition given by lemma in section 3.3 (which simply states that singularity occurs when all of the joint angles equals to each other in modulus π) will not be met, there would be no need for the modified serpenoid curve when the task trajectory is not strictly a linear one. Although it is true that the singularity condition will not be met in those cases, simply observing Figure 7.6, for example, shows how the performance measure H dramatically degrades when $\alpha = 0$ is approached [8] (although the point $\alpha = 0$ is not explicitly included in the grid). To demonstrate this degradation clearly, a simulation has been performed using the parameters given in Table 7.2 and for $\alpha = 0$ and for 25 seconds. The results of the simulation are given by Figure 8.1, Figure 8.2 and Figure 8.3. From these figures, one can see that although the overall deviation from the task trajectory decreases considerably, the price to be paid for that is the drastic increases in the lateral forces and motor torques applied by the actuators.

[8] As have been stated before, setting $\alpha = 0$ means the modified serpenoid curve is exactly equivalent to the given task trajectory

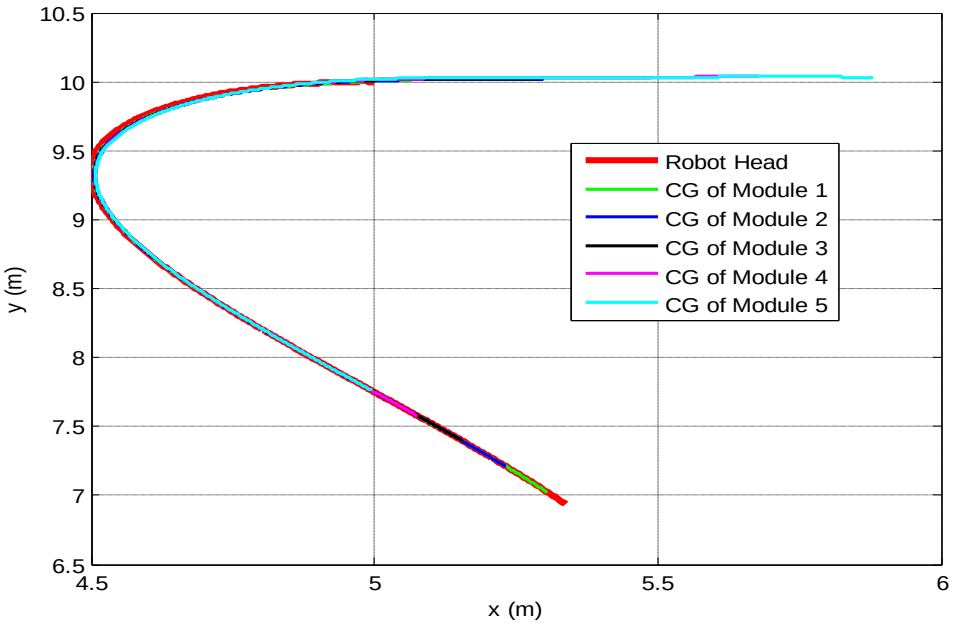


Figure 8.1: Trajectory of the robot head and modules following the reference $r_r(t) = x_r(t)\hat{i} + y_r(t)\hat{j}$

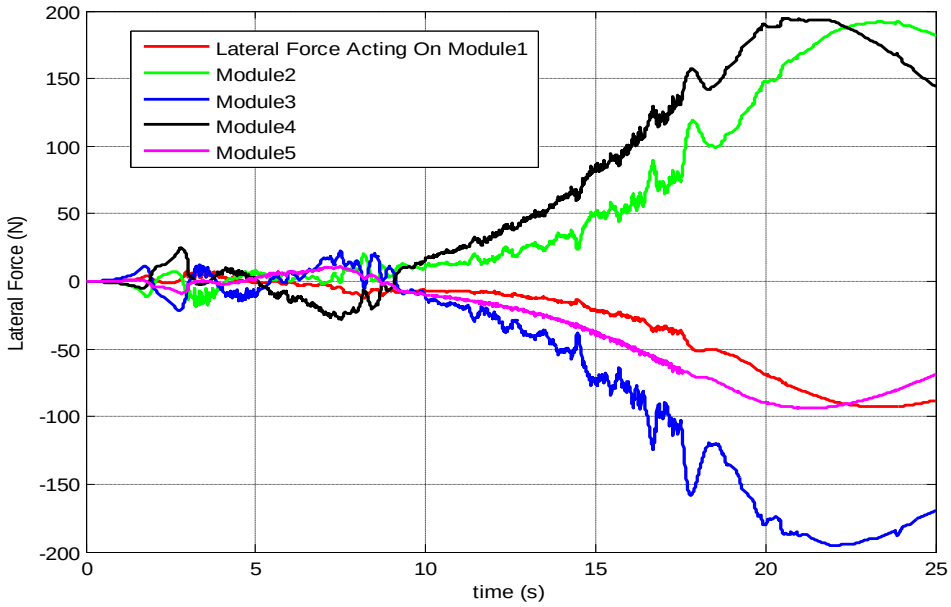


Figure 8.2: Lateral forces $f_1^N, \ldots, f_5^N$ applied by the ground to the modules

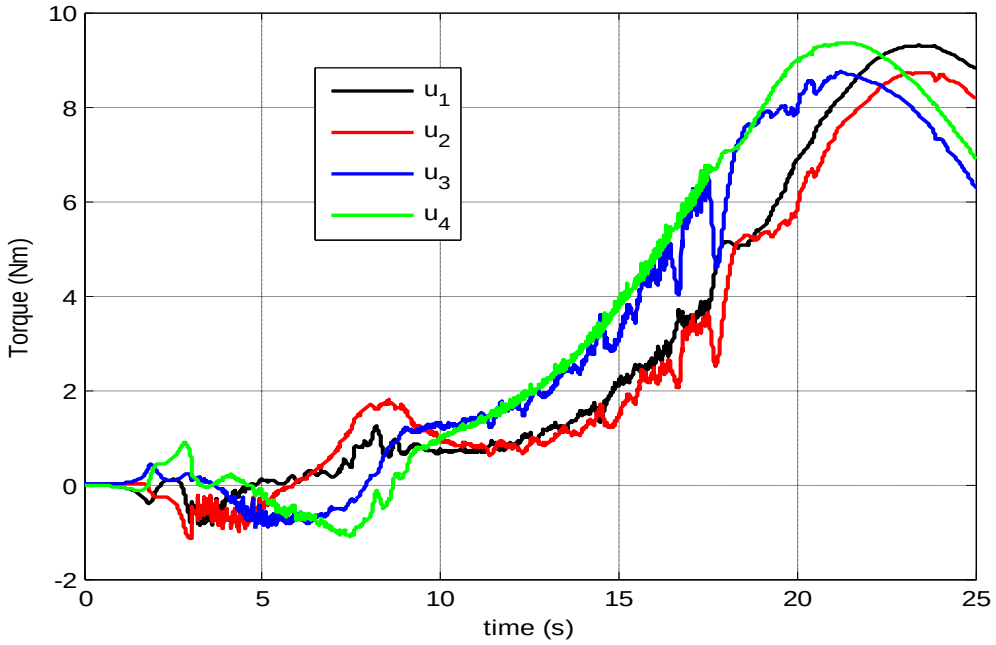


Figure 8.3: Torques $u_1,\ldots,u_4$ applied by the actuators to the robot joints

Noting also from Figure 8.2 and Figure 8.3 that, not only the magnitudes of the lateral forces and the joint toques have increased, but also their profile have become quite oscillatory which will be surely lethal for both the actuators and the modules. This price is impossible to afford since this is neither practical nor possible. This unfavorable locomotion condition is covered quite well by the proposed performance measure *H* such that, it turned out to be about 0.0054 which is about 35 times smaller than its maximum (optimal) value of 0.1880 for the second set of simulations that are run in section 7.3. The total amount of energy consumed, *E*, is 4.77 Joules. If we had not included the *structural term* in the definition of the performance measure *H* given by equation (7.4), this situation would be one of the most favorable ones which have the highest value of effective locomotion distance p_{eff} and low total energy consumption *E*. Assuming, for example, that the normal friction coefficient between the modules and the ground

maximum amount of friction that can be applied by the ground to a single module μ_N is 0.8 and knowing that mass of a single module m = 0.792 kg, then the maximum available lateral force is simply $f_{\max}^N = m\mu_N g = 6.215\ \text{N}$. This means that in order for the no slip condition to hold, the maximum of the lateral forces $[f_1^N\ f_2^N\ f_3^N\ f_4^N\ f_5^N]$ should not exceed $f_{\max}^N = 6.215\ \text{N}$. Observing Figure 8.2, it is seen that 200 N of lateral friction force is needed to prevent the modules from side slipping which is obviously impossible for a mass of 0.792 kg. To compare these results with those of the optimal one, the simulation results of the optimal locomotion (whose performance measure was found to be 0.1880) from the second set of simulation which also uses the same parameters (but with K_n^* and α^*) are given in Figure 8.4, Figure 8.5 and Figure 8.6. When these figures are observed, the striking differences between the optimal locomotion and the locomotion with $\alpha = 0$ can be clearly seen.

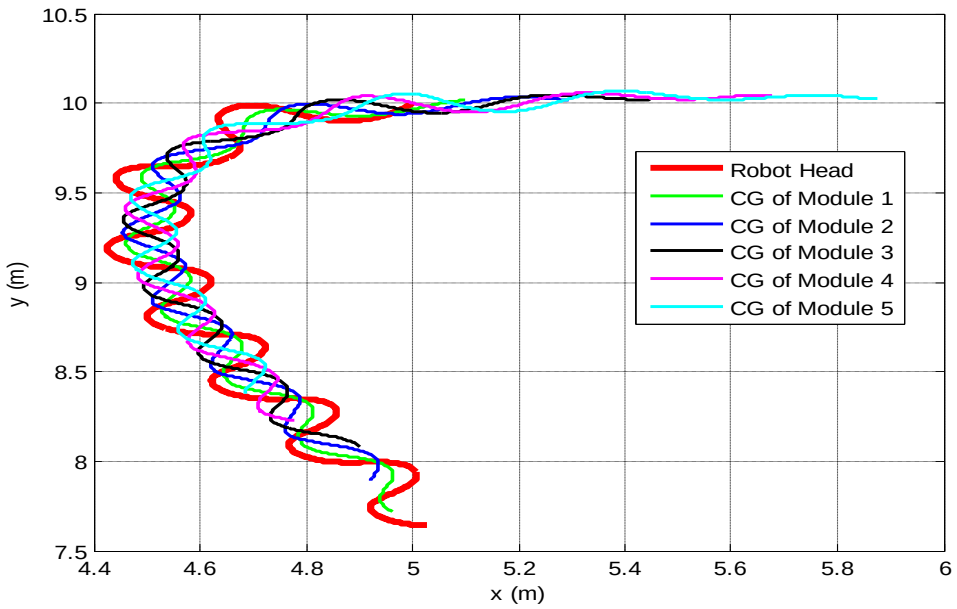


Figure 8.4: Trajectory of the robot head and modules following the reference $r_r(t) = x_r(t)\hat{i} + y_r(t)\hat{j}$

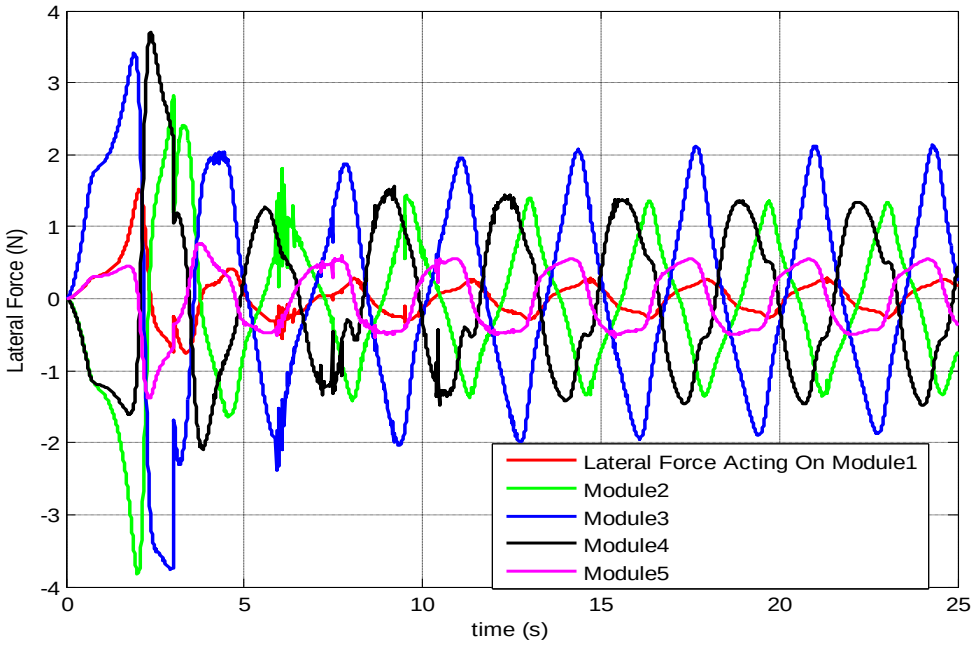


Figure 8.5: Lateral forces $f_1^N,\ldots, f_5^N$ applied by the ground to the modules

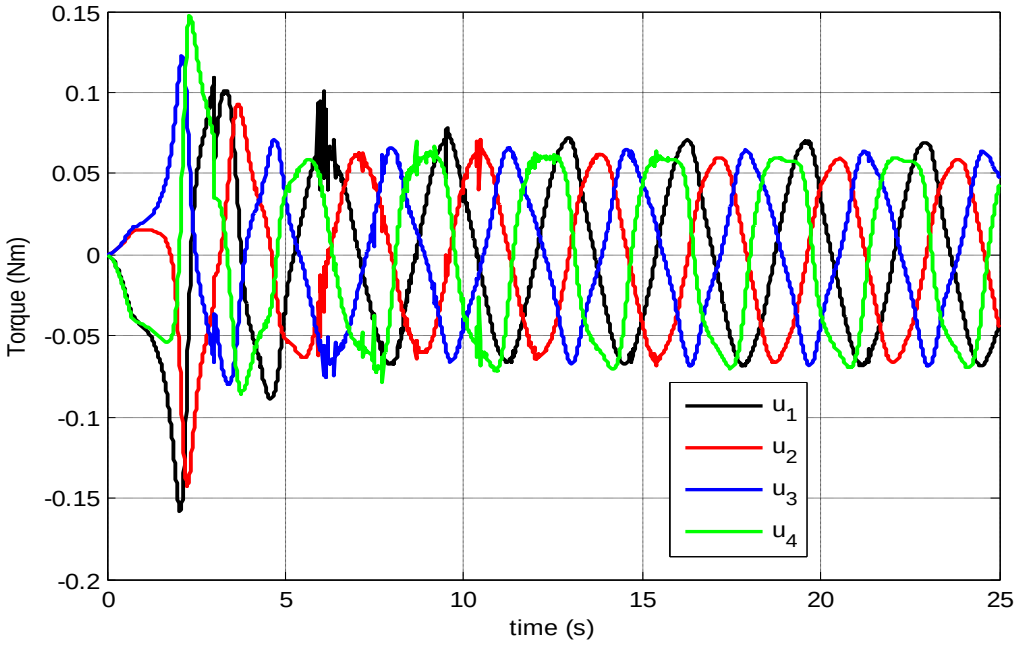


Figure 8.6: Torques $u_1,\ldots,u_4$ applied by the actuators to the robot joints

First, it should be noted that, the optimal one has considerably smoother and lower profiles for lateral forces and actuator torques. Having the maximum amount of necessary lateral force about 3.5 N, a normal friction coefficient of $\mu_R = 0.45$ is enough to prevent the robot from side slipping. However, this time a total of 6.18 Joules of energy is consumed. One cannot even call this a trade-off compared to 4.77 Joules. That is because, locomoting with 4.77 Joules of energy consumption is not possible due to high and oscillatory lateral and actuator forces and even if it is made possible somehow, it will not be sustainable.

One could also argue that, reasoning from the lemma given in section 5.4, if the speed at which the head traces the task trajectory is varied somehow about its nominal value, the singularity condition could also be avoided without utilizing the modified serpenoid curve. However, it should be noted that this lemma is a one sided implication. That is to say, constancy of the robot head velocity does not cover all of the cases which lead to the singularity i.e., the cases where the head velocity is not constant may still lead to the singularity condition. To illustrate what happens when the speed of the head is varied and the modified serpenoid curve is not employed the first simulation performed in section 5.4 will be repeated, with a robot head speed $v = 0.05 + 0.005\sin(t)$, this time. This means the speed is varied 10 % sinusoidally around the nominal value of 0.05 m/s along the task trajectory $r_t(t') = (-t'+5)\hat{i} + 10\,\hat{j}$, where the parameter *t* is explicitly the time and the parameter t' is a generic parameter. The simulation results are given by Figure 8.7, Figure 8.8, Figure 8.9 and Figure 8.10. When they are observed, it can be easily seen that the idea of varying the speed around a nominal value simply does not work as both of the actuator torques and the lateral forces diverge. This result might be attributed to the fact that although the controllers constantly work to attain the speed profile given by Figure 8.7, nothing is being done actually to prevent the joint angles φ_i, $i = 1,\ldots,4$ from converging to zero as depicted in Figure 8.10.

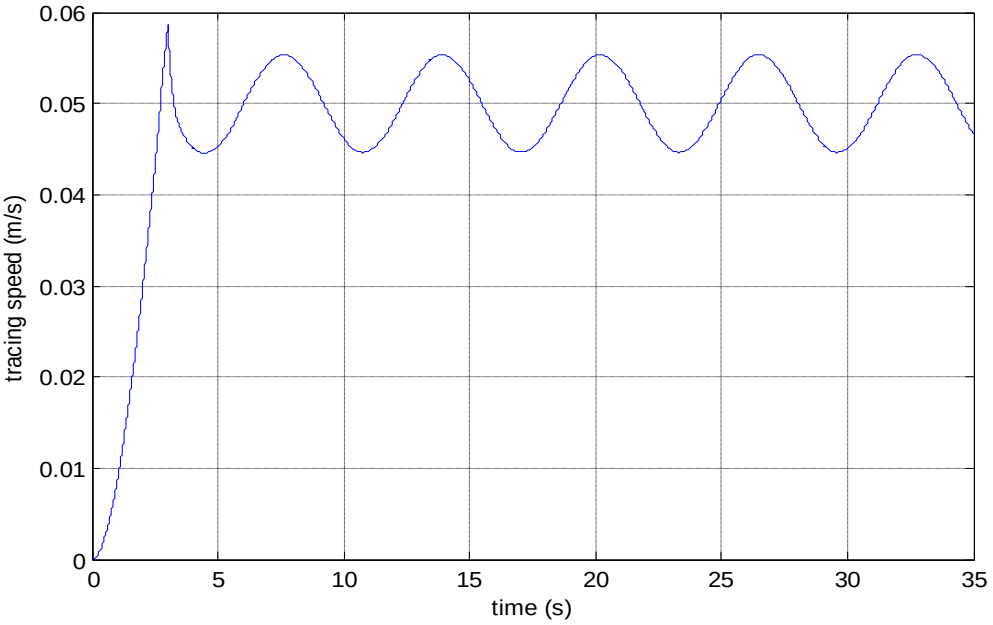


Figure 8.7: Actual speed v of the robot head along the trajectory

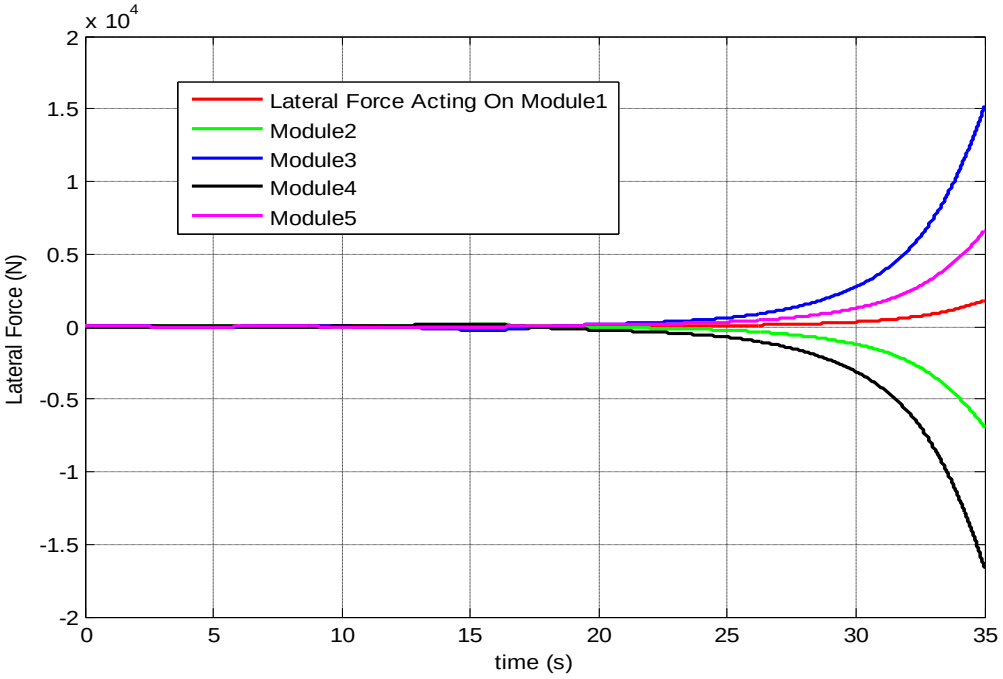


Figure 8.8: Lateral forces $f_1^N, \ldots, f_5^N$ applied by the ground to the modules

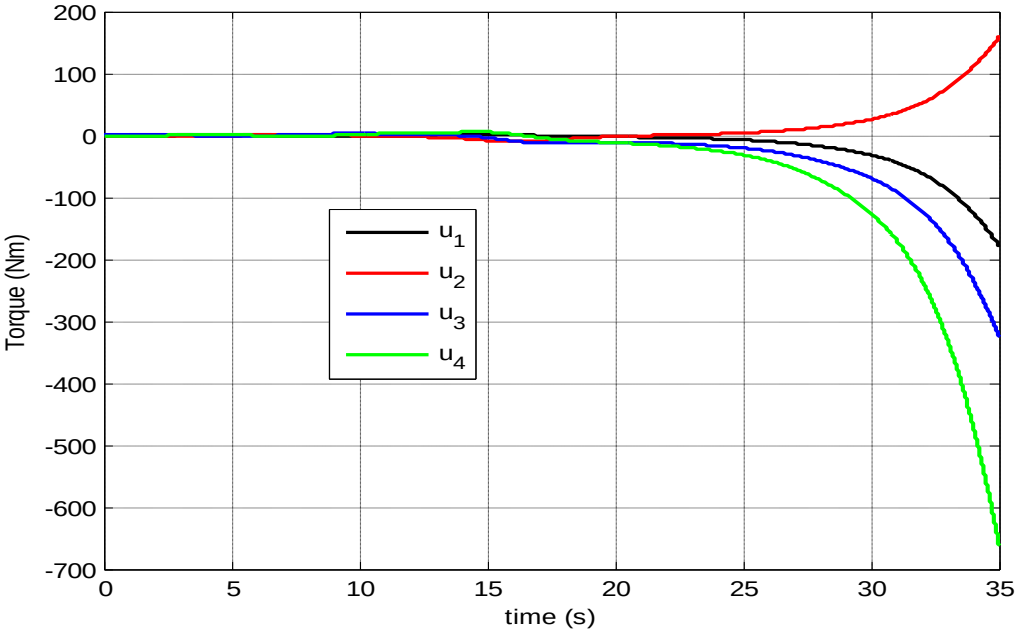


Figure 8.9: Torques $u_1,\ldots,u_4$ applied by the actuators to the robot joints

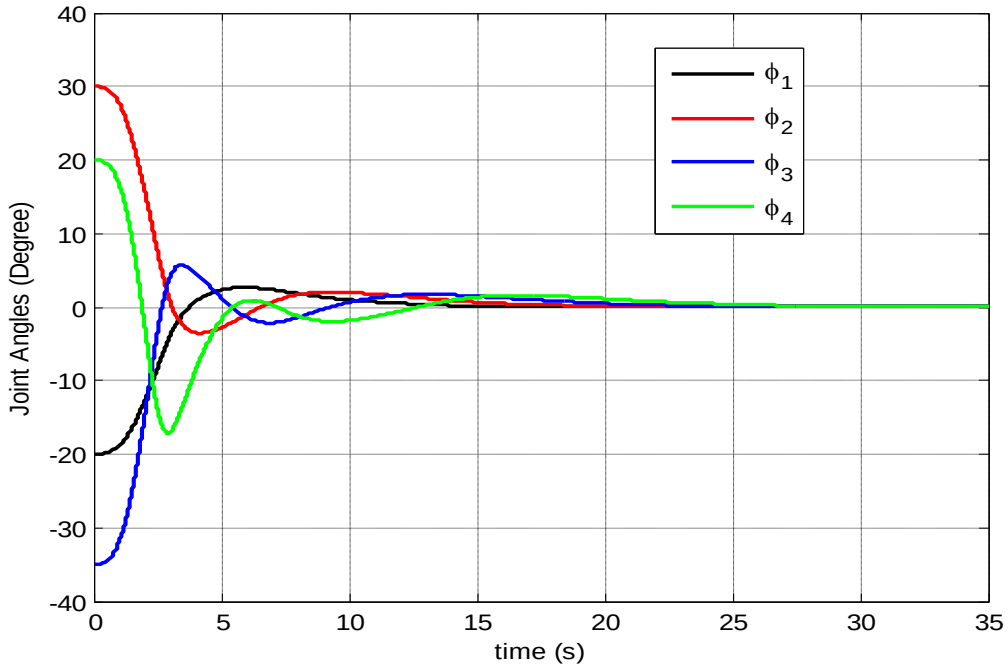


Figure 8.10: Joint angles φ_i, $i=1,\ldots,4$ of the robot during the locomotion

This convergence is due to the fact that regardless of its reference speed, the robot head always tracks the points lying on the given straight line. Consequently, all of the modules align with the given straight line i.e. their respective orientations converge to the same constant value at steady state.

Furthermore, it should be noted that inclusion of initial errors (error between the initial robot head position and the initial reference point) has been avoided as much as possible. However, even if the initial head position error is avoided by setting the initial head position to the initial point of the reference trajectory, transient periods are still observed in the simulation results which are especially visible in the lateral force and the applied joint torques histories. This results from the fact that at the start of the simulation, the controller, implicitly, attempts to align the overall orientation of the robot to the initial curvature of the reference path by modifying the joint angles.

It is also interesting to compare Figure 8.6 with Figure 8.3. As have been stated before, the modified serpenoid curve acts somewhat as a filter which rejects the effects of the reference path (task trajectory) as if it were a disturbance. As a result, regardless of the task trajectory to track, the actuators apply similar profiles of torques. However, if the modified serpenoid curve is not utilized, one can even roughly anticipate the trajectory followed simply by observing the actuator torque profile as in Figure 8.3.

Observing Figure 7.17, it can be clearly seen that, the further the eigenvalues of the linear controller are pushed to the left in the complex plane, the better the modified serpenoid curve is tracked. Eventually, this better tracking results in a higher locomotion performance i.e., a higher value of H is obtained. This result is enough to show that, the proposed modified serpenoid curve is still in alignment with the basic principles of the original serpenoid curve proposed by Hirose: to generate smooth and energy efficient gaits for snake like robots. However, we cannot increase the natural frequencies of the linear controller indefinitely. One

reason for that is the physical limitation of the real implementation of the gains. The other reason is the nature of the feedback linearization method. Controllers designed by feedback linearization method are generally more suitable for trajectory tracking problems than the regulating ones, as high gains may lead to oscillatory transient periods in the response of the system to initial errors such that, even some moderate amount of initial position error in the robot head may generate large lateral forces and actuators torques. Also, during the long periods of locomotion of the robot, there may be deviations from the reference path due to unexpected reasons such as collision with an obstacle. As a result, the controllers implemented with high gains may command the actuators to apply large joint torques (which automatically implies large lateral forces) in order to set the robot head to its path again by trying to reduce these perturbations.

On the other hand, one could consider driving at least one pair of the passive wheels attached to the modules. In such a case, all of the singularity conditions could be eliminated by simply driving the actuated wheel during and/or around the singularity region. This actuation will disrupt the singularity formation by forcing the module attached with the active wheels forward (or backward). However, driving one pair of wheels dramatically changes the governing dynamics. On that account, the question of when and how to drive this pair of wheels, in combination with the joint actuators, in order to obtain a smooth, energy efficient and singularity free locomotion with an ability to track arbitrary trajectories seems to be rather challenging. This topic can be the subject of another study.

8.2 Conclusion

To conclude, the goals set at the beginning of this study have been reached successfully. In other words, by introducing the modified serpenoid curve, the robot is enabled to track any given task trajectory smoothly and exactly (at the macro scale). At the same time, the singularity has been avoided automatically due to the nature of serpenoid curve. While accomplishing these, the locomotion is

maintained to be energy efficient and sustainable by introducing a rational performance measure. Based on this performance measure, several locomotion patterns are evaluated to obtain optimal parameters leading to an efficient locomotion. Furthermore, the relationship between the optimal modified serpenoid curve parameters and the locomotion speed has been obtained. On the other hand, effect of eigenvalues and the coefficient of friction in the tangential direction on the locomotion performance has been illustrated. The major improvements and contributions introduced in thesis can be listed as follows

1) The tangential friction forces have been introduced into the equations of motion of the robot and the affect of these forces on the locomotion performance has been demonstrated.

2) The feedback linearization method has been applied directly to control the robot head.

3) A modified version of the serpenoid curve has been proposed in order to make the robot locomote along any feasible arbitrarily defined trajectory.

4) A rational performance measure has been proposed which takes into consideration both efficiency and sustainability.

By performing several simulations, the validity, consistency and the rationality of the proposed performance measure has been demonstrated.

Notably, due to their universal natures, the proposed novel modified serpenoid curve and the performance measure can be readily and successfully applied to a wide range of snake robots to generate locomotion and evaluate it, respectively.

8.3 Future Work

In the future works, the following issues can be elaborated:

- Eigenvalues of the linear controller can be designed in an adaptive manner so as to obtain milder responses for the actuator torques and lateral forces when initial condition errors are present.

- Some sophisticated optimization methods can be applied in order to obtain the optimal locomotion parameters online.

- Constants α_1 and α_2 in the performance measure H can be determined in such a manner that the resultant optimal locomotion is performed at the maximum attainable tracing speed v such that the robot modules are on the merge of side slipping. This can be done using neural networks, online.

- All of the design parameters like $\omega_x = \omega_y$, K_n^* and α^* can be set adaptively during the locomotion when the environment changes (friction coefficients, damping constants).

- A detailed study could be made on how and when to drive at least one pair of the wheels in order to avoid singularity.

- The no side slip condition could be relaxed in order to determine the locomotion patterns and performance exhibited when the modules slip sideways.

REFERENCES

[1] D. Hu, J. Nirody, T. Scott, and M. Shelley, "The mechanics of slithering locomotion," *Proceedings of the National Academy of Sciences*, vol. 106, no. 25, p. 10081, 2009.

[2] M. Walton, B. Jayne, and A. Bennet, "The energetic cost of limbless locomotion," *Science*, vol. 249, no. 4968, p. 524, 1990.

[3] A. Bellairs, *The life of reptiles*. Universe Books, 1970.

[4] S. Hirose and M. Mori, "Biologically inspired snake-like robots," in *IEEE International Conference on Robotics and Biomimetics, ROBIO*, 2004, pp. 1–7.

[5] "Integrated Taxonomic Information System," *Serpentes (TSN 174118)*, 2010.

[6] K. Dowling, "Limbless locomotion: Learning to crawl with a snake robot," Ph.D. dissertation, Carnegie Mellon University, 1996.

[7] J. Gray, "The mechanism of locomotion in snakes," *Journal of Experimental Biology*, vol. 23, no. 2, p. 101, 1946.

[8] J. Hopkins, B. Spranklin, and S. Gupta, "A survey of snake-inspired robot designs." *Bioinspiration & biomimetics*, vol. 4, no. 2, p. 21001, 2009.

[9] D. Cundall, "Functional morphology," *Snakes: ecology and evolutionary biology*, pp. 106–140, 1987.

[10] J. Gray and H. Lissmann, "The kinetics of locomotion of the grass-snake," *Journal of Experimental Biology*, vol. 26, no. 4, p. 354, 1950.

[11] S. Gray, "Animal locomotion," pp. 166–193, 1968.

[12] C. Mattison, *The encyclopedia of snakes*. Blandford, 1995.

[13] W. Mosauer, "A note on the sidewinding locomotion of snakes," *American Naturalist*, vol. 64, no. 691, pp. 179–183, 1930.

[14] A. Wolf, H. Choset, B. Brown, and R. Casciola, "Design and control of a mobile hyper-redundant urban search and rescue robot," *Advanced Robotics*, vol. 19, no. 3, pp. 221–248, 2005.

[15] H. Yamada, S. Chigisaki, M. Mori, K. Takita, K. Ogami, and S. Hirose, "Development of amphibious snake-like robot ACM-R5," in *International symposium on robotics*, vol. 36, 2005, p. 133.

[16] A. Transeth and K. Pettersen, “Developments in snake robot modeling and locomotion,” in *Proc. IEEE Int. Conf. Control, Automation, Robotics and Vision*, 2006, pp. 1393–1400.

[17] S. Hirose, *Biologically Inspired Robots: Serpentile Locomotors and Manipulators*. Oxford University Press, 1993.

[18] A. Transeth, K. Pettersen, and P. Liljeb\"ack, “A survey on snake robot modeling and locomotion,” *Robotica*, vol. 27, no. 07, pp. 999–1015, 2009.

[19] J. Ostrowski and J. Burdick, “Gait kinematics for a serpentine robot,” in *IEEE International Conference on Robotics and Automation (ICRA 1996)*, 1996, pp. 1294–1299.

[20] P. Wiriyacharoensonthorn and S. Laowattana, “Analysis and design of a multi-link mobile robot (serpentine),” in *Proc. IEEE Int. Conf. Ind. Technol*, 2002, pp. 694–699.

[21] P. Krishnaprasad and D. Tsakiris, “G-snakes: Nonholonomic kinematic chains on Lie groups,” 1994.

[22] M. Spong, S. Hutchinson, and M. Vidyasagar, *Robot modeling and control*. Wiley New Jersey, 2006.

[23] M. Sato, M. Fukaya, and T. Iwasaki, “Serpentine locomotion with robotic snakes,” *IEEE Control Systems Magazine*, vol. 22, no. 1, pp. 64–81, 2002.

[24] P. Prautsch and T. Mita, “Control and analysis of the gait of snake robots,” in *Proceedings of the 1999 IEEE International Conference on Control Applications*, vol. 1, 1999.

[25] H. Date, Y. Hoshi, and M. Sampei, “Locomotion control of a snake-like robot based on dynamic manipulability,” in *Proceedings of IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS 2000)*, vol. 3, 2000, pp. 2236–2241.

[26] H. Sato, M. Tanaka, and F. Matsuno, “Trajectory Tracking Control of Snake Robots Based on Dynamic Model,” *Transactions*, vol. 42, no. 6, pp. 651–658, 2006.

[27] I. Grabec, “Control of a creeping snake-like robot,” in *Advanced Motion Control, 2002. 7th International Workshop on*. IEEE, 2002, pp. 526–531.

[28] O. Egeland and J. Gravdahl, *Modeling and simulation for automatic control*. Marine Cybernetics, 2002.

[29] G. Miller, “Snake robots for search and rescue,” *Neurotechnology for Biomimetic Robots*, pp. 271–284, 2002.

[30] D. Tsakiris, M. Sfakiotakis, A. Menciassi, G. La Spina, and P. Dario, “Polychaete-like undulatory robotic locomotion,” in *Robotics and Automation, 2005. ICRA 2005. Proceedings of the 2005 IEEE International Conference on*. IEEE, 2005, pp. 3018–3023.

[31] A. Ijspeert and A. Crespi, “Online trajectory generation in an amphibious snake robot using a lamprey-like central pattern generator model,” in *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA 2007)*. Citeseer, 2007.

[32] J. Ayers, J. Davis, and A. Rudolph, *Neurotechnology for biomimetic robots*. The MIT Press, 2002.

[33] Z. Lu, S. Ma, B. Li, and Y. Wang, “Serpentine locomotion of a snake-like robot controlled by cyclic inhibitory CPG model,” in *Intelligent Robots and Systems, 2005.(IROS 2005). 2005 IEEE/RSJ International Conference on*. IEEE, 2005, pp. 96–101.

[34] M. Sfakiotakis, D. Tsakiris, and A. Vlaikidis, “Biomimetic centering for undulatory robots,” in *Biomedical Robotics and Biomechatronics, 2006. BioRob 2006. The First IEEE/RAS-EMBS International Conference on*. IEEE, 2006, pp. 744–749.

[35] J. Conradt and P. Varshavskaya, “Distributed central pattern generator control for a serpentine robot,” in *International Conference on Artificial Neural Networks (ICANN 2003)*. Citeseer, 2003.

[36] A. Crespi and A. Ijspeert, “AmphiBot II: An amphibious snake robot that crawls and swims using a central pattern generator,” in *Proceedings of the 9th international conference on climbing and walking robots (CLAWAR 2006)*. Citeseer, 2006, pp. 19–27.

[37] J. Borenstein, G. Granosik, and M. Hansen, “The omnitread serpentine robot–design and field performance,” in *Proc. SPIE Defense and Security Conference: Unmanned Ground Vehicle Technology VII*. Citeseer, 2005.

[38] H. Zhang, W. Wang, Z. Deng, G. Zong, and J. Zhang, “A novel reconfigurable robot for urban search and rescue,” *International Journal of Advanced Robotic Systems*, vol. 3, no. 4, pp. 359–366, 2006.

[39] H. Ohno and S. Hirose, “Study on slime robot (proposal of slime robot and design of slim slime robot),” in *Intelligent Robots and Systems, 2000.(IROS 2000). Proceedings. 2000 IEEE/RSJ International Conference on*, vol. 3. IEEE, 2002, pp. 2218–2223.

[40] H. Date, Y. Hoshi, M. Sampei, and S. Nakaura, “Locomotion control of a snake robot with constraint force attenuation,” in *Proceedings of the American Control Conference*, vol. 1, 2001, pp. 113–118.

[41] A. Crespi, A. Badertscher, A. Guignard, and A. Ijspeert, "AmphiBot I: an amphibious snake-like robot," *Robotics and Autonomous Systems*, vol. 50, no. 4, pp. 163–175, 2005.

[42] J. Ostrowski and J. Burdick, "The geometric mechanics of undulatory robotic locomotion," *The international journal of robotics research*, vol. 17, no. 7, p. 683, 1998.

[43] R. Murray, Z. Li, and S. Sastry, *A mathematical introduction to robotic manipulation*. CRC, 1994.

[44] A. De Luca and G. Oriolo, "Modelling and control of nonholonomic mechanical systems," *Kinematics and Dynamics of Multi-Body Systems, CISM Courses and Lectures*, vol. 360, pp. 277–342, 1995.

[45] R. Penrose, "A generalized inverse for matrices," in *Mathematical proceedings of the Cambridge philosophical society*, vol. 51, no. 03. Cambridge University Press, 1955, pp. 406–413.

[46] A. Ben-Israel and T. Greville, *Generalized inverses: Theory and applications*. Springer Verlag, 2003.

[47] H. Sussmann, "A general theorem on local controllability," *SIAM Journal on Control and Optimization*, vol. 25, p. 158, 1987.

[48] H. Sussmann and V. Jurdjevic, "Controllability of nonlinear systems," *Journal of Differential Equations*, vol. 12, no. 1, pp. 95–116, 1972.

[49] H. Nijmeijer and A. Van der Schaft, *Nonlinear Dynamical Control Systems*. Springer, 1990.

[50] A. Bloch, J. Baillieul, P. Crouch, and J. Marsden, *Nonholonomic mechanics and control*. Springer Verlag, 2003.

[51] M. Tsuda, S. Nakaura, and M. Sampei, "Dynamic manipulability of a snake-like robot and its effect for sinus-lifting motion," in *SICE 2004 Annual Conference*, vol. 3, 2004.

[52] M. Yamakita, T. Yamada, and K. Tanaka, "Control of snake like robot for locomotion and manipulation," in *Proc. Int. Symp. on Adaptive Motion of Animals and Machines, WeP-III*, vol. 4, 2003.

[53] B. Torby, *Advanced dynamics for engineers*. Holt Rinehart & Winston, 1984.

[54] M. Spong and M. Vidyasagar, *Robot dynamics and control*. Wiley-India, 2009.

[55] K. Johnson, *Contact Mechanics*. Cambridge University Press, 1987.

[56] G. Modeling, S. Struc, G. Hicks, H. Banks, E. Stitzinger, H. Tran, and D. Zenkov, "Modeling and Control of a Snake-Like Serial Link Structure," 2003.

[57] S. LaValle, *Planning Algorithms*. Cambridge University Press, 2006.

[58] A. Isidori, *Nonlinear Control Systems*. Springer Verlag, 1995.

[59] S. Sastry, *Nonlinear systems: Analysis, Stability, and Control*. Springer Verlag, 1999.

[60] G. Oriolo, A. De Luca, M. Vendittelli *et al.*, "WMR control via dynamic feedback linearization: design, implementation, and experimental validation," *IEEE Transactions on Control Systems Technology*, vol. 10, no. 6, pp. 835–852, 2002.

[61] C. Samson, "Control of chained systems application to path following andtime-varying point-stabilization of mobile robots," *IEEE Transactions on Automatic Control*, vol. 40, no. 1, pp. 64–77, 1995.

[62] A. De Luca, G. Oriolo, and C. Samson, "Feedback control of a nonholonomic car-like robot," *Lecture Notes in Control and Information Sciences*, pp. 171–254, 1998.

[63] R. Murray and S. Sastry, "Nonholonomic motion planning: Steering using sinusoids," *IEEE Transactions on Automatic Control*, vol. 38, no. 5, pp. 700–716, 1993.

[64] S. Sastry and R. Murray, "Steering nonholonomic systems in chained form," *Decision and Control, 1991., Proceedings of the 30th IEEE Conference on*, pp. 1121–1126, 1991.

[65] H. Khalil and J. Grizzle, *Nonlinear Systems*. Prentice Hall, Englewood Cliffs, NJ, 1996.

[66] M. Spong, "Underactuated mechanical systems," *Control Problems in Robotics and Automation*, pp. 135–150, 1998.

[67] A. Megretski and R. Olfati-Saber, "Nonlinear control of underactuated mechanical systems with application to robotics and aerospace vehicles," Ph.D. dissertation, Massachusetts Institute of Technology, 2001.

[68] A. Bloch, M. Reyhanoglu, and N. McClamroch, "Control and stabilization of nonholonomic dynamic systems," *IEEE Transactions on Automatic Control*, vol. 37, no. 11, pp. 1746–1757, 1992.

[69] K. Watanabe, M. Iwase, S. Hatakeyama, and T. Maruyama, "Control strategy for a snake-like robot based on constraint force and verification by experiment," in *IEEE/RSJ International Conference on Intelligent Robots and Systems, 2008. IROS 2008*, 2008, pp. 1618–1623.

[70] A. Krener, “Feedback linearization,” *Mathematical Control Theory*, 1998.

[71] A. Isidori and A. Ruberti, “On the synthesis of linear input-output responses for nonlinear systems,” *Systems & Control Letters*, vol. 4, no. 1, pp. 17–22, 1984.

[72] A. Isidori and A. Krener, “On feedback equivalence of nonlinear systems,” *Systems & Control Letters*, vol. 2, no. 2, pp. 118–121, 1982.

[73] E. Kreund, “The structure of decoupled non-linear systems,” *International Journal of Control*, vol. 21, no. 3, pp. 443–450, 1975.

[74] Y. E. Changlong, M. A. Shugen, L. I. Bin *et al.*, “Turning and side motion of snake-like robot,” in *2004 IEEE International Conference on Robotics and Automation*, 2004, p. 075/5.

[75] M. Osborne and A. Rubinstein, *A course in game theory*. The MIT press, 1994.

[76] F. Warner, *Foundations of differentiable manifolds and Lie groups*. Springer Verlag, 1983.

APPENDIX A

SYSTEM MATRICES AND SOME PRELIMINARIES

A.1 ELEMENTS OF THE MATRIX $F(\theta) \in \mathbb{R}^{5\times 2}$

$$F(1,1) = \sin(\theta_1)/l$$

$$F(1,2) = -\cos(\theta_1)/l$$

$$F(2,1) = [\sin(\theta_2) - 2\cos(\theta_1-\theta_2)\sin(\theta_1)]/l$$

$$F(2,2) = -[\cos(\theta_2) - 2\cos(\theta_1-\theta_2)\cos(\theta_1)]/l$$

$$F(3,1) = [\sin(\theta_3) - 2\cos(\theta_1-\theta_3)\sin(\theta_1) - 2\cos(\theta_2-\theta_3)\sin(\theta_2) + 4\cos(\theta_1-\theta_2)\cos(\theta_2-\theta_3)\sin(\theta_1)]/l$$

$$F(3,2) = -[\cos(\theta_3) - 2\cos(\theta_1-\theta_3)\cos(\theta_1) - 2\cos(\theta_2-\theta_3)\cos(\theta_2) + 4\cos(\theta_1-\theta_2)\cos(\theta_2-\theta_3)\cos(\theta_1)]/l$$

$$F(4,1) = [\sin(\theta_4) - 2\cos(\theta_1-\theta_4)\sin(\theta_1) - 2\cos(\theta_2-\theta_4)\sin(\theta_2) - 2\cos(\theta_3-\theta_4)\sin(\theta_3) + 4\cos(\theta_1-\theta_2)\cos(\theta_2-\theta_4)\sin(\theta_1) + 4\cos(\theta_1-\theta_3)\cos(\theta_3-\theta_4)\sin(\theta_1) + 4\cos(\theta_2-\theta_3)\cos(\theta_3-\theta_4)\sin(\theta_2) - 8\cos(\theta_1-\theta_2)\cos(\theta_2-\theta_3)\cos(\theta_3-\theta_4)\sin(\theta_1)]/l$$

$$F(4,2) = -[\cos(\theta_4) - 2\cos(\theta_1-\theta_4)\cos(\theta_1) - 2\cos(\theta_2-\theta_4)\cos(\theta_2) - 2\cos(\theta_3-\theta_4)\cos(\theta_3) + 4\cos(\theta_1-\theta_2)\cos(\theta_2-\theta_4)\cos(\theta_1) + 4\cos(\theta_1-\theta_3)\cos(\theta_3-\theta_4)\cos(\theta_1) + 4\cos(\theta_2-\theta_3)\cos(\theta_3-\theta_4)\cos(\theta_2) - 8\cos(\theta_1-\theta_2)\cos(\theta_2-\theta_3)\cos(\theta_3-\theta_4)\cos(\theta_1)]/l$$

$$F(5,1) = [\sin(\theta_5) - 2\cos(\theta_1 - \theta_5)\sin(\theta_1) - 2\cos(\theta_2 - \theta_5)\sin(\theta_2) - 2\cos(\theta_3 - \theta_5)\sin(\theta_3) -$$
$$2\cos(\theta_4 - \theta_5)\sin(\theta_4) + 4\cos(\theta_1 - \theta_2)\cos(\theta_2 - \theta_5)\sin(\theta_1) + 4\cos(\theta_1 - \theta_3)\cos(\theta_3 - \theta_5)\sin(\theta_1) +$$
$$4\cos(\theta_1 - \theta_4)\cos(\theta_4 - \theta_5)\sin(\theta_1) + 4\cos(\theta_2 - \theta_3)\cos(\theta_3 - \theta_5)\sin(\theta_2) + 4\cos(\theta_2 - \theta_4)\cos(\theta_4 - \theta_5)\sin(\theta_2) +$$
$$4\cos(\theta_3 - \theta_4)\cos(\theta_4 - \theta_5)\sin(\theta_3) - 8\cos(\theta_1 - \theta_2)\cos(\theta_2 - \theta_3)\cos(\theta_3 - \theta_5)\sin(\theta_1) -$$
$$8\cos(\theta_1 - \theta_2)\cos(\theta_2 - \theta_4)\cos(\theta_4 - \theta_5)\sin(\theta_1) - 8\cos(\theta_1 - \theta_3)\cos(\theta_3 - \theta_4)\cos(\theta_4 - \theta_5)\sin(\theta_1) -$$
$$8\cos(\theta_2 - \theta_3)\cos(\theta_3 - \theta_4)\cos(\theta_4 - \theta_5)\sin(\theta_2) + 16\cos(\theta_1 - \theta_2)\cos(\theta_2 - \theta_3)\cos(\theta_3 - \theta_4)\cos(\theta_4 - \theta_5)\sin(\theta_1)] / l$$

$$F(5,2) = -[\cos(\theta_5) - 2\cos(\theta_1 - \theta_5)\sin(\theta_1) - 2\cos(\theta_2 - \theta_5)\cos(\theta_2) - 2\cos(\theta_3 - \theta_5)\cos(\theta_3) -$$
$$2\cos(\theta_4 - \theta_5)\sin(\theta_4) + 4\cos(\theta_1 - \theta_2)\cos(\theta_2 - \theta_5)\cos(\theta_1) + 4\cos(\theta_1 - \theta_3)\cos(\theta_3 - \theta_5)\cos(\theta_1) +$$
$$4\cos(\theta_1 - \theta_4)\cos(\theta_4 - \theta_5)\cos(\theta_1) + 4\cos(\theta_2 - \theta_3)\cos(\theta_3 - \theta_5)\cos(\theta_2) + 4\cos(\theta_2 - \theta_4)\cos(\theta_4 - \theta_5)\cos(\theta_2) +$$
$$4\cos(\theta_3 - \theta_4)\cos(\theta_4 - \theta_5)\cos(\theta_3) - 8\cos(\theta_1 - \theta_2)\cos(\theta_2 - \theta_3)\cos(\theta_3 - \theta_5)\cos(\theta_1) -$$
$$8\cos(\theta_1 - \theta_2)\cos(\theta_2 - \theta_4)\cos(\theta_4 - \theta_5)\cos(\theta_1) - 8\cos(\theta_1 - \theta_3)\cos(\theta_3 - \theta_4)\cos(\theta_4 - \theta_5)\cos(\theta_1) -$$
$$8\cos(\theta_2 - \theta_3)\cos(\theta_3 - \theta_4)\cos(\theta_4 - \theta_5)\cos(\theta_2) + 16\cos(\theta_1 - \theta_2)\cos(\theta_2 - \theta_3)\cos(\theta_3 - \theta_4)\cos(\theta_4 - \theta_5)\cos(\theta_1)] / l$$

A.2 ELEMENTS OF THE MATRIX $\dot{F}(\dot{\theta}, \theta) \in \mathbb{R}^{5\times 2}$

$$\dot{F}(1,1) = \cos(\theta_1)\dot{\theta}_1 / l$$

$$\dot{F}(1,2) = \sin(\theta_1)\dot{\theta}_1 / l$$

$$\dot{F}(2,1) = ([\cos(\theta_2) - 2\sin(\theta_1 - \theta_2)\sin(\theta_1)]\dot{\theta}_2 - [2\cos(\theta_1 - \theta_2)\cos(\theta_1) - 2\sin(\theta_1 - \theta_2)\sin(\theta_1)]\dot{\theta}_1) / l$$

$$\dot{F}(2,2) = ([\sin(\theta_2) - 2\sin(\theta_1 - \theta_2)\cos(\theta_1)]\dot{\theta}_2 - [2\cos(\theta_1 - \theta_2)\sin(\theta_1) - 2\sin(\theta_1 - \theta_2)\cos(\theta_1)]\dot{\theta}_1) / l$$

$$\dot{F}(3,1) = ([\cos(\theta_3) - 2\sin(\theta_1 - \theta_3)\sin(\theta_1) - 2\sin(\theta_2 - \theta_3)\sin(\theta_2) + 4\cos(\theta_1 - \theta_2)\sin(\theta_2 - \theta_3)\sin(\theta_1)]\dot{\theta}_3 -$$
$$[2\cos(\theta_2 - \theta_3)\cos(\theta_2) - 2\sin(\theta_2 - \theta_3)\sin(\theta_2) + 4\cos(\theta_1 - \theta_2)\sin(\theta_2 - \theta_3)\sin(\theta_1) -$$
$$4\cos(\theta_2 - \theta_3)\sin(\theta_1 - \theta_2)\sin(\theta_1)]\dot{\theta}_2 - [2\cos(\theta_1 - \theta_3)\cos(\theta_1) - 2\sin(\theta_1 - \theta_3)\cos(\theta_1) -$$
$$4\cos(\theta_1 - \theta_2)\sin(\theta_2 - \theta_3)\cos(\theta_1) + 4\cos(\theta_2 - \theta_3)\sin(\theta_1 - \theta_2)\sin(\theta_1)]\dot{\theta}_1) / l$$

$$\dot{F}(3,2) = ([\sin(\theta_3) + 2\sin(\theta_1 - \theta_3)\cos(\theta_1) + 2\sin(\theta_2 - \theta_3)\cos(\theta_2) - 4\cos(\theta_1 - \theta_2)\sin(\theta_2 - \theta_3)\cos(\theta_1)]\dot{\theta}_3 -$$
$$[2\cos(\theta_2 - \theta_3)\sin(\theta_2) + 2\sin(\theta_2 - \theta_3)\cos(\theta_2) - 4\cos(\theta_1 - \theta_2)\sin(\theta_2 - \theta_3)\cos(\theta_1) +$$
$$4\cos(\theta_2 - \theta_3)\sin(\theta_1 - \theta_2)\cos(\theta_1)]\dot{\theta}_2 - [2\cos(\theta_1 - \theta_3)\sin(\theta_1) + 2\sin(\theta_1 - \theta_3)\cos(\theta_1) -$$
$$4\cos(\theta_1 - \theta_2)\cos(\theta_2 - \theta_3)\sin(\theta_1) - 4\cos(\theta_2 - \theta_3)\sin(\theta_1 - \theta_2)\cos(\theta_1)]\dot{\theta}_1) / l$$

$$\dot{F}(4,1) = ([\cos(\theta_4) - 2\sin(\theta_1 - \theta_4)\sin(\theta_1) - 2\sin(\theta_2 - \theta_4)\sin(\theta_2) - 2\sin(\theta_3 - \theta_4)\sin(\theta_3) +$$
$$4\cos(\theta_1 - \theta_2)\sin(\theta_2 - \theta_4)\sin(\theta_1) + 4\cos(\theta_1 - \theta_3)\sin(\theta_3 - \theta_4)\sin(\theta_1) + 4\cos(\theta_2 - \theta_3)\sin(\theta_3 - \theta_4)\sin(\theta_2)$$
$$-8\cos(\theta_1 - \theta_2)\cos(\theta_2 - \theta_3)\sin(\theta_3 - \theta_4)\sin(\theta_1)]\dot{\theta}_4 -$$
$$[2\cos(\theta_3 - \theta_4)\cos(\theta_3) - 2\sin(\theta_3 - \theta_4)\sin(\theta_3) + 4\cos(\theta_1 - \theta_3)\sin(\theta_3 - \theta_4)\sin(\theta_1) -$$
$$4\cos(\theta_3 - \theta_4)\sin(\theta_1 - \theta_3)\sin(\theta_1) + 4\cos(\theta_2 - \theta_3)\sin(\theta_3 - \theta_4)\sin(\theta_2) -$$
$$4\cos(\theta_3 - \theta_4)\sin(\theta_2 - \theta_3)\sin(\theta_2) - 8\cos(\theta_1 - \theta_2)\cos(\theta_2 - \theta_3)\sin(\theta_3 - \theta_4)\sin(\theta_1) +$$
$$8\cos(\theta_1 - \theta_2)\cos(\theta_3 - \theta_4)\sin(\theta_2 - \theta_3)\sin(\theta_1)]\dot{\theta}_3 -$$
$$[2\cos(\theta_2 - \theta_4)\cos(\theta_2) - 2\sin(\theta_2 - \theta_4)\sin(\theta_2) - 4\cos(\theta_2 - \theta_3)\cos(\theta_3 - \theta_4)\cos(\theta_2)$$
$$+4\cos(\theta_1 - \theta_2)\sin(\theta_2 - \theta_4)\sin(\theta_1) - 4\cos(\theta_2 - \theta_4)\sin(\theta_1 - \theta_2)\sin(\theta_1)$$
$$+4\cos(\theta_3 - \theta_4)\sin(\theta_2 - \theta_3)\sin(\theta_2) - 8\cos(\theta_1 - \theta_2)\cos(\theta_3 - \theta_4)\sin(\theta_2 - \theta_3)\sin(\theta_1) +$$
$$8\cos(\theta_2 - \theta_3)\cos(\theta_3 - \theta_4)\sin(\theta_1 - \theta_2)\sin(\theta_1)]\dot{\theta}_2 -$$
$$[2\cos(\theta_1 - \theta_4)\cos(\theta_1) - 2\sin(\theta_1 - \theta_4)\sin(\theta_1) - 4\cos(\theta_1 - \theta_2)\cos(\theta_2 - \theta_4)\cos(\theta_1) -$$
$$4\cos(\theta_1 - \theta_3)\cos(\theta_3 - \theta_4)\cos(\theta_1) + 4\cos(\theta_2 - \theta_4)\sin(\theta_1 - \theta_2)\sin(\theta_1) +$$
$$4\cos(\theta_3 - \theta_4)\sin(\theta_1 - \theta_3)\sin(\theta_1) + 8\cos(\theta_1 - \theta_2)\cos(\theta_2 - \theta_3)\cos(\theta_3 - \theta_4)\cos(\theta_1) -$$
$$8\cos(\theta_2 - \theta_3)\cos(\theta_3 - \theta_4)\sin(\theta_1 - \theta_2)\sin(\theta_1)]\dot{\theta}_1) / l$$

$$\dot{F}(4,2) = ([\sin(\theta_4) + 2\sin(\theta_1 - \theta_4)\cos(\theta_1) + 2\sin(\theta_2 - \theta_4)\cos(\theta_2) + 2\sin(\theta_3 - \theta_4)\cos(\theta_3) -$$
$$4\cos(\theta_1 - \theta_2)\sin(\theta_2 - \theta_4)\cos(\theta_1) - 4\cos(\theta_1 - \theta_3)\sin(\theta_3 - \theta_4)\cos(\theta_1) -$$
$$4\cos(\theta_2 - \theta_3)\sin(\theta_3 - \theta_4)\cos(\theta_2) + 8\cos(\theta_1 - \theta_2)\cos(\theta_2 - \theta_3)\sin(\theta_3 - \theta_4)\cos(\theta_1)]\dot{\theta}_4 -$$
$$[2\cos(\theta_3 - \theta_4)\sin(\theta_3) + 2\sin(\theta_3 - \theta_4)\cos(\theta_3) - 4\cos(\theta_1 - \theta_3)\sin(\theta_3 - \theta_4)\cos(\theta_1) +$$
$$4\cos(\theta_3 - \theta_4)\sin(\theta_1 - \theta_3)\cos(\theta_1) - 4\cos(\theta_2 - \theta_3)\sin(\theta_3 - \theta_4)\cos(\theta_2) +$$
$$4\cos(\theta_3 - \theta_4)\sin(\theta_2 - \theta_3)\cos(\theta_2) + 8\cos(\theta_1 - \theta_2)\cos(\theta_2 - \theta_3)\sin(\theta_3 - \theta_4)\cos(\theta_1) -$$
$$8\cos(\theta_1 - \theta_2)\cos(\theta_3 - \theta_4)\sin(\theta_2 - \theta_3)\cos(\theta_1)]\dot{\theta}_3 -$$
$$[2\cos(\theta_2 - \theta_4)\sin(\theta_2) + 2\sin(\theta_2 - \theta_4)\cos(\theta_2) - 4\cos(\theta_1 - \theta_2)\sin(\theta_2 - \theta_4)\cos(\theta_1) +$$
$$4\cos(\theta_2 - \theta_4)\sin(\theta_1 - \theta_2)\cos(\theta_1) - 4\cos(\theta_2 - \theta_3)\cos(\theta_3 - \theta_4)\sin(\theta_2) -$$
$$4\cos(\theta_3 - \theta_4)\sin(\theta_2 - \theta_3)\cos(\theta_2) + 8\cos(\theta_1 - \theta_2)\cos(\theta_3 - \theta_4)\sin(\theta_2 - \theta_3)\cos(\theta_1) -$$
$$8\cos(\theta_2 - \theta_3)\cos(\theta_3 - \theta_4)\sin(\theta_1 - \theta_2)\cos(\theta_1)]\dot{\theta}_2 -$$
$$[2\cos(\theta_1 - \theta_4)\sin(\theta_1) + 2\sin(\theta_1 - \theta_4)\cos(\theta_1) - 4\cos(\theta_1 - \theta_2)\cos(\theta_2 - \theta_4)\sin(\theta_1) -$$
$$4\cos(\theta_2 - \theta_4)\sin(\theta_1 - \theta_2)\cos(\theta_1) - 4\cos(\theta_1 - \theta_3)\cos(\theta_3 - \theta_4)\sin(\theta_1) -$$
$$4\cos(\theta_3 - \theta_4)\sin(\theta_1 - \theta_3)\cos(\theta_1) + 8\cos(\theta_1 - \theta_2)\cos(\theta_2 - \theta_3)\cos(\theta_3 - \theta_4)\sin(\theta_1) +$$
$$8\cos(\theta_2 - \theta_3)\cos(\theta_3 - \theta_4)\sin(\theta_1 - \theta_2)\cos(\theta_1)]\dot{\theta}_1) / l$$

$$\dot{F}(5,1) = ([\cos(\theta_5) - 2\sin(\theta_1 - \theta_5)\sin(\theta_1) - 2\sin(\theta_2 - \theta_5)\sin(\theta_2) - 2\sin(\theta_3 - \theta_5)\sin(\theta_3) -$$
$$2\sin(\theta_4 - \theta_5)\sin(\theta_4) + 4\cos(\theta_1 - \theta_2)\sin(\theta_2 - \theta_5)\sin(\theta_1) + 4\cos(\theta_1 - \theta_3)\sin(\theta_3 - \theta_5)\sin(\theta_1) +$$
$$4\cos(\theta_1 - \theta_4)\sin(\theta_4 - \theta_5)\sin(\theta_1) + 4\cos(\theta_2 - \theta_3)\sin(\theta_3 - \theta_5)\sin(\theta_2) +$$
$$4\cos(\theta_2 - \theta_4)\sin(\theta_4 - \theta_5)\sin(\theta_2) + 4\cos(\theta_3 - \theta_4)\sin(\theta_4 - \theta_5)\sin(\theta_3) -$$
$$8\cos(\theta_1 - \theta_2)\cos(\theta_2 - \theta_3)\sin(\theta_3 - \theta_5)\sin(\theta_1) - 8\cos(\theta_1 - \theta_2)\cos(\theta_2 - \theta_4)\sin(\theta_4 - \theta_5)\sin(\theta_1) -$$
$$8\cos(\theta_1 - \theta_3)\cos(\theta_3 - \theta_4)\sin(\theta_4 - \theta_5)\sin(\theta_1) - 8\cos(\theta_2 - \theta_3)\cos(\theta_3 - \theta_4)\sin(\theta_4 - \theta_5)\sin(\theta_2)$$
$$+16\cos(\theta_1 - \theta_2)\cos(\theta_2 - \theta_3)\cos(\theta_3 - \theta_4)\sin(\theta_4 - \theta_5)\sin(\theta_1)]\dot{\theta}_5 -$$
$$[2\cos(\theta_4 - \theta_5)\cos(\theta_4) - 2\sin(\theta_4 - \theta_5)\sin(\theta_4) + 4\cos(\theta_1 - \theta_4)\sin(\theta_4 - \theta_5)\sin(\theta_1) -$$
$$4\cos(\theta_4 - \theta_5)\sin(\theta_1 - \theta_4)\sin(\theta_1) + 4\cos(\theta_2 - \theta_4)\sin(\theta_4 - \theta_5)\sin(\theta_2)$$
$$-4\cos(\theta_4 - \theta_5)\sin(\theta_2 - \theta_4)\sin(\theta_2) + 4\cos(\theta_3 - \theta_4)\sin(\theta_4 - \theta_5)\sin(\theta_3) - 4\cos(\theta_4 - \theta_5)\sin(\theta_3 - \theta_4)\sin(\theta_3) -$$
$$8\cos(\theta_1 - \theta_2)\cos(\theta_2 - \theta_4)\sin(\theta_4 - \theta_5)\sin(\theta_1) + 8\cos(\theta_1 - \theta_2)\cos(\theta_4 - \theta_5)\sin(\theta_2 - \theta_4)\sin(\theta_1) -$$
$$8\cos(\theta_1 - \theta_3)\cos(\theta_3 - \theta_4)\sin(\theta_4 - \theta_5)\sin(\theta_1) + 8\cos(\theta_1 - \theta_3)\cos(\theta_4 - \theta_5)\sin(\theta_3 - \theta_4)\sin(\theta_1) -$$
$$8\cos(\theta_2 - \theta_3)\cos(\theta_3 - \theta_4)\sin(\theta_4 - \theta_5)\sin(\theta_2) +$$
$$8\cos(\theta_2 - \theta_3)\cos(\theta_4 - \theta_5)\sin(\theta_3 - \theta_4)\sin(\theta_2) + 16\cos(\theta_1 - \theta_2)\cos(\theta_2 - \theta_3)\cos(\theta_3 - \theta_4)\sin(\theta_4 - \theta_5)\sin(\theta_1) -$$
$$16\cos(\theta_1 - \theta_2)\cos(\theta_2 - \theta_3)\cos(\theta_4 - \theta_5)\sin(\theta_3 - \theta_4)\sin(\theta_1)]\dot{\theta}_4 -$$
$$[2\cos(\theta_3 - \theta_5)\cos(\theta_3) - 2\sin(\theta_3 - \theta_5)\sin(\theta_3) - 4\cos(\theta_3 - \theta_4)\cos(\theta_4 - \theta_5)\cos(\theta_3) +$$
$$4\cos(\theta_1 - \theta_3)\sin(\theta_3 - \theta_5)\sin(\theta_1) - 4\cos(\theta_3 - \theta_5)\sin(\theta_1 - \theta_3)\sin(\theta_1) +$$
$$4\cos(\theta_2 - \theta_3)\sin(\theta_3 - \theta_5)\sin(\theta_2) - 4\cos(\theta_3 - \theta_5)\sin(\theta_2 - \theta_3)\sin(\theta_2) + 4\cos(\theta_4 - \theta_5)\sin(\theta_3 - \theta_4)\sin(\theta_3) -$$
$$8\cos(\theta_1 - \theta_2)\cos(\theta_2 - \theta_3)\sin(\theta_3 - \theta_5)\sin(\theta_1) + 8\cos(\theta_1 - \theta_2)\cos(\theta_3 - \theta_5)\sin(\theta_2 - \theta_3)\sin(\theta_1) -$$
$$8\cos(\theta_1 - \theta_3)\cos(\theta_4 - \theta_5)\sin(\theta_3 - \theta_4)\sin(\theta_1) + 8\cos(\theta_3 - \theta_4)\cos(\theta_4 - \theta_5)\sin(\theta_1 - \theta_3)\sin(\theta_1) -$$
$$8\cos(\theta_2 - \theta_3)\cos(\theta_4 - \theta_5)\sin(\theta_3 - \theta_4)\sin(\theta_2) +$$
$$8\cos(\theta_3 - \theta_4)\cos(\theta_4 - \theta_5)\sin(\theta_2 - \theta_3)\sin(\theta_2) + 16\cos(\theta_1 - \theta_2)\cos(\theta_2 - \theta_3)\cos(\theta_4 - \theta_5)\sin(\theta_3 - \theta_4)\sin(\theta_1) -$$
$$16\cos(\theta_1 - \theta_2)\cos(\theta_3 - \theta_4)\cos(\theta_4 - \theta_5)\sin(\theta_2 - \theta_3)\sin(\theta_1)]\dot{\theta}_3 -$$
$$[2\cos(\theta_2 - \theta_5)\cos(\theta_2) - 2\sin(\theta_2 - \theta_5)\sin(\theta_2) - 4\cos(\theta_2 - \theta_3)\cos(\theta_3 - \theta_5)\cos(\theta_2) -$$
$$4\cos(\theta_2 - \theta_4)\cos(\theta_4 - \theta_5)\cos(\theta_2) + 4\cos(\theta_1 - \theta_2)\sin(\theta_2 - \theta_5)\sin(\theta_1) - 4\cos(\theta_2 - \theta_5)\sin(\theta_1 - \theta_2)\sin(\theta_1) +$$
$$4\cos(\theta_3 - \theta_5)\sin(\theta_2 - \theta_3)\sin(\theta_2) + 4\cos(\theta_4 - \theta_5)\sin(\theta_2 - \theta_4)\sin(\theta_2) +$$
$$8\cos(\theta_2 - \theta_3)\cos(\theta_3 - \theta_4)\cos(\theta_4 - \theta_5)\cos(\theta_2) - 8\cos(\theta_1 - \theta_2)\cos(\theta_3 - \theta_5)\sin(\theta_2 - \theta_3)\sin(\theta_1) +$$
$$8\cos(\theta_2 - \theta_3)\cos(\theta_3 - \theta_5)\sin(\theta_1 - \theta_2)\sin(\theta_1) - 8\cos(\theta_1 - \theta_2)\cos(\theta_4 - \theta_5)\sin(\theta_2 - \theta_4)\sin(\theta_1) +$$
$$8\cos(\theta_2 - \theta_4)\cos(\theta_4 - \theta_5)\sin(\theta_1 - \theta_2)\sin(\theta_1) -$$
$$8\cos(\theta_3 - \theta_4)\cos(\theta_4 - \theta_5)\sin(\theta_2 - \theta_3)\sin(\theta_2) + 16\cos(\theta_1 - \theta_2)\cos(\theta_3 - \theta_4)\cos(\theta_4 - \theta_5)\sin(\theta_2 - \theta_3)\sin(\theta_1)$$
$$-16\cos(\theta_2 - \theta_3)\cos(\theta_3 - \theta_4)\cos(\theta_4 - \theta_5)\sin(\theta_1 - \theta_2)\sin(\theta_1)]\dot{\theta}_2 -$$

$$[2\cos(\theta_1-\theta_5)\cos(\theta_1)-2\sin(\theta_1-\theta_5)\sin(\theta_1)-4\cos(\theta_1-\theta_2)\cos(\theta_2-\theta_5)\cos(\theta_1)-$$
$$4\cos(\theta_1-\theta_3)\cos(\theta_3-\theta_5)\cos(\theta_1)-4\cos(\theta_1-\theta_4)\cos(\theta_4-\theta_5)\cos(\theta_1)+4\cos(\theta_2-\theta_5)\sin(\theta_1-\theta_2)\sin(\theta_1)+$$
$$4\cos(\theta_3-\theta_5)\sin(\theta_1-\theta_3)\sin(\theta_1)+4\cos(\theta_4-\theta_5)\sin(\theta_1-\theta_4)\sin(\theta_1)+$$
$$8\cos(\theta_1-\theta_2)\cos(\theta_2-\theta_3)\cos(\theta_3-\theta_5)\cos(\theta_1)+8\cos(\theta_1-\theta_2)\cos(\theta_2-\theta_4)\cos(\theta_4-\theta_5)\cos(\theta_1)+$$
$$8\cos(\theta_1-\theta_3)\cos(\theta_3-\theta_4)\cos(\theta_4-\theta_5)\cos(\theta_1)-8\cos(\theta_2-\theta_3)\cos(\theta_3-\theta_5)\sin(\theta_1-\theta_2)\sin(\theta_1)-$$
$$8\cos(\theta_2-\theta_4)\cos(\theta_4-\theta_5)\sin(\theta_1-\theta_2)\sin(\theta_1)-$$
$$8\cos(\theta_3-\theta_4)\cos(\theta_4-\theta_5)\sin(\theta_1-\theta_3)\sin(\theta_1)-16\cos(\theta_1-\theta_2)\cos(\theta_2-\theta_3)\cos(\theta_3-\theta_4)\cos(\theta_4-\theta_5)\cos(\theta_1)+$$
$$16\cos(\theta_2-\theta_3)\cos(\theta_3-\theta_4)\cos(\theta_4-\theta_5)\sin(\theta_1-\theta_2)\sin(\theta_1)]\dot{\theta}_1)/l$$

$$\dot{F}(5,2)=([\sin(\theta_5)+2\sin(\theta_1-\theta_5)\cos(\theta_1)+2\sin(\theta_2-\theta_5)\cos(\theta_2)+2\sin(\theta_3-\theta_5)\cos(\theta_3)+$$
$$2\sin(\theta_4-\theta_5)\cos(\theta_4)-4\cos(\theta_1-\theta_2)\sin(\theta_2-\theta_5)\cos(\theta_1)-4\cos(\theta_1-\theta_3)\sin(\theta_3-\theta_5)\cos(\theta_1)-$$
$$4\cos(\theta_1-\theta_4)\sin(\theta_4-\theta_5)\cos(\theta_1)-4\cos(\theta_2-\theta_3)\sin(\theta_3-\theta_5)\cos(\theta_2)-$$
$$4\cos(\theta_2-\theta_4)\sin(\theta_4-\theta_5)\cos(\theta_2)-4\cos(\theta_3-\theta_4)\sin(\theta_4-\theta_5)\cos(\theta_3)+$$
$$8\cos(\theta_1-\theta_2)\cos(\theta_2-\theta_3)\sin(\theta_3-\theta_5)\cos(\theta_1)+$$
$$8\cos(\theta_1-\theta_2)\cos(\theta_2-\theta_4)\sin(\theta_4-\theta_5)\cos(\theta_1)+8\cos(\theta_1-\theta_3)\cos(\theta_3-\theta_4)\sin(\theta_4-\theta_5)\cos(\theta_1)+$$
$$8\cos(\theta_2-\theta_3)\cos(\theta_3-\theta_4)\sin(\theta_4-\theta_5)\cos(\theta_2)-$$
$$16\cos(\theta_1-\theta_2)\cos(\theta_2-\theta_3)\cos(\theta_3-\theta_4)\sin(\theta_4-\theta_5)\cos(\theta_1)]\dot{\theta}_5-$$
$$[2\cos(\theta_4-\theta_5)\sin(\theta_4)+2\sin(\theta_4-\theta_5)\cos(\theta_4)-4\cos(\theta_1-\theta_4)\sin(\theta_4-\theta_5)\cos(\theta_1)+$$
$$4\cos(\theta_4-\theta_5)\sin(\theta_1-\theta_4)\cos(\theta_1)-4\cos(\theta_2-\theta_4)\sin(\theta_4-\theta_5)\cos(\theta_2)+$$
$$4\cos(\theta_4-\theta_5)\sin(\theta_2-\theta_4)\cos(\theta_2)-4\cos(\theta_3-\theta_4)\sin(\theta_4-\theta_5)\cos(\theta_3)+$$
$$4\cos(\theta_4-\theta_5)\sin(\theta_3-\theta_4)\cos(\theta_3)+8\cos(\theta_1-\theta_2)\cos(\theta_2-\theta_4)\sin(\theta_4-\theta_5)\cos(\theta_1)-$$
$$8\cos(\theta_1-\theta_2)\cos(\theta_4-\theta_5)\sin(\theta_2-\theta_4)\cos(\theta_1)+8\cos(\theta_1-\theta_3)\cos(\theta_3-\theta_4)\sin(\theta_4-\theta_5)\cos(\theta_1)-$$
$$8\cos(\theta_1-\theta_3)\cos(\theta_4-\theta_5)\sin(\theta_3-\theta_4)\cos(\theta_1)+$$
$$8\cos(\theta_2-\theta_3)\cos(\theta_3-\theta_4)\sin(\theta_4-\theta_5)\cos(\theta_2)-8\cos(\theta_2-\theta_3)\cos(\theta_4-\theta_5)\sin(\theta_3-\theta_4)\cos(\theta_2)-$$
$$16\cos(\theta_1-\theta_2)\cos(\theta_2-\theta_3)\cos(\theta_3-\theta_4)\sin(\theta_4-\theta_5)\cos(\theta_1)+$$
$$16\cos(\theta_1-\theta_2)\cos(\theta_2-\theta_3)\cos(\theta_4-\theta_5)\sin(\theta_3-\theta_4)\cos(\theta_1)]\dot{\theta}_4-$$
$$[2\cos(\theta_3-\theta_5)\sin(\theta_3)+2\sin(\theta_3-\theta_5)\cos(\theta_3)-4\cos(\theta_1-\theta_3)\sin(\theta_3-\theta_5)\cos(\theta_1)+$$
$$4\cos(\theta_3-\theta_5)\sin(\theta_1-\theta_3)\cos(\theta_1)-4\cos(\theta_2-\theta_3)\sin(\theta_3-\theta_5)\cos(\theta_2)+$$
$$4\cos(\theta_3-\theta_5)\sin(\theta_2-\theta_3)\cos(\theta_2)-4\cos(\theta_3-\theta_4)\cos(\theta_4-\theta_5)\sin(\theta_3)-$$
$$4\cos(\theta_4-\theta_5)\sin(\theta_3-\theta_4)\cos(\theta_3)+8\cos(\theta_1-\theta_2)\cos(\theta_2-\theta_3)\sin(\theta_3-\theta_5)\cos(\theta_1)-$$
$$8\cos(\theta_1-\theta_2)\cos(\theta_3-\theta_5)\sin(\theta_2-\theta_3)\cos(\theta_1)+8\cos(\theta_1-\theta_3)\cos(\theta_4-\theta_5)\sin(\theta_3-\theta_4)\cos(\theta_1)-$$
$$8\cos(\theta_3-\theta_4)\cos(\theta_4-\theta_5)\sin(\theta_1-\theta_3)\cos(\theta_1)+8\cos(\theta_2-\theta_3)\cos(\theta_4-\theta_5)\sin(\theta_3-\theta_4)\cos(\theta_2)-$$
$$8\cos(\theta_3-\theta_4)\cos(\theta_4-\theta_5)\sin(\theta_2-\theta_3)\cos(\theta_2)-$$
$$16\cos(\theta_1-\theta_2)\cos(\theta_2-\theta_3)\cos(\theta_4-\theta_5)\sin(\theta_3-\theta_4)\cos(\theta_1)+$$
$$16\cos(\theta_1-\theta_2)\cos(\theta_3-\theta_4)\cos(\theta_4-\theta_5)\sin(\theta_2-\theta_3)\cos(\theta_1)]\dot{\theta}_3-$$

$$
\begin{aligned}
&[2\cos(\theta_2-\theta_5)\sin(\theta_2)+2\sin(\theta_2-\theta_5)\cos(\theta_2)-4\cos(\theta_1-\theta_2)\sin(\theta_2-\theta_5)\cos(\theta_1)+\\
&4\cos(\theta_2-\theta_5)\sin(\theta_1-\theta_2)\cos(\theta_1)-4\cos(\theta_2-\theta_3)\cos(\theta_3-\theta_5)\sin(\theta_2)-\\
&4\cos(\theta_3-\theta_5)\sin(\theta_2-\theta_3)\cos(\theta_2)-4\cos(\theta_2-\theta_4)\cos(\theta_4-\theta_5)\sin(\theta_2)-\\
&4\cos(\theta_4-\theta_5)\sin(\theta_2-\theta_4)\cos(\theta_2)+8\cos(\theta_1-\theta_2)\cos(\theta_3-\theta_5)\sin(\theta_2-\theta_3)\cos(\theta_1)-\\
&8\cos(\theta_2-\theta_3)\cos(\theta_3-\theta_5)\sin(\theta_1-\theta_2)\cos(\theta_1)+8\cos(\theta_1-\theta_2)\cos(\theta_4-\theta_5)\sin(\theta_2-\theta_4)\cos(\theta_1)-\\
&8\cos(\theta_2-\theta_4)\cos(\theta_4-\theta_5)\sin(\theta_1-\theta_2)\cos(\theta_1)+\\
&8\cos(\theta_2-\theta_3)\cos(\theta_3-\theta_4)\cos(\theta_4-\theta_5)\sin(\theta_2)+8\cos(\theta_3-\theta_4)\cos(\theta_4-\theta_5)\sin(\theta_2-\theta_3)\cos(\theta_2)-\\
&16\cos(\theta_1-\theta_2)\cos(\theta_3-\theta_4)\cos(\theta_4-\theta_5)\sin(\theta_2-\theta_3)\cos(\theta_1)+\\
&16\cos(\theta_2-\theta_3)\cos(\theta_3-\theta_4)\cos(\theta_4-\theta_5)\sin(\theta_1-\theta_2)\cos(\theta_1)]\dot{\theta}_2-\\
&[2\cos(\theta_1-\theta_5)\sin(\theta_1)+2\sin(\theta_1-\theta_5)\cos(\theta_1)-4\cos(\theta_1-\theta_2)\cos(\theta_2-\theta_5)\sin(\theta_1)-\\
&4\cos(\theta_2-\theta_5)\sin(\theta_1-\theta_2)\cos(\theta_1)-4\cos(\theta_1-\theta_3)\cos(\theta_3-\theta_5)\sin(\theta_1)-\\
&4\cos(\theta_3-\theta_5)\sin(\theta_1-\theta_3)\cos(\theta_1)-4\cos(\theta_1-\theta_4)\cos(\theta_4-\theta_5)\sin(\theta_1)-\\
&4\cos(\theta_4-\theta_5)\sin(\theta_1-\theta_4)\cos(\theta_1)+8\cos(\theta_1-\theta_2)\cos(\theta_2-\theta_3)\cos(\theta_3-\theta_5)\sin(\theta_1)+\\
&8\cos(\theta_2-\theta_3)\cos(\theta_3-\theta_5)\sin(\theta_1-\theta_2)\cos(\theta_1)+8\cos(\theta_1-\theta_2)\cos(\theta_2-\theta_4)\cos(\theta_4-\theta_5)\sin(\theta_1)+\\
&8\cos(\theta_2-\theta_4)\cos(\theta_4-\theta_5)\sin(\theta_1-\theta_2)\cos(\theta_1)+8\cos(\theta_1-\theta_3)\cos(\theta_3-\theta_4)\cos(\theta_4-\theta_5)\sin(\theta_1)+\\
&8\cos(\theta_3-\theta_4)\cos(\theta_4-\theta_5)\sin(\theta_1-\theta_3)\cos(\theta_1)-\\
&16\cos(\theta_1-\theta_2)\cos(\theta_2-\theta_3)\cos(\theta_3-\theta_4)\cos(\theta_4-\theta_5)\sin(\theta_1)-\\
&16\cos(\theta_2-\theta_3)\cos(\theta_3-\theta_4)\cos(\theta_4-\theta_5)\sin(\theta_1-\theta_2)\cos(\theta_1)]\dot{\theta}_1)/l
\end{aligned}
$$

A.3 PROOF OF THE EXPONENTIAL TRACKING PROPOSITION

The proof of the exponential tracking proposition given in section 5.4 is as follows: (adapted from [43])

Let the error dynamics of a first order linear system be given by

$$
\frac{d}{dt}\begin{bmatrix} e \\ \dot{e} \end{bmatrix} = \underbrace{\begin{bmatrix} 0 & I \\ -K_p & -K_v \end{bmatrix}}_{A}\begin{bmatrix} e \\ \dot{e} \end{bmatrix}
$$

where $K_p, K_v \in \mathbb{R}^{n\times n}$ are positive definite symmetric matrices (in our case they are $K_p = \begin{bmatrix} K_p^x & 0 \\ 0 & K_p^y \end{bmatrix}, K_v = \begin{bmatrix} K_v^x & 0 \\ 0 & K_v^y \end{bmatrix}$). Let also $\lambda \in \mathbb{C}$ be an eigenvalue of *A* with the corresponding eigenvector $\upsilon = (\upsilon_1, \upsilon_2) \in \mathbb{C}^{2n}, \upsilon \neq 0$. Then it leads to

$$\lambda\begin{bmatrix}\upsilon_1\\ \upsilon_2\end{bmatrix}=\begin{bmatrix}0 & I\\ -K_p & -K_v\end{bmatrix}\begin{bmatrix}\upsilon_1\\ \upsilon_2\end{bmatrix}=\begin{bmatrix}\upsilon_2\\ -K_p\upsilon_1-K_v\upsilon_2\end{bmatrix}$$

Since $\lambda=0\rightarrow\upsilon=0$ then $\lambda=0$ is not an eigenvalue of *A*. Furthermore, if $\lambda\neq 0$ then $\upsilon_2=0\rightarrow\upsilon_1=0$. Hence, it is seen that $\upsilon_2,\upsilon_1\neq 0$ and the assumption $\|\upsilon_1\|=1$ can be made without loss of generality. Then it follows that

$$\lambda^2=\upsilon_1^*\lambda^2\upsilon_1=\upsilon_1^*\lambda\upsilon_2=\upsilon_1^*(-K_p\upsilon_1-K_v\upsilon_2)=-\upsilon_1^*K_p\upsilon_1-\lambda\upsilon_1^*K_v\upsilon_1$$

where υ_1^* is the complex conjugate transpose of υ_1. Finally, since $\alpha=\upsilon_1^*K_p\upsilon_1>0$ and $\beta=\upsilon_1^*K_v\upsilon_1>0$, it follows that

$$\lambda^2+\alpha\lambda+\beta=0\qquad\alpha,\beta>0$$

Thus, the real part of λ is negative and hence, the error dynamics exponentially converges to zero.

A.4 LIE GROUPS AND THE LIE ALGEBRA

Lie Groups, which constitute a special class of analytical manifolds, and the associated Lie Algebra are important concepts in differential geometry and algebraic topology. Also, they have wide areas of application in differential geometric control theories and methods. Basic definitions and the results given below are compiled from [76].

Definition*: A *Lie Group G is an analytical manifold which is also endowed with a group structure (i.e., *G* is a set with two structures: G is a group and G is a manifold) such that the multiplication map

$$G\times G\rightarrow G;\quad (g,h)\mapsto gh$$

and the inversion map

$$G \to G; \quad g \mapsto g^{-1}$$

are also *analytic* i.e., both of them are C^{∞} maps.

If the elements $g, h \in G$, the associated left translation L_g, on G by g is given as

$$L_g : G \to G; \quad h \mapsto gh$$

and the associated right translation R_g, on G by g is given as

$$R_g : G \to G; \quad h \mapsto hg$$

Following from the properties of Lie Groups, both L_g and R_g are diffeomorphisms and dL_g and dR_g denote their corresponding differentials.

Definition*: *The Lie Group Bracket. Let X and Y be analytic vector fields on an n dimensional manifold M. The Lie Bracket $[X,Y]$ is defined to be the unique operator such that

$$L_{[X,Y]} = L_X \circ L_Y - L_Y \circ L_X$$

holds, where L_X is the Lie derivative with respect to X. Let $\left(x^1, \ldots x^n\right)$ be the local coordinates around the point $x \in M$ and let

$$X(x) = X^i(x) \frac{\partial}{\partial x^i}\bigg|_x \quad ; \quad Y(x) = Y^i(x) \frac{\partial}{\partial x^i}\bigg|_x$$

From the most general definition of the Lie bracket

$$[X,Y](f) = X(Y(f)) - Y(X(f))$$

The Lie bracket of the vector fields X and Y is defined to be vector field

$$[X,Y](x) = X^i \frac{\partial Y^j}{\partial x^i}\frac{\partial}{\partial x^j}\bigg|_x - Y^i \frac{\partial X^j}{\partial x^i}\frac{\partial}{\partial x^j}\bigg|_x$$

in local coordinates.

Definition*: *Lie Algebra. A Lie algebra is a real vector space constructed on the bilinear operator Lie bracket $[.,.]: g \times g \to g$ such that for all $x, y, z \in g$ the properties

$$[x,y] = -[y,x] \qquad \text{(skew symmetricity)}$$
$$[x,[y,z]]+[y,[z,x]]+[z,[x,y]] = 0 \qquad \text{(Jacobi identity)}$$

hold.

A.5 THE VECTOR RELATIVE DEGREE OF MIMO SYSTEMS

Definition*: *Relative Degree. Consider the nonlinear affine time invariant system given by

$$\dot{x} = f(x) + g(x)u; \qquad x \in \mathbb{R}^n, \quad u \in \mathbb{R}^m$$
$$y = h(x) \in \mathbb{R}^p$$

where *f(x)*, *g(x)* and *h(x)*are smooth, and *h(x)* is the output vector of the system. If p=1, i.e., $y = h(x) \in \mathbb{R}$ is the scalar output of the system, it is said that $y = h(x)$ has a **relative degree *r*** with respect to the input u if;

1) $L_g h(x) = (L_g L_f h)(x) = (L_g L_f{}^2 h)(x) = (L_g L_f{}^{r-2} h)(x) = 0$ identically, in the neighborhood of $x = 0$

2) $(L_g L_f{}^{r-1} h)(x) \neq 0$ at $x = 0$

Basically, r is the number of times the output $y = h(x)$ must be differentiated so that the input u appears in its expression explicitly [58].

Definition*: *The Vector Relative Degree. Consider now the square MIMO system

$$\dot{x} = f(x) + \sum_{j=1}^{m} g_j(x)u; \qquad x \in \mathbb{R}^n, \quad u \in \mathbb{R}^m$$
$$y_i = h_i(x) \qquad i \in \{1, \ldots, p\}; \qquad p \geq 2, \quad m = p$$

Let $r_{i,m}$ be the relative degree for each y_i. Then, for $i \in \{1, \ldots, m\}$, $r_i = \min_{j \in \{1, \ldots, m\}} r_{i,j}$ is the number of times each output y_i has to be differentiated so that at least one of the inputs appear explicitly in its expression. This means

$$y_i^{r_i} = L_f^{r_i} h_i(x) + L_g L_f^{r_i - 1} h_i(x) u(t)$$

Combining all of these m equations leads to

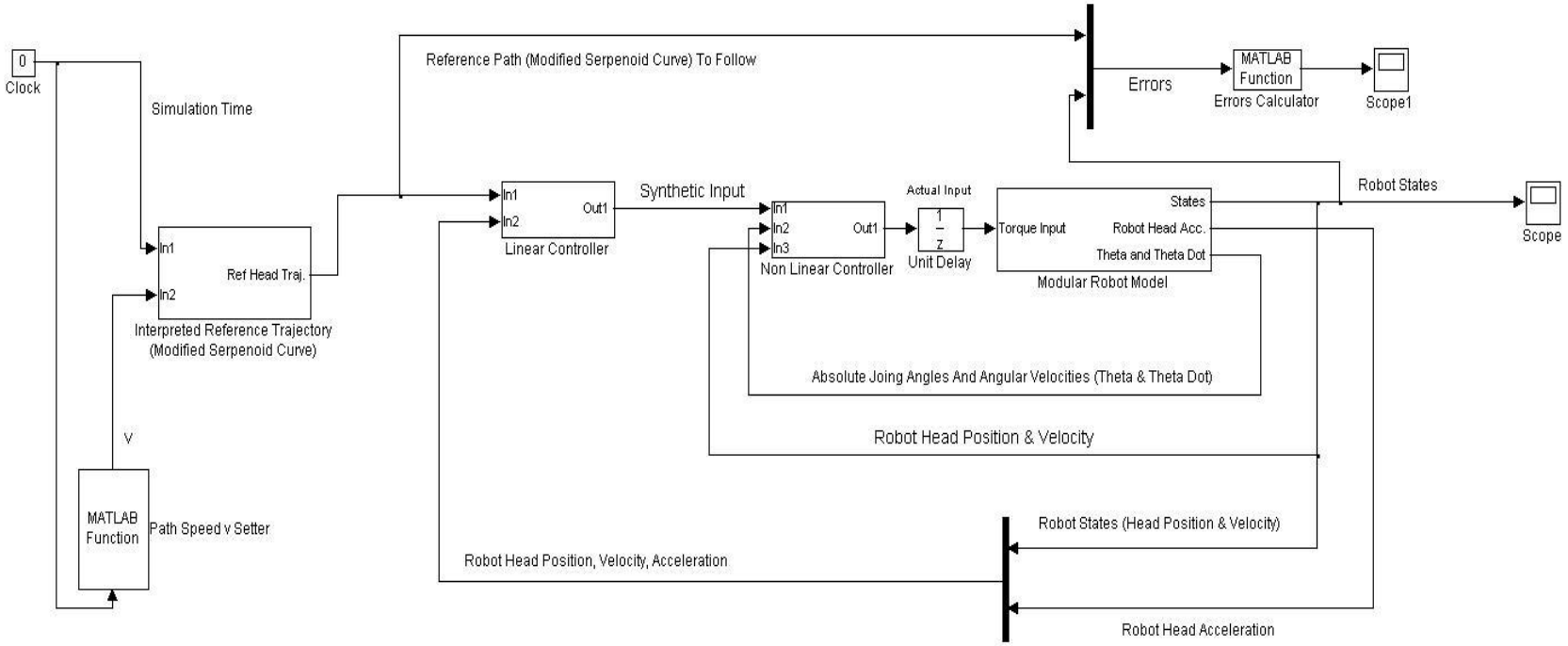
$$\begin{bmatrix} y_1^{r_1} \\ y_2^{r_2} \\ \vdots \\ y_2^{r_m} \end{bmatrix} = A(x) + B(x)u; \quad A(x) \in \mathbb{R}^m, \; B(x) \in \mathbb{R}^{m \times m}$$

Then, $(r_1, \ldots, r_m) \in \mathbb{R}^{1 \times m}$ is called ***the vector relative degree*** of the square MIMO system if $B(x)$ is invertible [58] . If the system is a non square one with $p > m$ and $B(x) \in \mathbb{R}^{m \times p}$, then the decoupling matrix $B(x)$ has to be right invertible [73].

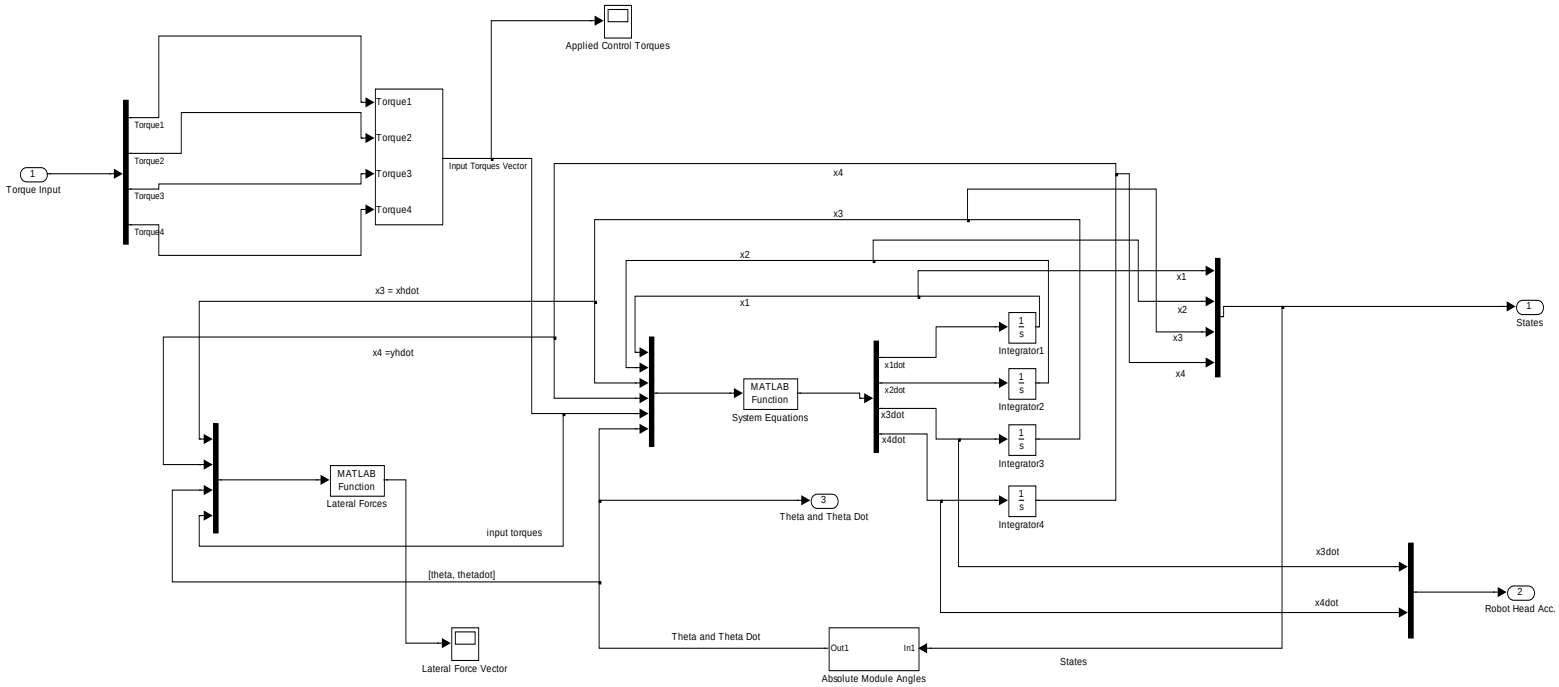
APPENDIX B

MATLAB BLOCK DIAGRAMS AND CODES

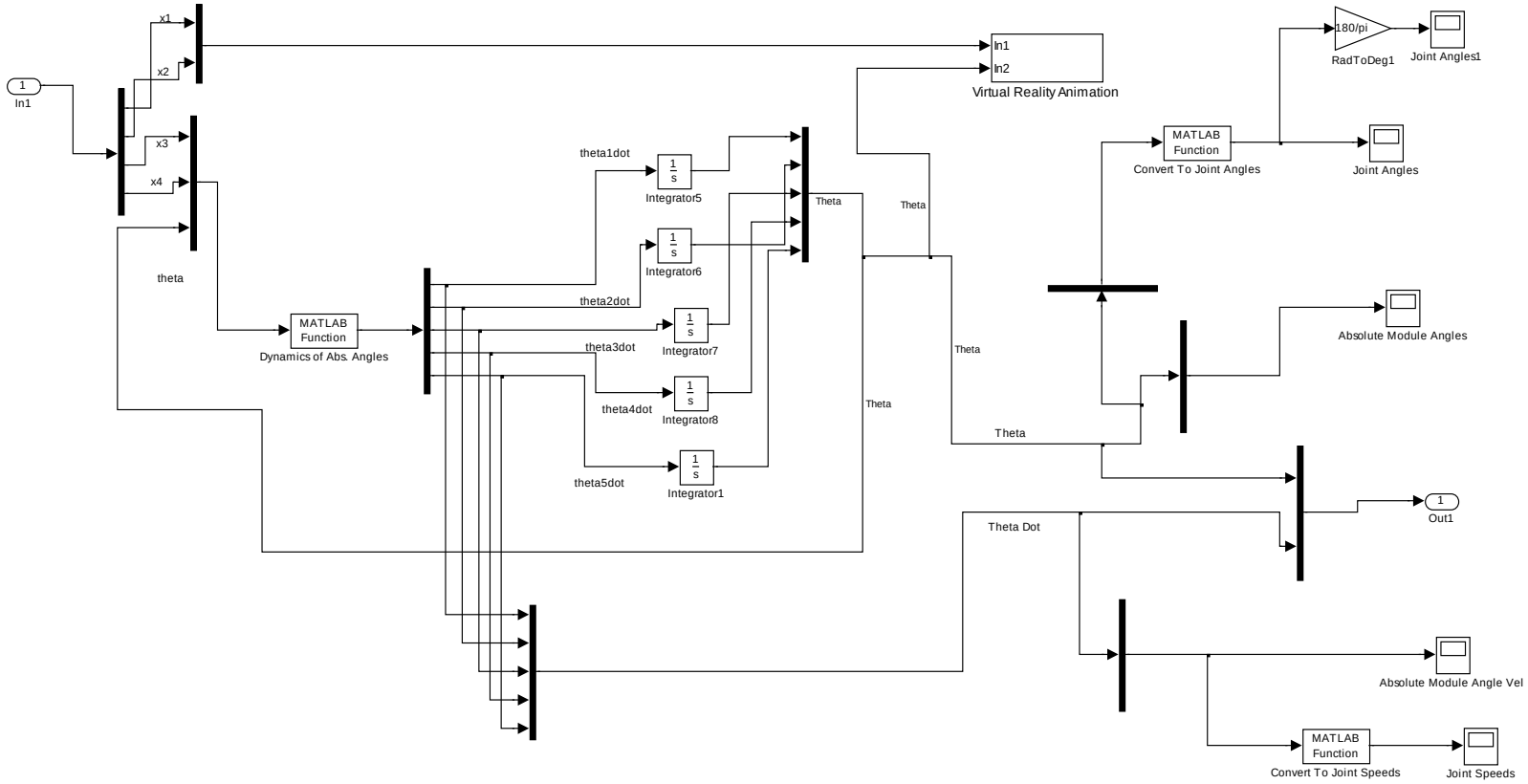
Matlab Simulink block diagrams of the dynamic model of the snake robot are given in this appendix. Also, the developed matlab m-files to generate the modified serpenoid curve are given. Subsystems and m-files are referred by their names which are given in their containing systems and m-files, respectively.



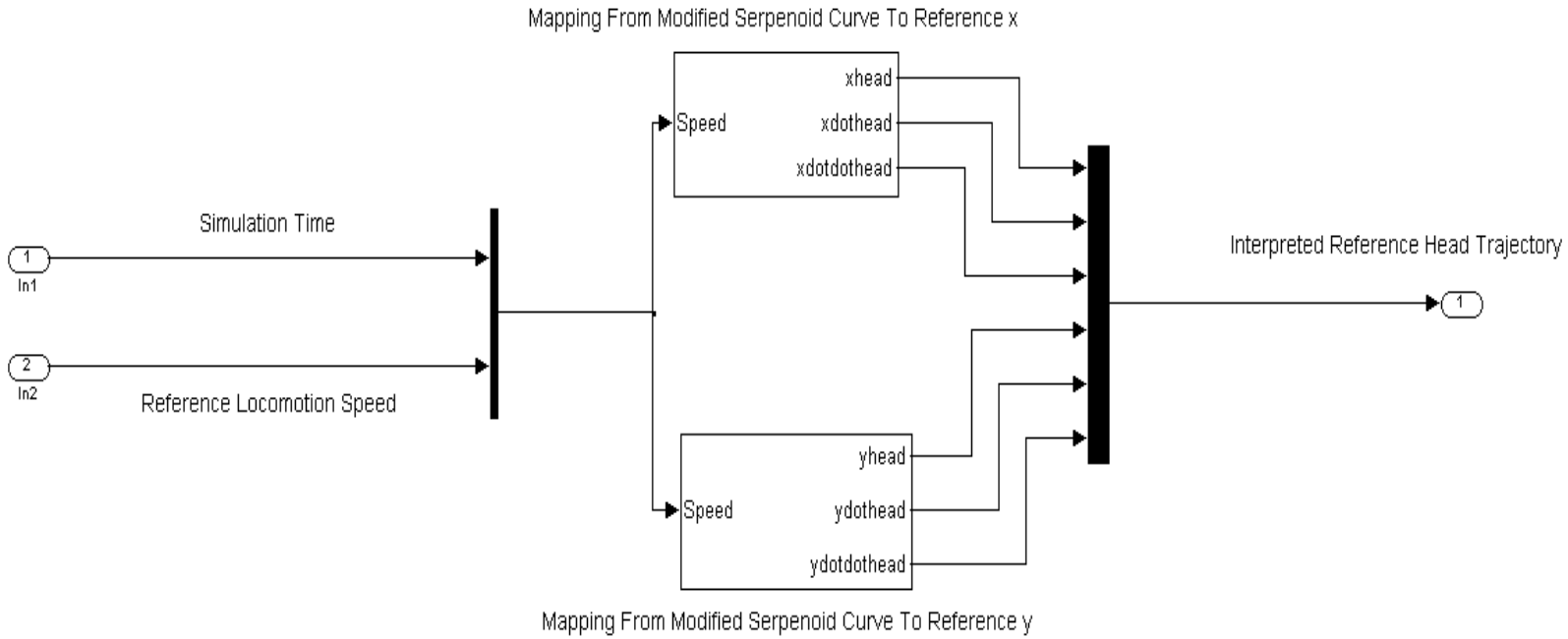
B.1 MATLAB BLOCK DIAGRAM OF THE OVERALL CONTROL ARCHITECTURE



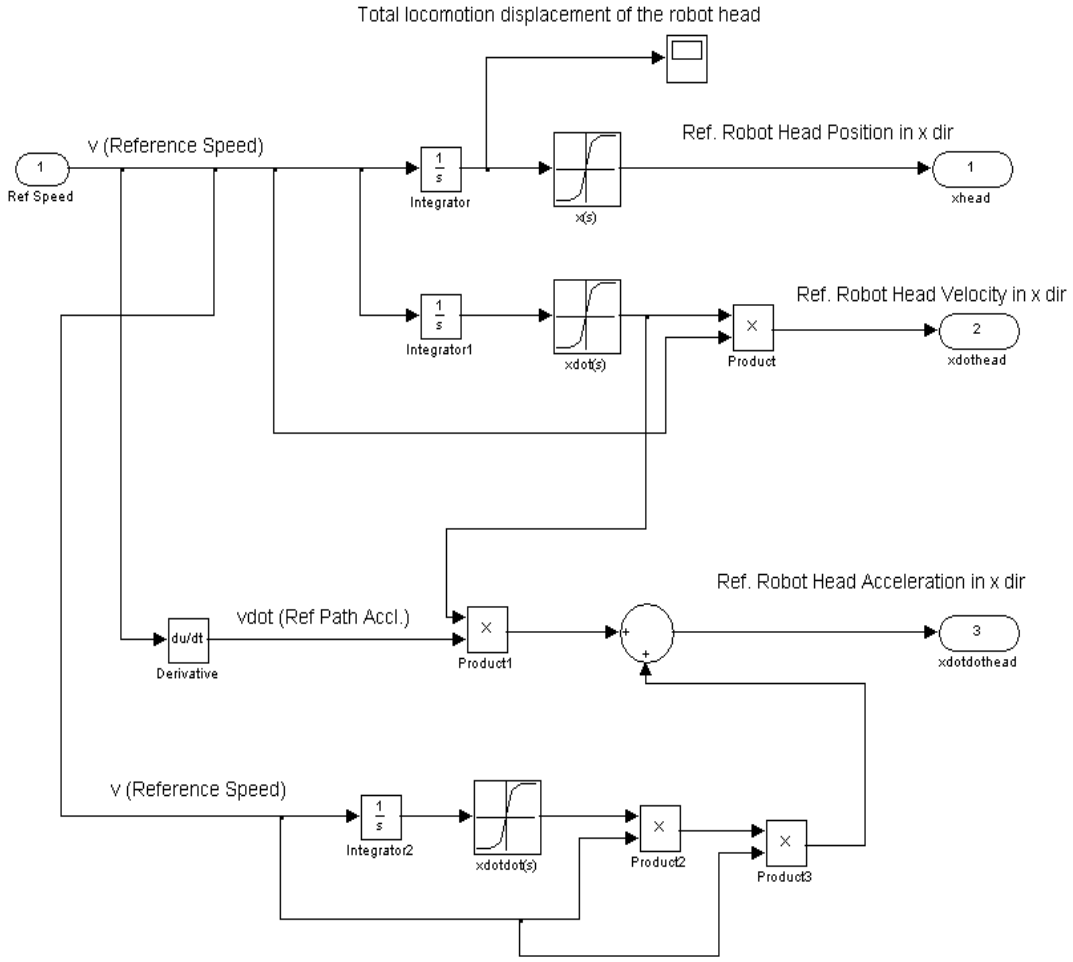
B.2 MATLAB BLOCK DIAGRAM "*MODULAR ROBOT MODEL*" AS THE NONLINEAR PLANT



B.3 MATLAB BLOCK DIAGRAM "*ABSOLUTE MODULE ANGLES*" FOR ANGLES $\theta_1,\ldots,\theta_5$



B.4 MATLAB BLOCK DIAGRAM "*INTERPRETED REFERENCE TRAJECTORY*"



B.5 MATLAB BLOCK DIAGRAM "*MAPPING FROM MODIFIED SERPENOID CURVE TO REFERENCE X* " [9]

[9] Matlab Block Diagram "*Mapping From Modified Serpenoid Curve To Reference y"* has the same structure except for that, interpolators receive y(s) as the input, instead of x(s).

B.6 MATLAB M-FILES FOR THE MODIFIED SERPENOID CURVE

```
%-----------------------------------------------------------------------
% Locomotion And Control of A Modular Snake Like Robot
% Function serpany: called by solver ode15i in script file
% SerpToDesired.m
%-----------------------------------------------------------------------

function Z = serpany(s,x,xprime)

global OldAngle AbsAngle;

tfinal = 100;
dt = 0.01;
Z = zeros(3,1);

r = task(dt,tfinal);
t =0:dt:tfinal;

Pars = SetSerp();
a = Pars(1);
b= Pars(2);

xr = interp1(t,r(:,1),x(3));
yr = interp1(t,r(:,2),x(3));

xrd = interp1(t,r(:,3),x(3));
yrd = interp1(t,r(:,4),x(3));

if (s == 0)

SetOldAngle(atan2(yrd,xrd));
SetAbsAngle(atan2(yrd,xrd));
end

NewAngle = atan2(yrd,xrd);


if NewAngle*OldAngle < 0

    if NewAngle-OldAngle < -pi

        NewAngle = NewAngle + 2*pi;

    elseif NewAngle-OldAngle > pi


 NewAngle = NewAngle - 2*pi;


    end
```

```
end

SetAbsAngle(AbsAngle+(NewAngle-OldAngle));

NDir = atan2((x(2)-yr),(x(1)-xr));
TDir = atan2(yrd,xrd);

NormalAng = pi/2;

Z(1) = xprime(1) - cos(a*cos(b*s)+ AbsAngle);
Z(2) = xprime(2) - sin(a*cos(b*s)+ AbsAngle);

 if abs(abs(TDir)-pi/2)*180/pi > 5

    Z(3) = ((x(2)-yr))*(yrd/xrd) + 1*(x(1)-xr) ;

 else

   Z(3) = ((x(2)-yr)) + (x(1)-xr)*(xrd/yrd) ;

 end

SetOldAngle(atan2(yrd,xrd));

End

%---------------------End of function serpany.m-------------------


%-----------------------------------------------------------------
% Function task: called by the function serpany
% It descritizes the task trajectory and its time derivatives
%-----------------------------------------------------------------

function r = task(dt,tc)

t=0:dt:tc;

sizet = size(t);
sizet = sizet(1,2);


r = zeros(sizet,4,1);

 for z=1:sizet

r(z,1) = -0.5*sin(t(z)*pi/2)+5 ; % x-coordinate of the reference

r(z,2) = 2.25*cos(t(z)*pi/4) + 7.75;% y-coordinate of the %
% reference task path

end
```

```
for z=2:sizet-1
  r(z,3) = (r(z+1,1)-r(z-1,1))/(2*dt) ; % xdot of the reference
  r(z,4) = (r(z+1,2)-r(z-1,2))/(2*dt);% ydot of the reference
end

r(1,3) = r(2,3) ;%initial xdot
r(1,4) = r(2,4) ;%initial ydot

r(sizet,3) = r(sizet-1,3) ;%final xdot
r(sizet,4) = r(sizet-1,4) ;%final ydot

end

%---------------------End of function task.m----------------------



%-----------------------------------------------------------------
% Matlab Script SerpToDesired.m: map fitted ferpenoid curve co
% the referense trajectory. This reference is then forwarded to
% the linear controller
%-----------------------------------------------------------------

global Spath ;

Spath = zeros(6,1);

options = odeset('RelTol', 1e-6);
dt = 0.01;
tfinal = 10;

tspan = 0:dt:150; % time span of the simulation
v = 0.03 ;        % m/s ,speed of the head on the serpenoid curve,
sspan = v*tspan ; % curve length of the serpenoid curve

ds = dt*v ;

TimeCount = size(tspan);
TimeCount = TimeCount(1,2);

Headini = [5 10];
Spath = zeros(TimeCount,6,1);

Pars = SetSerp();

a = Pars(1);
b = Pars(2);

y0 = [5; 10; 0];
yp0 = [cos(a*cos(b)+atan2(0,0)); sin(a*cos(b)+atan2(0,0)); 0];

[T,Y] = ode15i(@serpany,sspan,y0,yp0,options);
```

```
rt = task(dt,Y(TimeCount,3)*1.1); % reference task trajectory to
% the final point robot head

for z=1:TimeCount

    Spath(z,1,1) = Y(z,1); % get the x-pos of Serpenoid Curve
    Spath(z,4,1) = Y(z,2); % get the y-pos of Serpenoid Curve

end

for z=2:TimeCount-1

    Spath(z,2,1) = (Spath(z+1,1,1)-Spath(z-1,1,1))/ds; % set the
% xdot
    Spath(z,5,1) = (Spath(z+1,4,1)-Spath(z-1,4,1))/ds; % set the
% ydot
end

Spath(1,2,1) = Spath(2,2,1) ; % set the initial xdot
Spath(TimeCount,2,1) = Spath(TimeCount-1,2,1) ; % set the final
% xdot

Spath(1,5,1) = Spath(2,5,1) ;% set the initial ydot
Spath(TimeCount,5,1) = Spath(TimeCount-1,5,1); % set the final
% ydot

for z=2:TimeCount-1

    Spath(z,3,1) = (Spath(z+1,2,1)-Spath(z-1,2,1))/ds; % set the
% xdotdot
    Spath(z,6,1) = (Spath(z-1,5,1)-Spath(z-1,5,1))/ds; % set the
% ydotdot
end

   Spath(1,3,1) = Spath(2,3,1);  % set the initial xdotdot
   Spath(TimeCount,3,1) = Spath(TimeCount-1,3,1) ;  % set the
final xdotdot

   Spath(1,6,1) = Spath(2,6,1); % set the initial ydotdot
   Spath(TimeCount,6,1) = Spath(TimeCount-1,6,1); % set the final
% ydotdot

%---------------------End of script SerpToDesired.m---------------
```

B.7 MATLAB M-FILE FOR THE OPTIMIZATION PROCEDURE

```
%-----------------------------------------------------------------
% Matlab Script opt.m: runs the opitimization cylce
% It runs the constructed matlab model for selected parameters in
% the selected ranges
%-----------------------------------------------------------------

global alpha kn;

imax = 6;
jmax = 6;
dt = 0.01;

H = zeros(imax,jmax);

ispan = linspace(15,85,imax);
jspan = linspace(0.5,3,jmax);

SimCounter = 0;

SimTime = 25 ;
stime = 0:dt:SimTime;

tstart = tic;

for i = 1:imax % alpha in degrees

    for j = 1:jmax % Kn

        tstart = tic;

        JLatF = 0;

        Parsx = SetOpt(ispan(i),jspan(j));
        alpha = pi*Parsx(1)/180;
        kn = Parsx(2);

        SerpToDesired;
        warning off all;
        sim('Robot_Control_Architecture');

        GetEnergy;

        for z=2:6
        JLatF = JLatF + trapz(x(:,1),LatForce(:,z).^2)/x(end,1);
        End

        dist = (x(end,2)-rt(:,1)).^2+(x(end,3)-rt(:,2)).^2;
        index = find(dist == min(dist));

        time = 0:dt:index*dt;
```

```
        rt = task(dt,index*dt); % reference task trajectory

        plength = trapz(time, sqrt(rt(:,3).^2+rt(:,4).^2));
        stime = 0:dt:SimTime;
        rlength = trapz(stime, sqrt(x(:,4).^2+x(:,5).^2));

        J = plength/(Energy+rlength*sqrt(JLatF));

        [maxa,ind] = max(H(:));
        [m,n] = ind2sub(size(H),ind)

        SimCounter = SimCounter + 1;
        H(i,j)= J ;
        display(SimCounter);
        stime = toc(tstart);

    end

end


%----------------------End of script opt.m----------------------
```