Application of Dynamic Movement Primitive to Hexapedal Locomotion

A DISSERTATION

Submitted in partial fulfillment of the requirements for the degree of

MASTER OF TECHNOLOGY
in
ELECTRICAL ENGINEERING
(With Specialization in System and Control)

By
JANMEJAYA NANDA

DEPARTMENT OF ELECTRICAL ENGINEERING
INDIAN INSTITUTE OF TECHNOLOGY ROORKEE
ROORKEE, INDIA - 247667

MAY 2016

CANDIDATE'S DECLARATION

I hereby declare that this piece of work entitled APPLICATION OF DYNAMIC MOVEMENT PRIMTIVE TO HEXAPEDAL LOCOMOTION, submitted to the Department of Electrical Engineering, Indian Institute of Technology, Roorkee, India, in partial fulfillment of the requirements for the award of the Degree of Master of Technology in Electrical Engineering with specialization in System and Control is an authentic record of the work carried out by me during the period June 2015 to May 2016, under the supervision of Dr.(Ms) Indra Gupta, Associate Professor in the Department of Electrical Engineering, Indian Institute of Technology, Roorkee. The matter presented in this thesis report has not been submitted by me for the award of any other degree of this institute or any other institutes.

Date:
Place: Roorkee

Janmejaya Nanda

CERTIFICATE

This is to certify that the above statement made by the candidate is true to the best of my knowledge and belief.

Dr.(Ms) Indra Gupta
Associate Professor
Department of Electrical Engineering
Indian Institute of Technology, Roorkee

ABSTRACT

Legged locomotion holds numerous advantage over wheeled one on rough terrain. But computational complexity and use of fix pattern of leg movement in classical control method severely reduces the advantage of legged locomotion. Recent research suggests that a movement is composed of few simpler sub-movements called movement primitives. Dynamic Movement Primitives, DMP is one of the many way of introducing the concept of movement primitive to robotics.

Locomotion requires strong co-ordination within and between the legs, as well as continuous modulation of the trajectories. In this context, Dynamic Movement Primitive (DMP) provides one of the most powerful tool to represent and modulate a movement. In addition to that, different DMPs can be coupled together to achieve complex co-ordination. DMP encodes a trajectory in the form of a differential equation in phase space of the canonical system. Then these differential equations are integrated by a numerical integration method to reproduce the encoded trajectory. This work uses rhythmic DMP to produce locomotion in a virtual hexapod build in Matalb's SimMechanics environment. Instead of using Receptive Field Weighted Regression (RFWR) as a function approximator, a neuro fuzzy inference system ELANFIS has been used. Use of ELANFIS in DMP's framework has been given. Along with that a coupling architecture have been used to couple the canonical system, present in each leg, to achieve inter leg co-ordination. To validate control mechanism Hexapod is trained to perform three gait pattern namely, Tripod, Ripple, Wave. In each experiment Hexapod is trained with a gait pattern and left alone to reproduce the learned gait.

Acknowledgements

I would like to express my deep sense of gratitude and sincere thanks to my guide Dr.(Ms) Indra Gupta, Department of Electrical Engineering, Indian Institute of Technology Roorkee, for her valuable guidance and support. I am highly indebted to her for her encouragement and constructive criticism throughout the course of this project work. Inspite of her hectic schedule, she was always there for clarifying my doubts and reviewed my dissertation progress in a constructive manner. Without her help, this thesis would not have been possible.

Janmejaya Nanda

Contents

List of Figure

List of Tables

Abbreviations

DMP	Dynamic Movement Primitive
CPG	Central Pattern Generator
GRF	Ground Reaction Force
PoWER	Policy learning by Weighting Exploration with the Returns
PI^2	Policy Improvement with Path Integral
CNN	Cellular Neural Network
RFWR	Receptive Field Weighted Regression
ANFIS	Adaptive Neuro-Fuzzy Inference System
ELM	Extreme Learning Machine
ELANFIS	Extreme Learning ANFIS
CAD	Computer-aided Design
GUI	Graphical User interface
DOF	Degrees Of Freedom

Chapter 1

INTRODUCTION

Locomotion in Robot can basically be divided into two types, wheels and legs. On flat surface wheel robot provides one the most energy efficient way of locomotion and they are very easy to control. But when it comes to uneven/rough surfaces they have given very less preferences as they require complete ground support throughout its path for locomotion and it can't always be guaranteed on uneven surfaces. Whereas legged robot requires only a set of foothold positions, which gives it an upper edge.

Due to high maneuverability of legged robot it has widely been used in military mission, surveillance of hazardous environment (like some area of nuclear or chemical plant), civil projects (like inspection of tunnels/pipelines) etc. As the Robots more and more coming into the real world, working alongside with the Humans, they have to deal with the uncertain environment as we and other animals do. So, in the recent year research in the field of legged Robots gaining more and more importance. But, most of control technique that are used in legged robots have a fixed pattern of leg movement which will work well in some but not in all environments. So, by using a control strategy which has a fixed pattern of leg movement severely reduces the advantage of legged locomotion. In this chapter problem with the classical control technique has been discussed very briefly and a few modern controllers like Central Pattern Generator(CPG) and Movement Primitive have been analyzed.

Classical control techniques plan a set of foot hold positions in cartesian co-ordinate and calculate the inverse kinematics to get the desired joint angles. Finding correct foot hold position is key to have a stable locomotion and this process is computationally very expensive. So the robot takes too much time to perform a cycle of locomotion, which we very unlikely see in animals. Along with that classical control technique requires a precise mechanical model. So if model changes, one need recalculate everything in order to able the new model to perform

locomotion. In other scenarios like, malfunctioning of legs/ joints or a leg stuck in an obstacle, then the complete control techniques fails. To take care of these situations programmer has to made contingency plans or robot has to recalculate all the trajectory as a part of contingency plan, which wastes a lot of computational resources.

Most of the recent research suggests that locomotion control in animal is mostly feedforward where the spinal cord receives high level control signal from brain and modulate the firing patterns of its neurons to change locomotion pattern. Feedback from sensory organs and the necessary actions like reflexes are taken care of by other part of the brain. From the experiments of decerebrated cat and many other experiments neurobiologists suggest that rhythmic pattern such as locomotion are generated by central pattern generators (CPGs), a set of interconnected neurons whose output oscillate with certain frequency and amplitude, present in our spinal cord. Gaining support from these Neurobiological studies, a lot of research in the field of legged robot uses CPG based controller technique, which essentially a feedforward control technique, for locomotion.

Various methods have been applied to model a CPG [2] starting from complex biophysical model to more simpler oscillator model. Due to its simplicity Non-linear oscillator models are widely used by researchers to model a CPG. W.Chen et al.[3] have used hopf oscillator and first order low pass filter to generate different gaits in a hexapod robot. They have showed one of the simplest way to perform different gait pattern in the hexapod with the help of a simple oscillator and low pass filter. Each leg is represented by an oscillator which have sinusoidal dynamics. When amplitude of the oscillator exceeds the defined threshold value that corresponding leg is activated. Phase difference between the oscillator is maintained by the time constant parameter of the low pass filter. Dynamics of the low pass filter is designed in such a manner that it will only affect the phase of the oscillator not its amplitude. In[4] Katsuyoshi Tsujita et al. have used two controller, leg motion controller and gait pattern controller, to make the quadruped walk. Leg motion controller is a simple PD controller, whereas gait pattern controller takes care of gait pattern by modifying the phase of the leg. Each leg's phase is modified by considering local sensor information. R. Campos et al.[5] have used a nonlinear hopf oscillator whose swing and stance time can independently be controlled. By varying a simple parameter phase and duty factor of each leg is varied and smooth transition between different gait has been achieved. C. Roghett & A. J. Isspeert in their work [6] have used a modified hopf oscillator and a coupling architecture(discussed in [6]) to model a CPG and interconnection between them. Then they have integrated sensory feedback from the local touch sensor to explicitly change the phase space of the oscillator, which prevents immature lifting of the legs. All the CPG models that have been discussed above numerically integrates the set of coupled differential equations, but Paolo Arena et al. have proposed a unconventional way of modeling a CPG that directly realized in analog circuits [7] called cellular neural network(CNN).

Recent researches have been suggesting that complex movement seen in animals are composed of some elementary movements called movement primitive [2]. Like sentences are made from words and words from phonems, motor control in vertebrate and invertebrates are composed of simpler building blocks. These simpler blocks/motor primitives are bounded to

each other by some rules to form an action [8]. As discussed in [8], motor primitive can be kinematic, behavioral level as it can easily be observed in stroke patient, continuous movement is composed of simpler discrete sub-movements. And as the patient recovers number of sub-movement decreases and overlapping between them increases, producing a smooth and continuous movement. Firing pattern of neurons can be observed as primitives at neural level as the arm motion can be reconstructed from studying the firing pattern of neuron in cerebral cortex. The concept of motor primitives have successfully applied to robotics by Ijspeert et al. [9].

Dynamic Movement Primitive, DMP provides one of the most powerful tool to represent and modulate a movement, where a movement is learned and represented in the form of a differential equation. The framework of DMP is motivated from the analogy that exist between control policies and dynamical system, i.e. both represents change in a state as a function of current state and encodes a desired goal in the form of an attractor. Working principle of DMP is very similar to the learning procedure of animals. We all learns new thing by observing a teacher. Similarly in DMP first a teacher teaches a robot certain task and then the robot is left alone to reproduce the learned task to different real world situations. While teaching, robot records all the kinematics variable of the motion and then uses a sophisticated non-linear approximation technique to learn those movements. Due to its simplicity its getting very popular in the field of robot learning. In their very beginning work [9] Ijspeert et al. have trained the robot to perform tennis fore hand and back hand swing to different ball positions. D. Pongas et al. [10] have taught the robot a complex drumming beat and by modulating the frequency of the canonical system with the external signal it synchronizes its drumming pattern according to the external rhythm generator. Apart from learning these simpler tasks, DMP along with RL have been used to train the robot to perform much more challenging tasks like, under actuated pendulum swing up and playing ball in a cup [11] etc. As described in [12] playing ball in a cup is one of the challenging motor task as the player has to move the cup sidewise to induce a motion in the ball then he has to toss the ball up and catch it in the cup. Its even more challenging in robotics due to requirement of precise co-ordination between multiple degrees of freedom. But DMP along with Policy learning by Weighting Exploration with the Returns (PoWER) able to perform this task whose efficiency roughly comparable to the efficiency of a teen ager. Another policy search technique called Policy Improvement with Path Integral (PI^2)[13] had been integrated with DMP to perform jumping task in Boston Dynamics's virtual little dog. It improves it's jumping trajectory to overcome a gap without falling over.

J. Nakanishi et al. [14] had proposed a framework for learning biped locomotion using DMP. In their work they had obtained the data during the walking of a robot and train the DMP to generate the same trajectory in joint space. With the help of phase resetting and frequency adaptation mechanism robot was able to walk on different surface with different frictional properties.

This work can be viewed as extension of work done by J. Nakanishi et al. [15] [14] to hexapedal locomotion. Unlike the conventional DMP, where Receptive Field Weighted Regression (RFWR)[16] has been used to approximate the *forcing function*, this work uses Extreme Learning Adaptive Neuro-fuzzy Inference System (ELANFIS)[17]. RFWR has various advantage over ELANFIS like, dealing with bias-variance dilemma etc. but these

benefits comes at the cost of tuning few open parameters. Looking at the application of DMP in this work ELANFIS is preferred over RFWR as it has only one meta parameter, parameter which is given as input to the algorithm, to tune. In their work J. nakanishi et al. have used a canonical system per joint, but this work uses canonical system per leg, as it is adequate enough for performing locomotion and gives a compact representation of coupling. By using a canonical system per leg, intra-leg co-ordination has been taken care of by the canonical system itself and for inter-limb co-ordination results obtained in [1] has been used.

To perform experimental work a virtual hexapod built in Matlab's SimMechanics Environment has been used. Designed hexapod is able to perform three different gait pattern namely wave, ripple and tripod in independent experiments. Each leg is designed to follow an ellipsoid like trajectory in task space, which is then converted into joint space by applying inverse kinematics method. These data are then used to train the DMP.

Following sections are designed as follows, in Section 2, a detailed description of SimMechanics and its different components have been given. Along with that a procedure to build a Hexapod model is described. Section 3 starts with a brief overview of DMP, specifically Rhythmic DMP, then use of ELANFIS in DMP's structure is given and finally a coupling architecture is discussed to achieve inter and intra limb co-ordination. Complete experimental result can be found in section 4. And Section 5 ends this thesis with conclusions and possible works for future.

Chapter 2

HEXAPOD MODEL

In order to perform experimental works a platform is used to build a legged robot, where

1. Model can be built very easily. And at the same time that platform has to be widely accepted by the researcher to perform physical world simulation.
2. Robot must be joint actuated, as our controller is generating joint trajectories.
3. Programming on the platform must be programmer friendly and multiple controller can be easily integrated to the model.

SimMechanics in MATLAB best fits to our requirement as it has following features. A multi body system can be built very easily by using existing library files. Programming in Matlab is very user convenient and robot models in SimMechanics are joint actuated. Instead of building a model one can also import a CAD model to its interface to perform experimental work.

Simmechanics is a multibody simulation environment in Matlab, where one can design the rigid body system and define motion between them with the help of existing library files. During simulation Simscape solves the equation of motion for complete mechanical model and show the simulated result in 3D GUI of Matlab. There are two versions of SimMechanics in Matlab, 1st generation, which is found prior to "R2012a" release version and 2nd generation, which has a very improved library and more powerful computational engine, found in latter releases. Compare to 1st generation, latest generation i.e. 2nd have various advantages

- Movement of inertia of a body is calculated automatically from given geometry and mass

- An advance visualization based on computer graphics, which doesn't slow down simulation speed while animation is turned on.
- Animation replay option is available.
- Initial states can be defined in each joint blocks.

During beginning of the thesis work, a Hexapod was build using SimMechanics 1st generation but due to its slower simulation speed and lack of ability to set initial joint targets 2nd generation has been preferred. 2nd generation of SimMechanics provides complete different way of designing a model than its 1st generation. As discussed above, this work uses 2nd generation of Simmechanics to build a Hexapod, so next few paragraphs will only focus on 2nd generation.

In Simmechanics one can very easily build a complex mechanical system by simply joining different predefined blocks. It is quite similar to assembling different parts while building mechanical structure in physical world. Each blocks in SimMechanics have some specific uses and according to the task it performs each have been categorized into different libraries. Few of them have been discussed bellow

Body Element : This library contains different body blocks for representing a simpler rigid structure. "Solid" blocks serve as basic building blocks for designing a compound rigid structure. It can be used to represent a cylinder, brick, sphere or ellipsoid etc. and movement if inertia of each structure is calculated from it's mass and geometry. "Inertia" block is used to add point mass with custom shape and center.

Constraint : As the name suggests this library contain blocks to add angular or distance metric constraint between two rigid bodies.

Force and Torque : Different blocks of this Library are used to provide internal/external torques/ forces to the attached frame.

Frame and Transform : This library contains blocks to add or transform frames in a rigid body. "World Frame" is a unique motionless, right handed co-ordinate system, serves as base co-ordinate for any mechanical Model. "Rigid Transform" blocks can be used define rotational and translational transformation between two frames independently.

Joint : It contains all sort of joint that are used in mechanical structure, starting from immovable "weld" joint to completely freely movable "6-DOF" joint. Each and every joint block performs only one task i.e. it defines the motion between two rigid bodies. For example, " Revolute" joint block allows motion around "z-axis" only.

By joining different blocks in a specific manner any mechanical model can be build. In the next section, building of single leg of the hexapod has been described and then this work has been extended to build a complete Hexapod model.

2.1 Building of Hexapod in SimMechanics

In order to familiar to with different blocks of SimMechanics, first single leg of Hexapod has been build. Then it is extended to build a complete Hexapod. Hexapod model consists of leg subsystems, a body and a ground subsystem (**Fig. 2.1).** There are six leg

subsystem represented by their type. Each leg subsystem block receives the joint angle command form the controller and the reference co-ordinate frame from the body subsystem. State of each joint is taken as output from the system and given to the continuous trajectory generator block discussed in section 4.1. Body subsystem outputs the co-ordinate frame to attach legs.

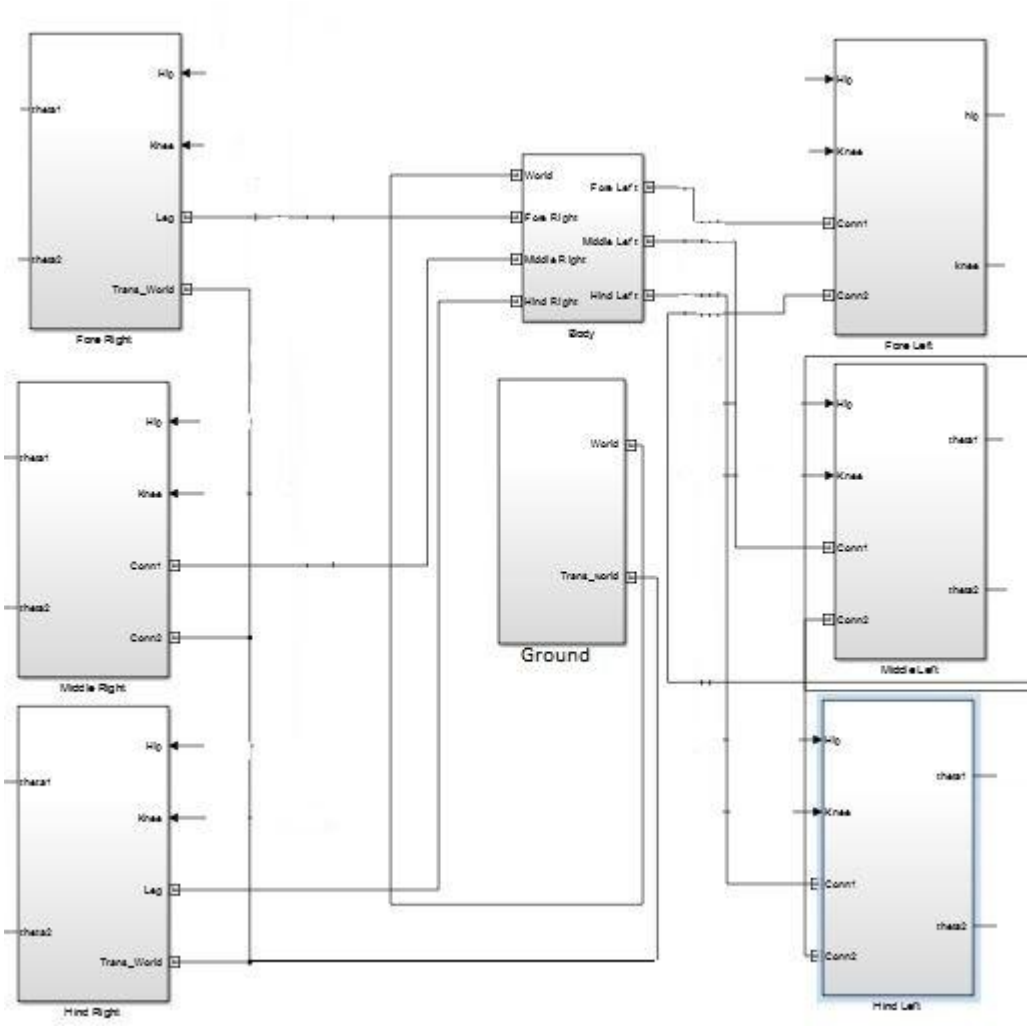

Fig. 2.1 Complete Hexapod model showing Ground, Leg and Body subsystems.

2.1.1 Building Single Leg

Each leg of Hexapod is having 2DOF per leg. So, the properties like, mass, dimension etc. of thigh and shank are defined inside the solid block named “Thigh” and “Shank” (given in Table 2.1). Then these two blocks are connected together through a revolute joint “Knee joint”. It should be noted that revolute joint is a 1DOF joint, which allows movement around its z-axis only. So “Rigid transform” blocks are used to set the co-ordinate frame in such a manner that, thigh and shank move in a desired manner. Internal dynamics of each joint is set to zero and initial joint target is set according to current locomotion pattern. Each leg is connected to the body subsystem through “ HIP joint”. “PS-converter” blocks are used convert Physical signal to Simulink input signal and vice-versa. PD controller is used track the joint angle command coming from the controller.

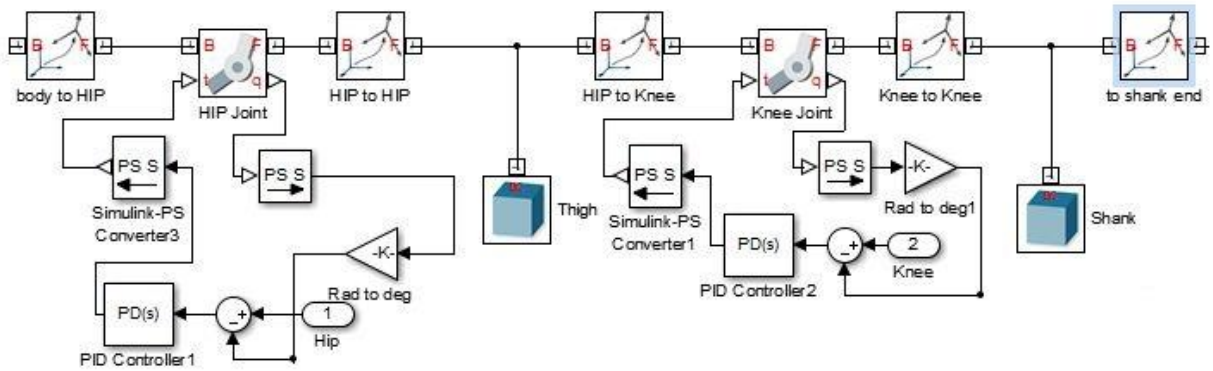

Fig. 2.2 Single Leg model in SimMechanics

2.1.2 Building Complete Hexapod Model

As shown in **Fig 2.5,** complete hexapod model consists of three subsystem namely Leg, Body and Ground. Details about Leg subsystem is given above and it is worth noting that legs on each side of the body (i.e. all Right/Left legs) are identical to each other but some dissimilarity exist between the legs on opposite side of the body(i.e. between left and right leg). So leg model for both side are designed separately. Body subsystem (**Fig. 2.3**) consists of a "solid" blocks, specifying the body parameters(Table 2.1) and "Rigid Transform" blocks to specify the co-ordinate frame to connect legs. And each leg is connected to the body through a revolute joint. Design of hexapod model completes with ground subsystem(**Fig. 2.4**), which essentially a solid block with infinite mass and dimension. Body subsystem connected to the ground Subsystem through a "6-DOF joint".

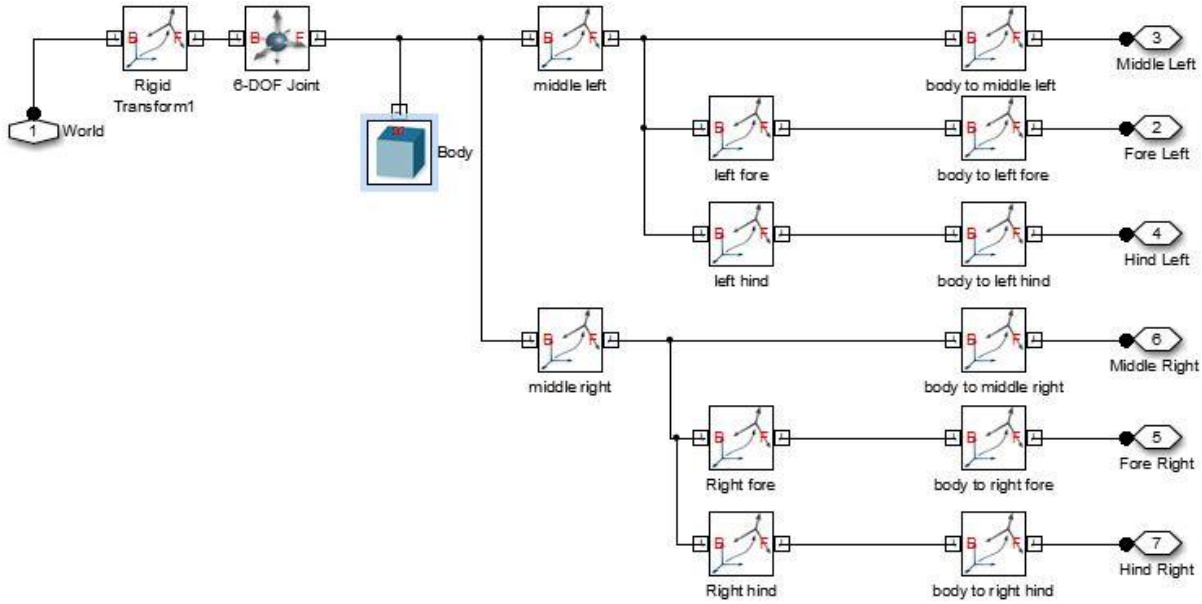

Fig. 2.3 SimMechanics model for Body Subsystem

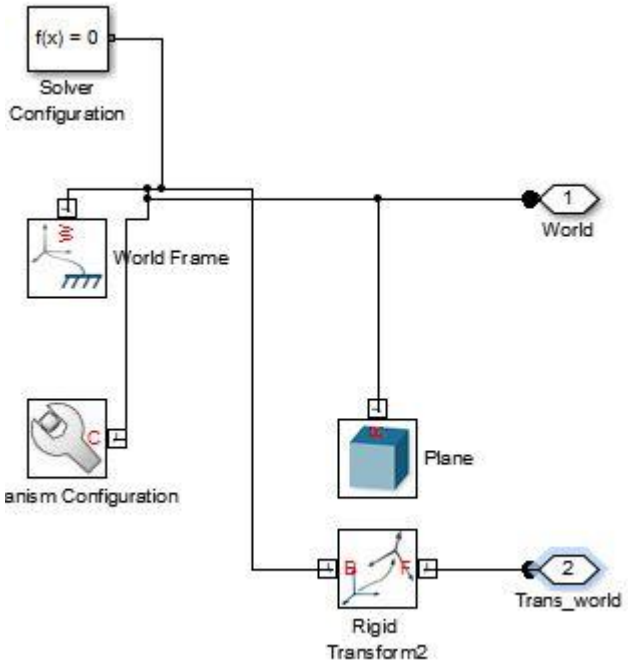

Fig 2.4 SimMechanics model for Ground Subsystem

	Mass (Kg)	**Length (m.)**	**Width (m.)**	**Thickness (m.)**
Body	2	0.5	0.3	0.05
Thigh	0.5	0.15	0.01	0.001
Shank	0.5	0.15	0.01	0.001

Table 2.1 Different parameters of Leg and Body of Hexapod

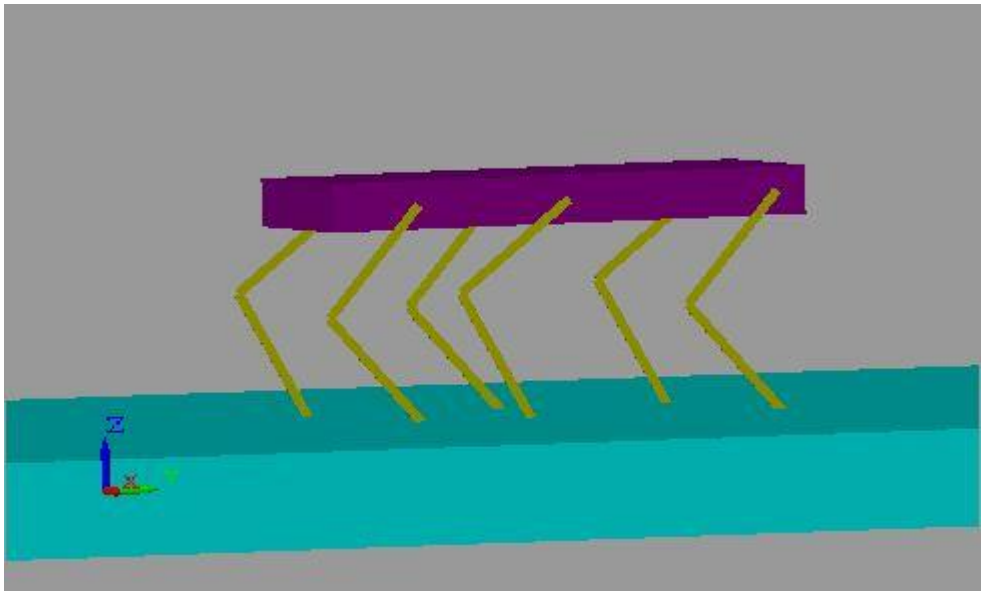

Fig. 2.5 Figure of Virtual Hexapod model build in SimMechanics II

One of the demerits of building a robot in SimMechanics is that, it doesn't allow the detection of collision between two rigid bodies. That means, two rigid bodies will pass through each other without colliding. Which is happened in this case, leg of hexapod instead of colliding with the ground, passes through it. In order to keep the legs above the ground or make the Hexapod walk another subsystem has been developed, called GRF (Ground Reaction Force) subsystem.

2.2 GRF Model

A spring-damper system between leg touchdown point and leg is used to model the GRF (**Fig 2.7**). When a leg tries to penetrate through the ground, a virtual spring-damper system present on the surface of the ground provides the necessary reaction force to prevent penetration. In a similar manner another spring damper model is used to provide the force in horizontal direction, which serves as frictional force. Below equations describes the frictional force(F_x) and ground reaction forces(F_y)

$F_x = -K_x(x - x_0) - b_x(\dot{x} - \dot{x_0})$ in contact with ground

$= 0$ not in contact with ground

$F_y = -K_y y - b_y \dot{y}$ in contact with ground

$= 0$ not in contact with ground

K_x, K_y, b_x, b_y are the spring and damper constant in 'x' and 'y' direction respectively. x_0 is the leg touch down point on x- axis. And for y-axis, y_0 is considered 0. $\dot{x_0}$ is the touchdown velocity in x-direction. x and y are current position of end effector of the leg.

While implementing in Simulink, a sensor block is used to sense leg's x and y position and velocity. Whenever y position of a leg falls bellow zero, indicating leg in contact with the ground. At that moment x-position and velocity are stored and necessary reaction and frictional force is applied to the leg. Model generating frictional force is shown in the figure bellow.

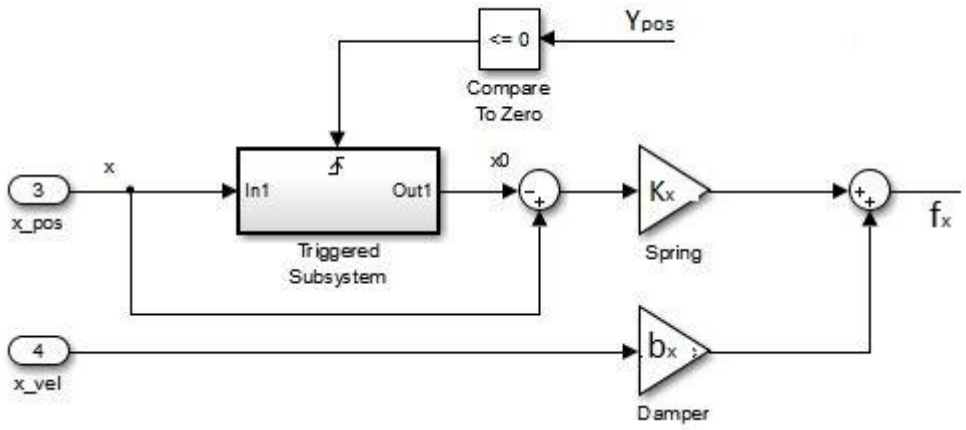

Fig. 2.6 Block diagram showing generation of frictional force

Figure 2.7 shows the contact modeling with spring damper system in both horizontal and vertical direction. Values of spring and damper constant are given in Table 2.2. Finding right value for the spring damper is one the challenging task, because smaller the value more the leg penetrates into the ground. And for higher value, above differential equation becomes very stiff and Matlab solver takes too much time to solve it. As the designed controller is completely open loop a few hundred change in parameters value(given in Table 2.2), Hexapod will behave in a complete different manner.

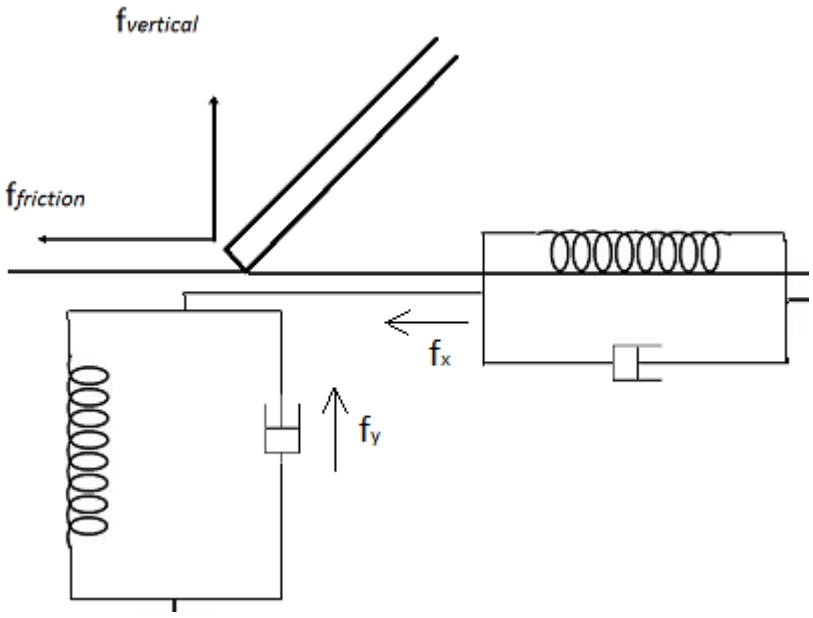

Fig. 2.7 Force exerted by spring, damper system on a leg during contact with the ground.

	Spring	**Damper**
Horizontal	2130	1056
Vertical	3910	2745

Table 2.2 Value of Spring and Damper constant for both Horizontal and Vertical force

Chapter 3

LEARNING LOCOMOTION USING DMP

Recent researches have suggested that, complex movement seen in animals are composed of some elementary movements called movement primitive [2]. Like sentences are made from words and words from phonems, motor control in vertebrate and invertebrates are composed of simpler building blocks. These simpler blocks/motor primitives are bounded to each other by some rules to form an action [8].

As discussed in [8], motor primitive can be kinematic, at behavioral level, as it can easily be observed in stroke patient that continuous movement is composed of simpler discrete sub-movements. And as the patient recovers number of sub-movement decreases and overlapping between them increases, producing a smooth and continuous movement. Firing pattern of neurons can be observed as primitives at neural level as the arm motion can be reconstructed from studying the firing pattern of neuron in cerebral cortex.

The concept of motor primitives have successfully been applied to robotics by Ijspeert et al.[9], where a dynamical system is used to encode a trajectory. Discrete movements are encoded through a point attractor dynamics whereas oscillator are used for rhythmic movement. As the locomotion is a rhythmic task in the next section discusses only about rhythmic DMP.

This chapter organized as follows, first a brief discussion about the structure and learning procedure of rhythmic DMP is given. Then the use of ELANFIS as function approximator in DMP's framework has been discussed. And in the end a coupling architecture is given, which plays a vital role in inter limb co-ordination.

3.1 Rhythmic DMP

DMP [9] is a trajectory encoding scheme, where trajectories are encoded in the form of a non-linear differential equation. Representation of DMP is motivated from the analogy that exist between control policies and dynamical system, i.e. both represents change in a state as a function of current state and both encodes a desired goal in the form of an attractor.

A simple 2nd order linear dynamical system represented by the equation $\ddot{y} = c_1\dot{y} + c_2(g - y)$ is sufficient to represent a movement. where y represents angular position, $\dot{y}$ velocity and $\ddot{y}$ acceleration. Solving the above DE for appropriate value of $c_1$and c_2 , a monotonically increasing y starting from any desired position y_0 to goal/attractor g can be obtained (**Figure 3.1**). if we add some non-linear term to the above DE a complex trajectory (as shown in the **Figure 3.2**) can be created.

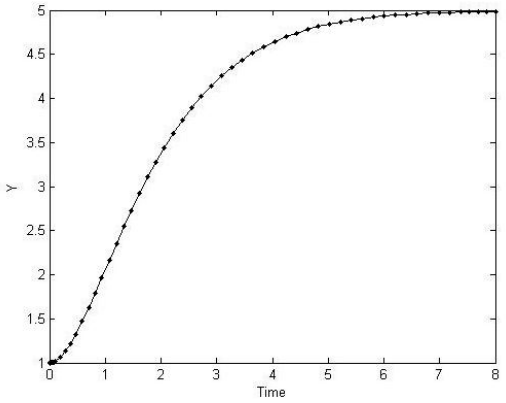

Fig 3.1 : Plot of y for a critically damped system starting from y_0 =1 to goal $g = 5$

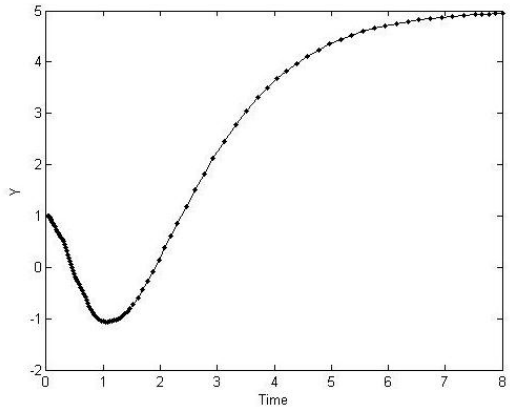

Fig 3.2 : Solution of differential equation after adding some Non-linear term

In the original frame work [9], DMP consists of two 2nd order differential equation, a transformation system and a canonical system. In a very generalized form both the equations are given bellow

$$\ddot{y} = h(y, \dot{y}, x, w) \quad 3.1$$

$$\dot{x} = g(x, \dot{x}) \quad 3.2$$

In the transformation system (**Eq. 3.1**), w is a parameter vector, which shapes y's trajectory to any desired complex shape and it is the parameter which is learned by the function approximator. And job of the canonical system (**Eq. 3.2**) to make transformation system time-independent. Time invariance property is highly desirable as one can couple different canonical system together to achieve desired co-ordination between the transformation system. In their original work[9], a 2nd order dynamical system has been proposed for canonical system but in the later research it had been suggested that a 1st order system is sufficient to represent it.

In this thesis the convention used in [18] is followed. A movement in one dimension can be obtained by integrating following set of differential equation, known as transformation system.

$$\tau \dot{z} = \alpha_z(\beta_z(y_0 - y) - z) + f \quad 3.3$$

$$\tau \dot{y} = z \quad 3.4$$

Where the variables y, z, $\dot{z}$ represents position, velocity and acceleration of the movement respectively, τ is the temporal scaling factor, α_z, β_z are constants which determines damping of the above system, y_0 represents the base-line of oscillation and f is a non-linear forcing function, which makes the above system to represent any complex trajectory. This forcing function is learned with the help of a function approximator (ELANFIS, in this case. See section 3.2). Basis functions of these function approximator are made dependent on the solution of another differential equation, canonical system. A simple phase oscillator(**Eq. 3.5**) is most commonly used as *canonical system* in rhythmic DMP.

$$\tau \dot{\phi} = 1 \quad 3.5$$

Where ϕ is the phase of oscillator, $\phi \in [0, 2\pi]$. Amplitude of oscillation is assumed as one in this work.

Working principle of DMP is very similar to the learning procedure of animals. We all learns new thing by observing a teacher. Similarly, in DMP first a teacher teaches a robot certain task and then the robot is left alone to reproduce the learned task to different real world situations. The next section discuses, how DMP learns and reproduce a trajectory. And finally different distinguishing characteristics of DMP which makes him popular in the field of trajectory learning are given.

3.1.1 Learning a Movement

One of the distinguishing characteristic of DMP is that, it can learn from a single demonstration. During demonstration of a teacher, all the kinematics variables like joint angle, velocity and acceleration of the movements are recorded and latter they are used to train the DMP. There are several ways one can get these data for training, In various experiment authors have used different methods to get this information. For e.g. learning a tennis swing [9] or performing ball in a cup experiment [19] they have used sense suit to record the trajectory, but for same ball in a cup experiment[11] trajectory has been recorded by taking it robot in hand. For some cases where Human and Machine interaction is not that straight forward (like

generating swing and stance data for legged robot) one can give this initial knowledge in hand coded form[20]. In this thesis for locomotion task, each leg of the robot is made to follow an ellipsoid trajectory in the task space (see figure 4.2) and with the help of inverse kinematics these trajectories are transformed into joint space (see section 4.2 for more details) to obtained the data for training.

These data are then used in the following equation to generate target for function approximator

$$f_{target} = \tau^2 \ddot{y}_{demo} - \alpha_z(\beta_z(g - y_{demo}) - \tau \dot{y}_{demo}) \quad 3.6$$

Where $y_{demo}, \dot{y}_{demo}, \ddot{y}_{demo}$ are the recorded joint angle, velocity and acceleration respectively. f_{target} is the target forcing function which has to be learned with the help of a statistical learning rule and input to these function approximators are obtained by integrating the canonical system(Eqn. 3.5). In this work ELANFIS has been used as function approximator to approximate the nonlinear forcing function. More details about ELANFIS and its use in DMP has been given in section 3.2.

3.1.2 Reproducing the Learned movement

Learned parameters of the function approximator are used approximate the forcing function, which is then added to the Transformation system to shape the trajectory and the solution of canonical system drives the function approximator. In order to reproduce the learned movement both *Transformation System* and *Canonical System* are numerically integrated. And by varying the high level parameters (e.g. τ, y_0 etc) one can change the behavior of the task in a particular context.

3.1.3 Distinguishing Characteristics of DMP

Few of the distinguishing characteristics which makes DMP very popular among the robotics researchers are given bellow

- **Co-ordination in Multi DOF :** In multi DOF robot, a transformation system is used to represent each DOF and a canonical system is shared between them to achieve co-ordination. Even coupling among the canonical systems are used achieve co-ordination between them.
- **Learning complex task :** it can very easily be integrated with RL(Reinforcement Learning) technique to learn very complex motor task.
- **Movement recognition :** with the help of RFWR, DMP can be used to recognize a movement. This can be done by comparing weights of function approximator of two movements.

3.2 Use of ELANFIS in DMP

Receptive field weighted regression(RFWR) is the most popular choice for function approximator in DMP's framework. This is due to it's fast learning algorithm and ability to achieve stable parametrization. Fast learning procedure is highly desirable as the robot has to modify its trajectory online to different real world situations. Having a stable parametrization comes handy while recognizing a movement like hand written recognition. As different kernels are learned independent of each other for a particular task/trajectory weights of linear models are stable and invariant to change in high level parameters like τ, g, r etc., where each of them is having their usual meaning in DMP.

As described in [18], one can made any changes to DMP unless and until it is not disturbing its basic design principle. i.e. A canonical system, which should generate the necessary phase to avoid explicit time dependency. A dynamical system as Transformation System, whose stability can be easily analyzed when exited with non-linear forcing function. And a function approximator, to learn the non-linear forcing function. Few of the striking feature of RFWR which makes it popular in DMP's framework are its fast learning method and ability to add and modify *receptive fields* whenever required. These flexibilities come at the cost of tuning few open loop parameters like, threshold for adding or removing receptive fields, first or second order learning rates etc. This work, with the help of ELANFIS, reduces these open parameters which is very convenient from function approximation point of view. Like RFWR, ELANFIS has very fast learning algorithm but it lacks in the ability of adding/removing the receptive fields. These addition and/or modification of receptive fields plays a key role while recognizing a movement as parameters of the receptive field remains constant for a particular task [18]. Movement recognition is very useful while identifying hand written characters [18], but certainly a redundant advantage while performing locomotion. Looking into these facts, this work uses ELANFIS, which essentially have only one open parameter i.e. number of hidden neurons. ElANFIS also has one shot learning algorithm which makes him equally applicable for online learning.

3.2.1 ELANFIS

ELANFIS[17] is a neuro fuzzy learning machine which combines the structure of ANFIS[21] and the learning mechanism of ELM[22]. It overcomes the drawback of ELM and ANFIS. By adopting sugeno type fuzzy system it overcomes the randomness associated with the ELM strategy and the Computational complexity of ANFIS is hugely reduced by using ELM like learning method.

As shown in the figure 3.3 ELANFIS has same structure as that of ANFIS. Given figure shows the structure for two inputs and each input is having two membership functions. To understand the figure clearly a sugeno type two rules are given bellow

$$if\ X\ is\ A_1 and\ Y\ is\ B_1\ \ then\ \ f_1 = p_1x + \ q_1y + \ r_1$$

$$if\ X\ is\ A_2 and\ Y\ is\ B_2\ \ then\ \ f_2 = p_2x + \ q_2y + \ r_2$$

First part of the rule (before 'then') is represented in the top part of the figure called the Premise part and the bottom part of the figure represents the consequent part of the rule. A_1, A_2 and

B_1, B_2 are the fuzzy membership functions. Most common choice for the membership function is ball shape function given as,

$$\mu(x) = {}^{1}\Big/_{1+\left(\frac{x-c_i}{a_i}\right)^{2b_i}}$$

where c_i, a_i, b_i are the center and shape deciding parameters. Involvement of input 'X' and 'Y' to these membership function is given by μ(x) and μ(y) called the membership grades. Firing strength of the fuzzy rule is calculated by multiplying (while finding T-norm) the membership grades of the inputs, which is done in the second layer of the premise part of the figure. Apart from finding the T-norm two membership grades can be 'OR'ed together to find the firing strength. The firing strength w_i of the fuzzy rule is given as

$$w_i = \mu_{A_i}(x) * \mu_{\boldsymbol{B_i}}(y) \qquad 3.7$$

Then these firing strengths are normalized and multiplied with the consequent part of the rule. Parameters of the consequent part i.e. p,q,r are learned with by using least square estimation method. Second layer of the consequent part in the figure 3.3 finds the final output as

$$t = \sum \overline{w_i}\, x_i \qquad 3.8$$

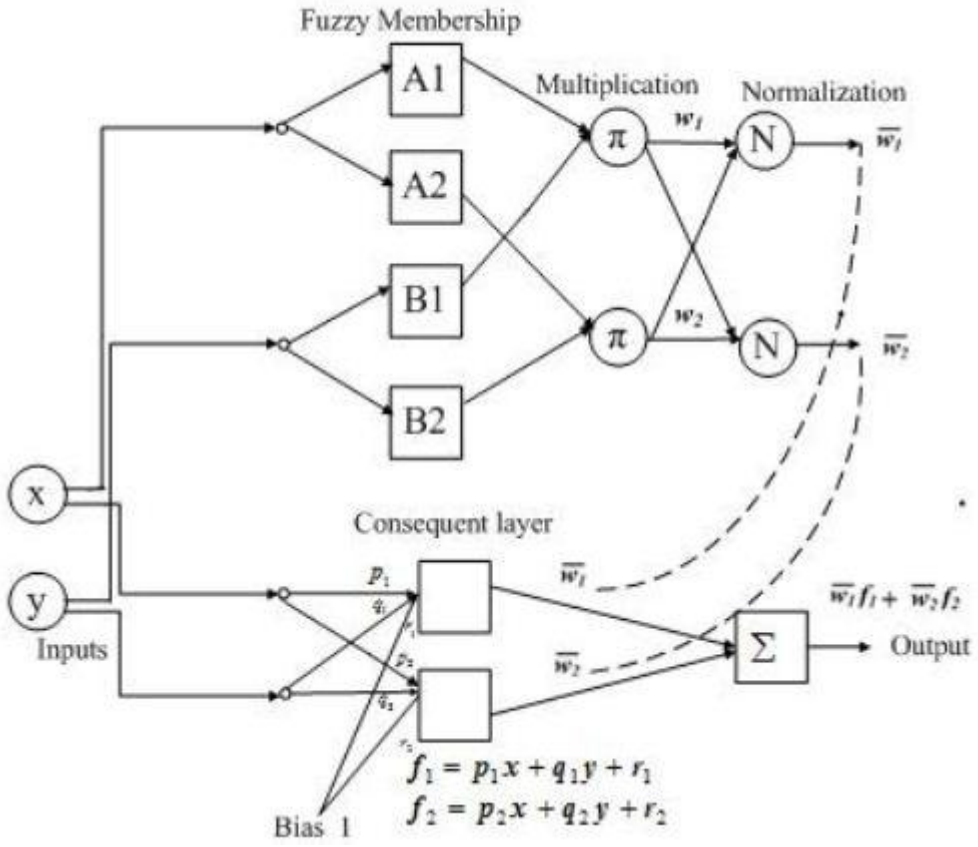

Fig. 3.3 Structure of ELANFIS[17]

In ELANFIS parameters of the fuzzy membership function i.e. for bell shape membership function center and width , are kept constant, Which hugely reduces the computational complexity of ELANFIS as they doesn't have to be calculated by

computationally expensive gradient descent method. For given input normalized firing strength is calculated from randomly chosen premise parameter. Then the consequent parameters are found out by applying Moore-penrose pseudoinverse. It had successfully been applied to control the height of water in a conical tank with the help of inverse control and model predictive control mechanism.

3.2.2 ELANFIS in DMP

As one can observe that phase of the canonical system (**Eqn. 3.5**) has a linearly increasing dynamics. In order to produce rhythmic behavior basis function has to repeat itself after certain interval. So *von mises* basis function (Eqn. 3.9) is used, which is essentially a periodic Gaussian function

$$\Psi_i = \exp(\, h_i(\cos(\Phi - \, c_i) - 1)) \qquad 3.9$$

h_i and c_i are the width and center of the i^{th} basis function. Presence of cosine function makes it repeat after 2π interval. Fig bellow shows the plot of different variable of rhythmic Dynamic System.

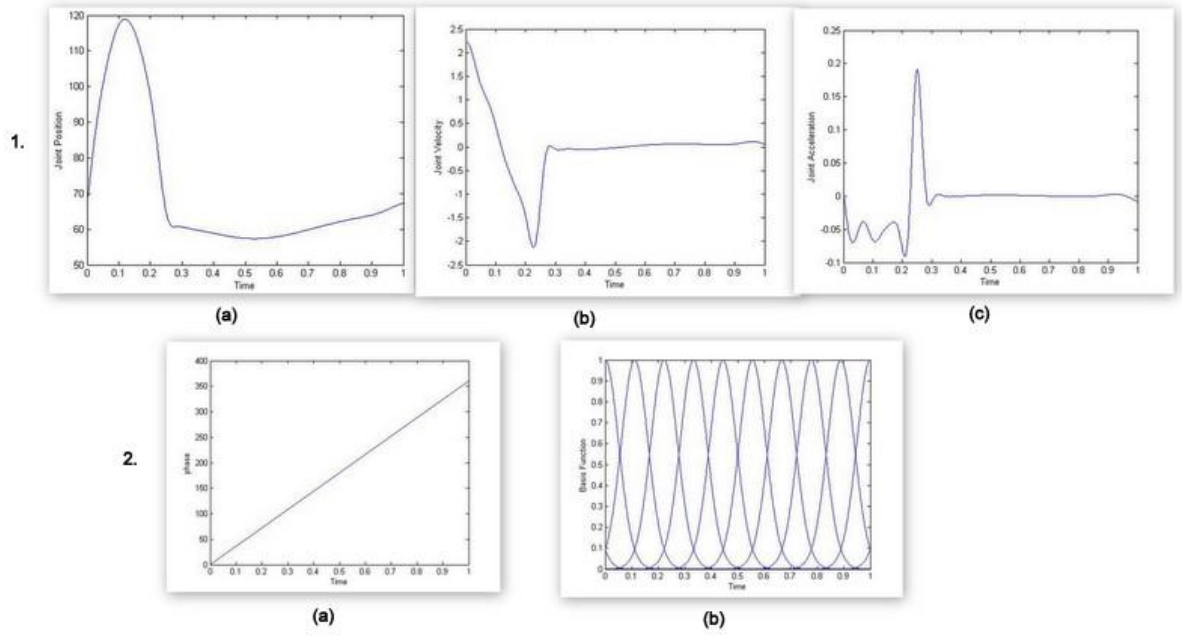

Fig. 3.4 Time evolution of joint position(1.a), velocity(1.b) and acceleration(1.c). 2.a shows the solution of phase dynamics. 2.b plots of basis function with respect to time(10 basis function per period)

Solution of the canonical system is given i/p to the function approximator and the target is generated after putting the necessary data in the **Eqn. 3.6.** After training, learned consequent parameters are used at the time of reproducing the movement.

3.3 Inter and Intra limb Co-ordination

Till now trajectory generation for a single joint or a DOF has been discussed. In order to produce co-ordination between different DOF, each DOF has to couple to each other. Co-ordination between the DOF is highly necessary in order to perform a specific task. Taking the example of locomotion task, each joint with in a leg and each leg with in themselves have to co-ordinate with each other to perform stable locomotion.

Unlike [15], where a canonical system per joint has been used, this work uses a canonical system per leg. This representation has couple of benefits, First, it reduces number of coupling term from 12(as Hexapod is having 2DOF per leg) to 6 and gives a compact representation of phase Dynamics. Second, it gives one the better way to achieve inter and intra limb coordination. As shown in the figure bellow, trajectory of each joint is taken care of by the Transformation system and by sharing a canonical system between two (in this case) transformation system intra limb co-ordination has been achieved. For inter limb co-ordination, the coupling architecture discussed in [1] has been used.

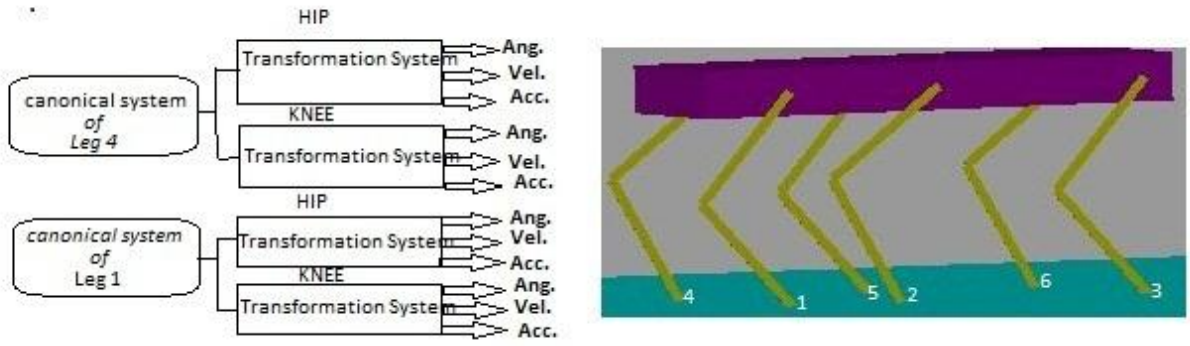

Fig. 3.5 Coupling Architecture to achieve Intra-limb co-ordination.

In order to perform a particular gait pattern each leg has to maintain certain phase difference with other. Canonical system which represents the phase of a leg has to coupled with other canonical systems in a such a manner that, certain desired phase difference among them can be maintained. In [1] E. Klavins et al. have proposed a framework in which two oscillatory systems can achieve in phase and out phase relationship depending on the connection specified between them.

A potential energy function 'V'(Eqn. 3.10) is defined on the phase difference in such a manner that, it has unique minima at π.

$$V = \cos(\Phi_1 - \Phi_2) \tag{3.10}$$

Then the phase dynamics/ reference field is added with -ve gradient of the energy function. This -ve gradient term drifts phase to its stable unique minima. The oscillator dynamics after adding with -ve gradient term is given in Eqn 3.11

$$\dot{\Phi}_i = k_1 - k_2 \sum_{j=1}^{n} c_{i,j} \sin(\Phi_i - \Phi_j) \tag{3.11}$$

Where k_1 is the natural frequency of the oscillator, k_2 is the gradient constant determines the rate of convergence of stable limit cycle. $c_{i,j}$ is the coupling matrix, which defines phase

relationship between two oscillators. If $c_{i,j}$ is set to '1' then the oscillator 'i' and 'j' are in phase, if set to '-1' then they are out of phase and '0' indicates no coupling. Phase dynamics of the canonical system is added with the -ve gradient term, which dives the complete dynamical system to have a stable limit cycle defined by the coupling matrix. Details about the coupling matrix and desired phase difference to achieve different gait pattern is given in Section 4.

Chapter 4

EXPERIMENTAL RESULT

In order to validate the control architecture discussed in chapter 3, a virtual hexapod build in Matlab's Simulink environment has been used. But model development discussed in chapter 2 is not sufficient to perform experimental work. So, few additional blocks are designed which has been discussed in this chapter. Section 4.1 discusses about generation of smooth joint trajectory from discrete joint angle command by using cubic spline method. Procedure adapted to generate joint trajectory data for different locomotion pattern has been discussed in section 4.2. Section 4.3 concludes this chapter with discussion about the experimental work.

4.1 Experimental Setup

In chapter 2 building of a Hexapod model along with ground and necessary ground reaction force has been discussed. In order to use the developed model in experiment a lower level controller has been developed. This is due to, DMP acts as a higher level controller and generates the necessary joint angle command, so system requires a lower level controller to keep track of these commands. In this work computed torque control technique [23] has been used as lower level controller.

DMP controller generates joint angle command at a rate of 100 samples per second. When these discrete angle commands are given to the Hexapod model, the "ode solver" of Matlab takes too much time to solve the dynamics of the model. In order to overcome this problem these discrete joint angle commands are made continuous with the help of cubic spline interpolation method[24].

There are several smooth function which can join initial joint angle $\theta(t_0)$ to the final joint angle $\theta(t_f)$, cubic spline is one of the simplest way to do that. In this method joint angle is represented with 3rd order polynomial of time given in **Eqn 4.1**

$$\theta(t) = a_0 + a_1 t + a_2 t^2 + a_3 t^3 \quad 4.1$$

Taking the derivative of the above equation and putting the initial conditions values, the above four coefficient can be determined.

$$\dot{\theta}(t) = a_1 + 2a_2 t + 3a_3 t^2 \quad 4.2$$

These initial conditions i.e. initial and final joint angles and joint velocities are obtained from system and controller respectively. Joint angle and velocity of each joint of the system are sampled and used as initial joint angle $\theta(t_0)$ and velocity $\theta(\dot{t}_0)$. Final joint joint angle $\theta(t_f)$ and velocity $\theta(\dot{t}_f)$ are given by the controller. For simplicity this work assumes initial and final velocity as zero. **Fig 4.1** shows the Matlab implementation of cubic spline method.

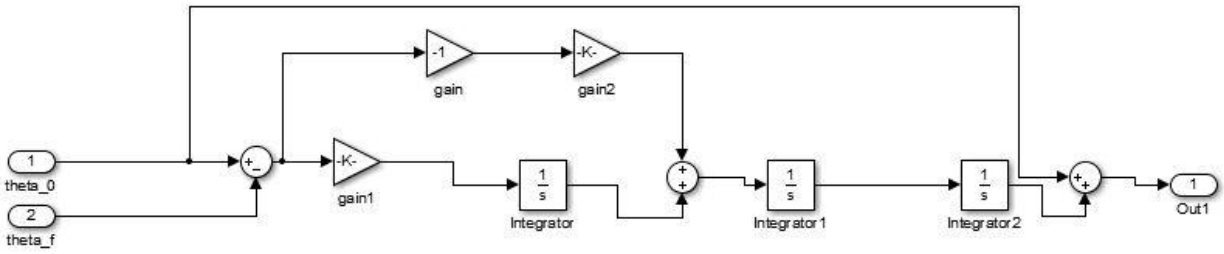

Fig. 4.1 Cubic spline interpolation method developed in Matlab to generate continuous trajectory

4.2 Generation of joint trajectory Data

Generating training data is one of the vital part while designing a controller using DMP. As discussed in chapter 3.2, various method has been used to train a DMP. Overall what is common to all these approach is that, they all are recording kinematics parameters of the movement. In this work, end effector of the leg has been made to follow the trajectory as shown in the figure 4.2. The 'x' and 'y' co-ordinates of the end effector have been obtained as given below

$x = A\cos\phi \qquad (0 \leq \phi < 2\pi)$

$y = B\sin\phi \qquad (0 \leq \phi < \pi)$

$y = B'\sin\phi \qquad (\pi \leq \phi < 2\pi)$

Where A, B, B' are positive constant which take care of stride length and height of the trajectory respectively. Trajectory in the task space have been converted into joint space with the help of inverse kinematics. And as discussed in the section 3.2, these joint trajectories are used to train the DMP.

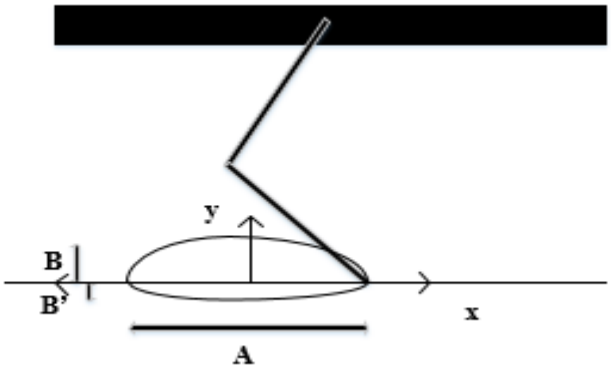

Fig. 4.2 Trajectory followed by a leg while performing locomotion

Geometrical approach for finding the inverse kinematics has been used. As the leg is planar, one can apply plane geometry to find the solution. As shown in the figure 4.3, θ_1 and θ_2 are the hip and knee angle respectively which has to be found out from x and y co-ordinate of the leg. From figure 4.3 it is clear that

$$\phi = tan^{-1}(x/y) \quad 4.3$$

and $\theta_2 = \gamma - \phi + 90^0 - \theta_1$ 4.4

applying law of cosine to ΔABC

$$\gamma = cos^{-1}(l_2/\sqrt{l_1^2 + l_2^2}) \quad 4.5$$

and $\theta_1 = 90^0 - \phi - cos^{-1}(l_1/\sqrt{l_1^2 + l_2^2})$ 4.6

putting the values obtained from equation 4.3,4.5,4.6 in equation 4.4, θ_2 can be found out.

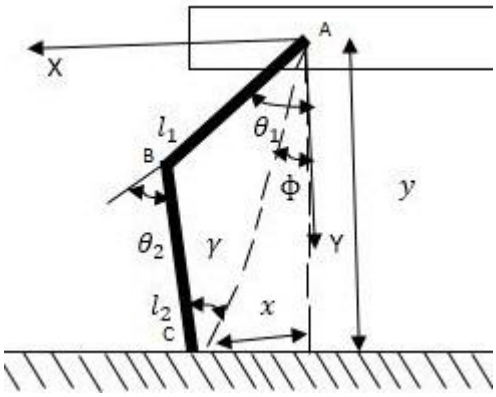

Fig. 4.3 Different parameters of Inverse kinematics

This work is intended to make the hexapod walk with three different gait patterns namely Wave, Ripple and Tripod. And it has been achieved by training DMP with different trajectories(shown in figure 4.4, a pair of trajectories for each gait). The only parameter which distinguishes different gaits is the duty factor, ratio of legs stance period to total cycle period. Tripod have lowest duty factor of 0.5, for ripple it increases to ¾ and for wave gait it is 5/6.

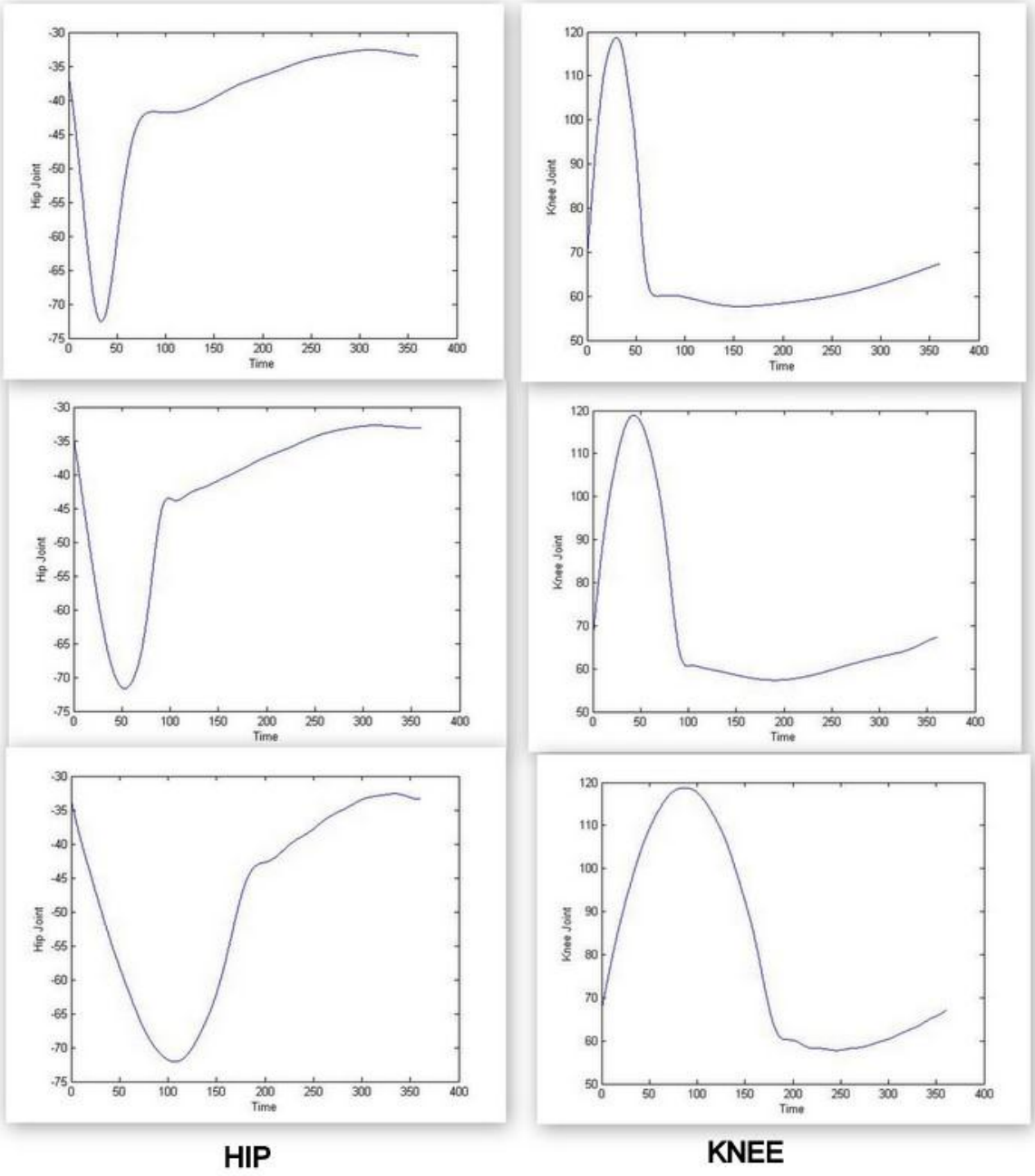

Fig. 4.4 Joint angle trajectory of Hip and Knee for Wave (Top), Ripple (middle) and Tripod(bottom)

4.3 Experiments

Each gait patterns are learned and reproduced by Hexapod independently. Learned parameters of ELANFIS are used approximate the forcing function. Transformation system augmented with these forcing function are integrated with the help of semi-implicit euler method to reproduce the learned trajectory. This integration is purely discrete in nature and trajectory samples are generated at the rate of 100 samples per second. These discrete samples are interpolated with the help of cubic spline interpolation to generate continuous like trajectory. And finally a PD controller is used to keep track of these commands.

Animals in nature produces very diverse gait pattern, for instance, insects have tripod, tetrapod, wave, transition gaits etc. Apart from these regular gaits, hexapod is found with some other irregular gaits[25] which is described as due to the probing behavior of front leg. Due to existence of complexity in analyzing these gait pattern, it attracted considerable amount of research. Various coupling methods has been proposed to model these behaviors. In this work, a very simple rule described in[3] has been followed. i.e. legs on ipsilateral side are always in phase, while legs on contralateral side of same segment of the body are out of phase with each other. For instance, fore right leg is in phase with middle right leg and hind right leg, but it is 180º out of phase with fore left leg.

This coupling architecture has been achieved with the help of the framework discussed in[1](and briefly in section 3.3 of this thesis). By specifying values of coupling parameters to '1' or '-1', two oscillators can be made in phase or out of phase. Phase of each leg is represented by a simple uncoupled phase oscillator (Eqn. 4.6). Then these oscillators are coupled with the -ve gradient term as given in Eqn. 4.7. Here the subscript 'i' represents the leg number. And the hexapod legs are numbered in such a fashion that leg number increases from front to back while starting from fore left i.e., 1 correspond to fore left leg, 2 for middle left etc.

$$\tau\dot{\phi} = 1 \tag{4.6}$$

$$\dot{\phi}_i = \frac{1}{\tau} - k \sum_{j=1}^{6} c_{i,j} \sin(\phi_i - \phi_j) \tag{4.7}$$

'c' is the coupling matrix, which defines phase relationship between two oscillator and is given in Eqn. 4.8. Each row and column represents the corresponding leg number. There is no self-coupling exist between a oscillator so, all the diagonal element of the coupling matrix are set to zero.

$$c = \begin{bmatrix} 0 & 1 & 1 & -1 & -1 & -1 \\ 1 & 0 & 1 & -1 & -1 & -1 \\ 1 & 1 & 0 & -1 & -1 & -1 \\ -1 & -1 & -1 & 0 & 1 & 1 \\ -1 & -1 & -1 & 1 & 0 & 1 \\ -1 & -1 & -1 & 1 & 1 & 0 \end{bmatrix} \tag{4.8}$$

As discussed coupling helps in achieving only 0^0 or 180^0. But in order to perform different gait pattern, each oscillator has to attain certain phase difference other than 0^0 or 180^0. This has been achieved by creating a pseudo phase variable Φ' given in eqn. bellow

$$\Phi' = \Phi + \alpha \tag{4.9}$$

Now this Φ' is added to the phase dynamics in place of Φ, so Eqn. 4.7 modifies to

$$\dot{\phi}_i = \frac{1}{\tau} - k \sum_{j=1}^{6} c_{i,j} \sin(\phi'_i - \phi'_j) \tag{4.10}$$

When $\Phi = \Phi'$, all the phase variable behave according to they have coupled i.e. Φ_1, Φ_2, Φ_3 and Φ_4, Φ_5, Φ_6 forms two group, all the oscillator inside a group are in phase with each other but, they are out of phase with the oscillator outside of the group as shown in the figure 4.5(before 10 sec.). At the moment some α is added(at t=10sec.), that particular pseudo phase variable(Φ'_2in this case) suddenly shifts from stable phase to a unstable one. According to the designed dynamics negative gradient term will try to bring the pseudo phase variable to stable

one. But as per the eqn. 4.9, the real phase Φ_2 will maintain 'α' phase difference and shifts from the group by the same phase difference.

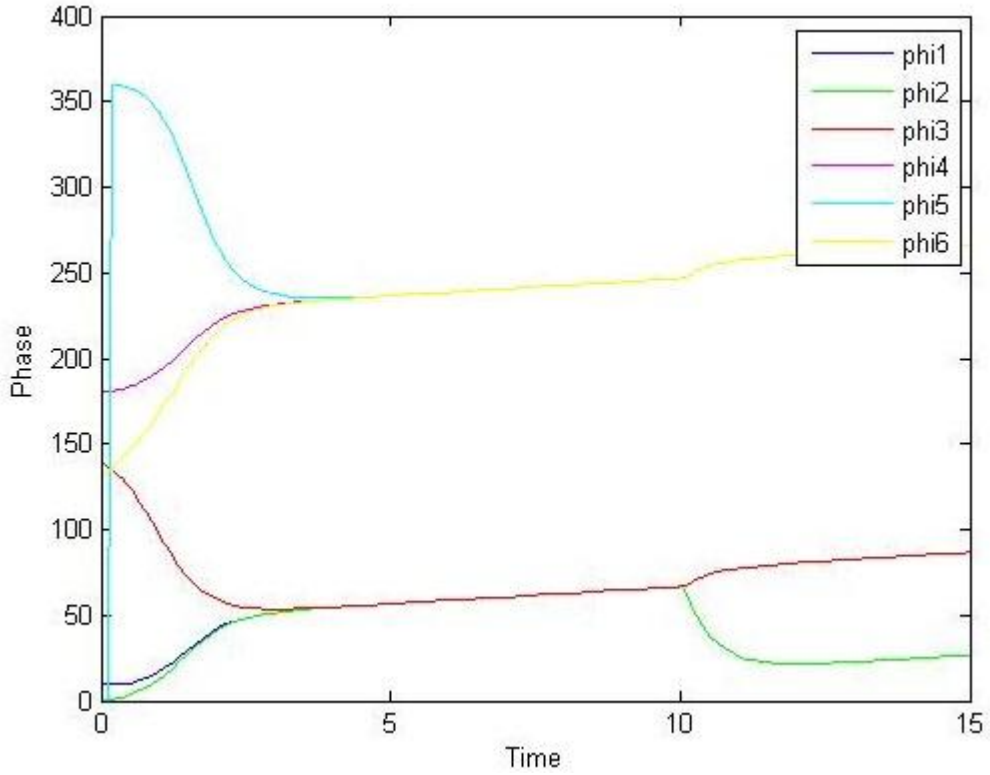

Fig. 4.5 Plot of Phase dynamics(Eqn. 4.10) with and without α

Phases starts from different initial condition and converges to the stable value around t = 4sec. A keen observer must have noticed that, there is a certain deviation in the phase of Φ_4, Φ_5, Φ_6 group and . Φ_1, and . Φ_3 as well at t = 10 (at the time of adding nonzero α). This is due to, when two homogeneous oscillators are coupled to each other it can be observed as a virtual spring connected between them [26]. When pseudo phase tries to stabilize itself it affects phase of all other oscillators. But its effect is very weak as majority of oscillators are already stable and combine to form a strong group.

In order to perform different gait pattern different values of α is added the different oscillator which is given in Table bellow. It should be noted that, coupling matrix is already taken care of the in and out of phase relationship between the legs, so Table 4.1 lists only additional phase (α) that has to be added to ϕ in order to perform different locomotion pattern.

	ϕ_1	ϕ_2	ϕ_3	ϕ_4	ϕ_5	ϕ_6
Wave	0	$\pi/3$	$2\pi/3$	0	$\pi/3$	$2\pi/3$
Ripple	0	$\pi/2$	π	0	$\pi/2$	π
Tripod	0	π	0	0	π	0

Table 4.1. Additional phase required to perform different gaits

Desired gait patterns are shown in the Fig. 4.6, Where shaded area represents legs are in stance phase and light area for swing.

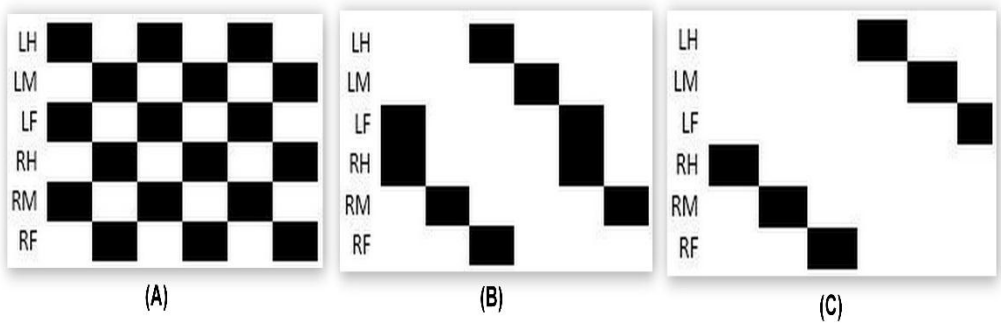

Fig. 4.6 Swing and Stance Phase of each leg for Tripod gait (A), Ripple gait (B) and Metacronal Wave gait (C). Here each leg is represented by its initial. LH for Left Hind leg, LM for Left Middle leg etc.

First joint trajectory for ripple gait is obtained by using inverse kinematics method. Then with the help of ELANFIS non-linear forcing function is learned. This completes the learning phase of DMP. At the time of reproducing these learned parameters are used to approximate the non-linear forcing function, which in turn augments the Transformation System. After integrating the Transformation System and Canonical System learned trajectory is reproduced. Similar procedure is used to perform the remaining two gaits. And Fig. 4.7 shows hexapod performing above mentioned three gait patterns.

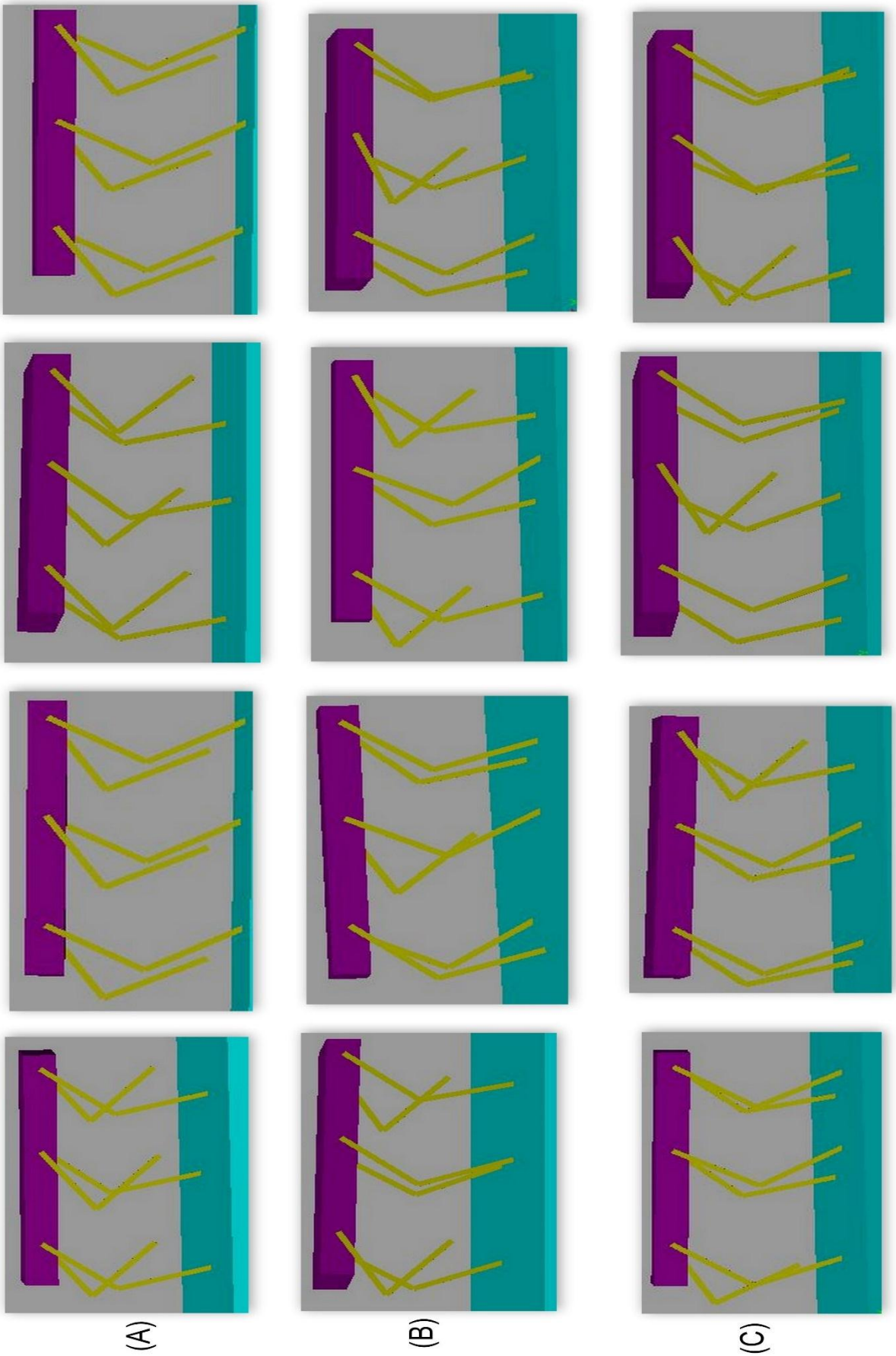

Fig. 4.7 Hexapod Robot Performing Different Gait Patterns. A. Tripod Gait, B. Ripple Gait, C. Metacronal Wave Gait.

Chapter 5

CONCLUSION AND FUTURE WORK

In this thesis, DMP is used as a CPG to generate different gait patterns in a virtual Hexapod robot build in Matlab's Simulation environment, SimMechanics. Details about SimMechanics and building of multibody structure in it's environment has been discussed. And with the help of a virtual spring damper system ground reaction force has been modeled. A brief overview of Rhythmic DMP and use of ELANFIS as function approximator in DMPs frame work has been given. In the end a coupling framework is discussed to achieve inter leg co-ordination. With the help of coupling Hexapod is able to perform three different gait pattern, wave, ripple and tripod.

In this work, Hexapod is able to perform three different gait patterns but in three independent experiments. This work can be extended to perform all these gait pattern in a single run and show a stable transition between gaits.

Publications

1. Janmejaya Nanda and Indra Gupta, "Learning Locomotion in Hexapod using Dynamic Movement Primitive" in IEEE International conference on power electronics, intelligent control and Energy systems to be hold at Delhi Technical University in july 2016.(communicated)

Bibliography

[1] E. Klavins and D. E. Koditschek, “Phase Regulation of Decentralized Cyclic Robotic Systems,” *Int. J. Rob. Res.*, vol. 21, no. 3, pp. 257–275, 2002.

[2] A. J. Ijspeert, “Central pattern generators for locomotion control in animals and robots: A review,” *Neural Networks*, vol. 21, no. 4, pp. 642–653, 2008.

[3] W. Chen, G. Ren, J. Zhang, and J. Wang, “Smooth transition between different gaits of a hexapod robot via a central pattern generators algorithm,” *J. Intell. Robot. Syst. Theory Appl.*, vol. 67, no. 3–4, pp. 255–270, 2012.

[4] K. Tsujitat, K. Tsuchiyat, and A. Onatt, “Adaptive Gait Pattern Control of a Quadruped Locomotion Robot,” in *IEEE/RJS International Conference on Intelligent Robots and Systems*, 2001, pp. 2318–2325.

[5] R. Campos, V. Matos, and C. Santos, “Hexapod locomotion: A nonlinear dynamical systems approach,” in *IECON 2010 - 36th Annual Conference on IEEE Industrial Electronics Society*, 2010, pp. 1546–1551.

[6] L. Righetti and A. J. Ijspeert, “Pattern generators with sensory feedback for the control of quadruped locomotion,” in *Proceedings - IEEE International Conference on Robotics and Automation*, 2008, pp. 819–824.

[7] P. Arena, L. Fortuna, and M. Frasca, “Multi-template approach to realize central pattern generators for artificial locomotion control,” *Int. J. Circuit Theory Appl.*, vol. 30, no. 4, pp. 441–458, 2002.

[8] T. Flash and B. Hochner, “Motor primitives in vertebrates and invertebrates,” *Curr. Opin. Neurobiol.*, vol. 15, no. 6, pp. 660–666, 2005.

[9] A. J. Ijspeert, J. Nakanishi, and S. Schaal, “Learning Attractor Landscapes for Learning Motor Primitives,” in *Advances in Neural Information Processing Systems 15 (NIPS2002)*, 2002, pp. 1547–1554.

[10] D. Pongas, A. Billard, and S. Schaal, “Rapid synchronization and accurate phase-locking of rhythmic motor primitives,” in *2005 IEEE/RSJ International Conference on Intelligent Robots and Systems, IROS*, 2005, pp. 1917–1922.

[11] J. Kober and J. Peters, “Policy Search for Motor Primitives in Robotics,” in *Advances in Neural Information Processing Systems 21*, 2009, pp. 849–856.

[12] J. Kober, “Reinforcement Learning for Motor Primitives,” 2008.

[13] E. Theodorou, J. Buchli, and S. Schaal, “A Generalized Path Integral Control Approach to Reinforcement Learning,” *J. Mach. Learn. Res.*, vol. 11, pp. 3137–3181, 2010.

[14] J. Nakanishi, J. Morimoto, G. Endo, G. Cheng, S. Schaal, and M. Kawato, “Learning from demonstration and adaptation of biped locomotion,” *Rob. Auton. Syst.*, vol. 47, no. 2–3, pp. 79–91, 2004.

[15] M. K. JUN NAKANISHI,JUN MORIMOTO,GEN ENDO,GORDON CHENG,STEFAN SCHAAL, “A FRAMEWORK FOR LEARNING BIPED LOCOMOTION WITH DYNAMICAL MOVEMENT PRIMITIVES,” in *2004 4th IEEE/RAS International Conference on Humanoid Robots*, 2004, pp. 925 – 940.

[16] S. Schaal and C. Atkeson, “Constructive incremental learning from only local information,” *Neural Comput.*, vol. 10, no. 8, pp. 2047–84, 1998.

[17] G. N. Pillai, J. Pushpak, and M. G. Nisha, “Extreme learning ANFIS for control applications,” in *2014: 2014 IEEE Symposium on Computational Intelligence in Control and Automation, Proceedings*, 2014, pp. 1–8.

[18] A. J. Ijspeert, J. Nakanishi, H. Hoffmann, P. Pastor, and S. Schaal, “Dynamical movement primitives: learning attractor models for motor behaviors.,” *Neural Comput.*, vol. 25, no. 2, pp. 328–73, 2013.

[19] J. Kober, B. Mohler, and J. Peters, “Learning perceptual coupling for motor primitives,” in *2008 IEEE/RSJ International Conference on Intelligent Robots and Systems, IROS*, 2008, pp. 834–839.

[20] J. Kober, J. A. Bagnell, and J. Peters, “Reinforcement learning in robotics : A Survey,” *Int. J. Rob. Res.*, vol. 32, no. 11, pp. 1238–1274, 2013.

[21] J.-S. R. Jang, “ANFIS : Adap tive-Network-Based Fuzzy Inference System,” *IEEE Trans. Syst. Man. Cybern.*, vol. 23, no. 3, pp. 665–685, 1993.

[22] G.-B. Huang, Q. Zhu, C. Siew, G. H. Ã, Q. Zhu, C. Siew, G.-B. Huang, Q. Zhu, and C. Siew, “Extreme learning machine: Theory and applications,” *Neurocomputing*, vol. 70, no. 1–3, pp. 489–501, 2006.

[23] C. T. A. L.Lewis, Frank, Darren M.Dawson, “Manipulator Control Theory and Practice,” in *Marcel Dekker, Inc.*, 2004, p. 607.

[24] J. J. Craig, “Introduction to Robotics,” in *Pearson Education International*, vol. 1, no. 2, 1986, pp. 1–31.

[25] M. Grabowska, E. Godlewska, J. Schmidt, and S. Daun-Gruhn, “Quadrupedal gaits in hexapod animals - inter-leg coordination in free-walking adult stick insects,” *J. Exp. Biol.*, pp. 4255–4266, 2012.

[26] F. Dörfler and F. Bullo, “Synchronization in complex networks of phase oscillators: A survey,” *Automatica*, vol. 50, no. 6, pp. 1539–1564, 2014.