Feedback Error Learning for Gait Acquisition

Master-Thesis von Nakul Gopalan
November 2012

TECHNISCHE
UNIVERSITÄT
DARMSTADT

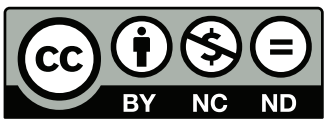
Fachbereiche ETIT und Informatik
Intelligent Autonomous Systems

Feedback Error Learning for Gait Acquisition

Vorgelegte Master-Thesis von Nakul Gopalan

1. Gutachten: Jan Peters
2. Gutachten: Klaus Hofmann

Tag der Einreichung:



Erklärung zur Master-Thesis

Hiermit versichere ich, die vorliegende Master-Thesis ohne Hilfe Dritter nur mit den angegebenen Quellen und Hilfsmitteln angefertigt zu haben. Alle Stellen, die aus Quellen entnommen wurden, sind als solche kenntlich gemacht. Diese Arbeit hat in gleicher oder ähnlicher Form noch keiner Prüfungsbehörde vorgelegen.

Darmstadt, den November 6, 2012

(N. Gopalan)

Acknowledgements

Firstly, I would like to thank my supervisors Prof. Jan Peters and Prof. Klaus Hofmann.

Jan has helped me realize importance of presentation skills, and taught me most of the basics of robotics that I had not even known a year back.

Prof. Hofmann has been really kind in accepting me as an internal thesis student at the last minute, and putting up with my mails.

I would like to thank Dr. Marc Deisenroth. Marc has put up with the most absolute worst of my presentation skills and helped me with innumerous drafts, apart from helping me on the theoretical problems.

I would also like to thank the members of Intelligent Autonomous Systems Lab, they have been most kind, and helpful, and the best of friends.

I would also like to thank my friend Sriram Prabhu Kumble who also read these drafts.

Last but not the least, I would like to thank my Mother, my Father and my Brother who have always supported me with my decisions and put up with my antics.

Abstract

Robots execute a wide array of rhythmic behaviors like walking, running, dribbling, etc. Rhythmic movement primitives are parametric models that can be used to represent these behaviors. Previously, these models have been learned either by means of demonstrations or by means of reward signals. While executing a rhythmic behavior, a stabilizing controller is needed in most robots to prevent failure. An example for this controller can be that of a balance controller, needed to keep the torso of a biped upright while walking. The stabilizing controller uses a feedback mechanism to maintain stability. Learning a balanced forward trajectory or gait in case of a biped, which mutes the stabilizing controller would avoid the use of high gain feedbacks, and instead use the feedbacks only in case of emergencies.

We propose to learn the balanced forward trajectory, i.e., its movement primitive using feedback error learning. Feedback error learning is a model-learning method that we modify to learn rhythmic movement primitives using the feedback of the stabilizing controller. This modification, along with the architecture of feedback error learning allows us to learn new trajectories without complicated reward functions, or training examples as are needed by reinforcement learning or supervised learning respectively. Moreover, this method is online. Thus, the trajectories can be learned by the system while executing them stably. We test our novel method, using simulations on two problems: forward trajectory learning in a two link manipulator and gait adaptation by a biped robot to confirm the method's efficacy in learning trajectories using feedback errors.

Contents

1 Introduction

1.1 Learning Rhythmic Behaviors

Animals exhibit many rhythmic behaviors like walking, running, swimming, etc. These behaviors are learned and developed by animals over their life spans for sustenance. We would like robots too to acquire and develop skills like animals, instead of needing handcoding or programming actions. Anthropomorphic robots would need to learn skills exhibited by humans and other animals, including the rhythmic behaviors discussed previously. This makes learning rhythmic behavior an important problem in the field of motor skill acquisition.

Ijspeert developed the idea of parametrized dynamical systems that can be used to model, and generate trajectories [1] using supervised learning methods. These models are also used to learn rhythmic behaviors using supervised learning of demonstrations and are called *Rhythmic movement primitives*. Subsequently, reinforcement learning techniques [2, 3] can be used for improving the behavior or trajectory with respect to a cost function. In recent past, skills like ball paddling [3], drumming [4], walking, and running [5, 6] have been acquired using these methods. We also use rhythmic movement primitives to model our trajectory, in this thesis.

A common concept observed in most rhythmic behaviors is the presence of a balancing component, e.g., humans can walk while carrying a cup of water in our hands. This balancing component ensures that when body parts like hands and legs are performing a rhythmic behavior, the other body parts like the head and the torso are stable so as to perform other tasks like watching a target or carrying an object safely. This balancing can be observed in Fig. 1.1, where a man is shown running in a series of pictures. The torso in the photographs can be seen to be at the same angle throughout the run, i.e., the torso maintains balance by following a trajectory with respect to the limbs of the man, such that he does not fall over. This stability is a dynamic equilibrium and different torso angles would be found for different running speeds of the man. While walking for example the torso might be almost upright. If we learned only the rhythmic behavior and ignored learning its balancing component, we would get solutions that are not optimal, i.e., the robots would need high gain feedbacks to maintain balance if they did not learn the balancing component. Large feedback gains are bad for stability of the whole system as even a small disturbance can cause the system to provide large torques. Consequently, the robots become unsafe for humans and expend more energy.

Instead, if we learn the balancing component we generate rhythmic behavior with balance, i.e., the head and torso will receive balancing rhythmic trajectory and the arms and limbs will receive the rhythmic trajectories previously learned,

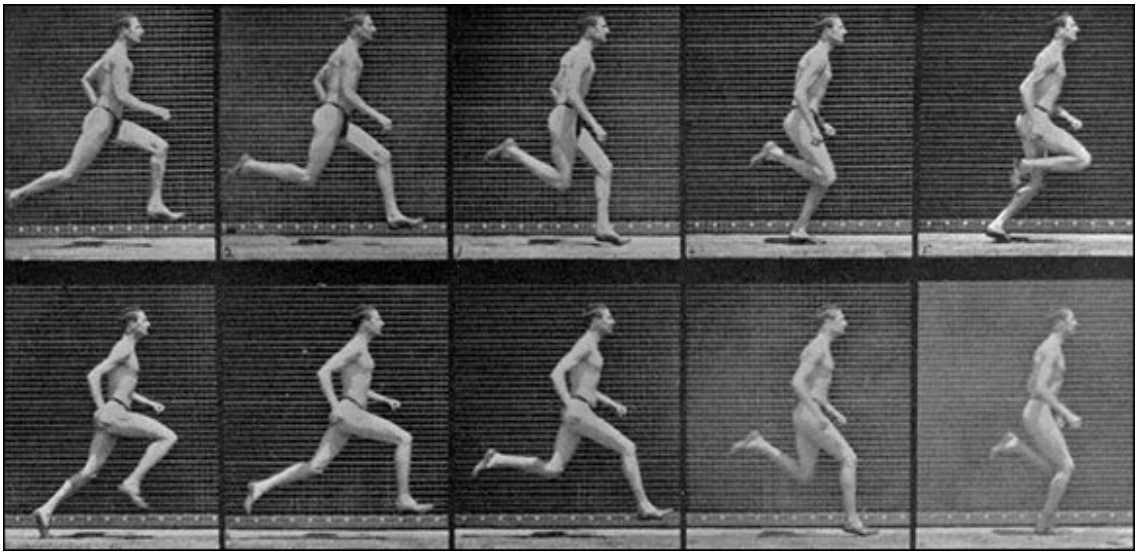


Figure 1.1.: Photo series taken by Eadweard Muybridge to show a man running, it can be noticed that the man's torso is balanced at the same angle through the run, while the arms and legs are moving in a periodic fashion. This means the torso has a joint trajectory w.r.t. the legs which keeps it stable. Photo from `http://www.gabrielsolomon.ro`

performing the whole rhythmic behavior optimally. In the absence of large balancing feedbacks we can use cheaper and lighter motors, lower sampling rates, and less electronics, while performing the desired behavior. Once the rhythmic behavior is learned with the balancing component, the purpose of the stabilizing controller would only be to function in case of emergencies, when the robot is completely out of balance because of external factors.

Nevertheless, most common approaches [1, 3] ignore this additional constraint of balance learning, despite the fact that important tasks, such as hopping, walking, and running of both robot or animals, all require a balance controller in addition to their rhythmic behavior (e.g., their gaits). The objective of this thesis is to develop a method to learn the balancing forward trajectory of a robot, such that the stabilizing controller falls silent after learning, while executing the learned rhythmic behavior.

Learning of the balancing trajectory to reduce the stabilizing feedback appears similar to a reinforcement learning problem, as value function methods can be used to reduce the feedback, but it has a much simpler solution in the form of *feedback error learning* [7]. Feedback error learning is a model learning method similar to supervised learning. In feedback error learning the feedback from a system is used as an error signal to learn the open parameters of the required model. Kawato originally used the model for learning inverse dynamics models. In our case the open parameters would be the parameters of the Rhythmic movement primitives that we use to model trajectories, and not an inverse model. We assume the inverse model to be already known. This difference is novel to our method, i.e., we learn the model of the trajectory and not the inverse model. Hence, we

adapt the trajectory generator to generate stable trajectories based on feedbacks given by the stabilizing controller or a control law.

Feedback error learning of rhythmic motor primitives has substantial advantages over both imitation and reinforcement learning. Unlike classical imitation learning, it does not require a well-demonstrated behavior where the balance controller already remains silent; however, it can be used in conjunction when a (potentially bad) demonstration is used for the initialization of the rhythmic motor primitives. Nevertheless, it still results in a supervised learning problem, it is hence a generically easier problem than reinforcement learning. Furthermore, learning can be achieved online and on-policy. As a result, it can even be used to adapt a behavior to a non-stationary environment, e.g., adapt a walking gait to a novel terrain.

We evaluate this insight and the resulting learning architecture in two scenarios, i.e., firstly, a toy problem with a two-link robot arm, and, secondly, a planar biped walking robot with a torso. Both evaluations illustrate the applicability of this novel way of learning rhythmic behavior.

The structure of this thesis is as follows, Chapter 2 describes the required background information used in the development of this thesis, Chapter 3 discusses the development of the modified learning method and experiments conducted on the method. Further a conclusion and some ideas about future work is discussed in Chapter 4.

2 Background

This chapter details the background concepts utilized in this thesis. First, we discuss representing trajectories using movement primitives. Subsequently we describe the architecture that we use for our learning problem, i.e. Feedback Error Learning. Finally we take a look at the problem of Legged locomotion in robotics.

2.1 Rhythmic Motor Primitives

Animals and humans perform actions in their everyday life. These actions are performed by trajectories that are followed by the limbs or other body parts, over the period of the action. Robots need to perform actions too, either to imitate anthropomorphic actions or to perform specific tasks. The trajectories for these actions need to be modeled, such that these trajectories can be given as motor commands to generate torques. For example, to teach a robot to kick a ball, we can teach it the trajectory to be followed by each of its joints, which can then be converted to motor commands. We need model these trajectories so that they can be reproducible and parameterizable.

2.1.1 Trajectory Modeling

Previously, trajectory modeling has been done by hand-coding or by using splines. Hand-coded trajectories cannot be parameterized even for small changes like changes in the amplitude or period of the trajectory, and, hence, their use is avoided. Splines are functions made piece-wise from polynomial functions, and are used to fit smooth trajectories. Splines can be learned using machine learning methods, but they do not satisfy the need to be inherently stable and are not suitable for online learning [1].

In this work, we use Movement Primitives [5], because apart from giving smooth, learnable trajectories, Movement Primitives are parameterizable, robust to perturbations, and provide goal directed behavior.

Movement Primitives [5] are a framework to model kinematic motor behaviors in robots. In the context of this thesis, Movement Primitives model the joint trajectories that the robot follows. Movement Primitives must provide stable trajectories and must be easy to parameterize, hence Schaal et al. [5] use dynamical systems to model trajectories. Dynamical systems have an attractor behavior that can be modulated using nonlinear terms to get the *desired* attractor behavior, i.e., stable trajectories form a baseline that can be modulated using nonlinear terms to get desired trajectories that are both precise and stable. The parameters to learn the model are the nonlinear terms that modulate the attractor behavior to get the desired behavior.

2.1.2 Biological inspiration for movement primitives

Movement primitives have biological inspiration from motor behavior models called motor primitives. Mussa-Ivaldi [8] models muscular movement in frogs with vectorial superposition of force fields in neurons. This can be seen as an inspiration to use sum of nonlinear terms to generate trajectories. Also Schöner et al. [9, 10] modeled motor behaviors with dynamical systems. They lead to the inspiration of using adjustable attractor landscapes to learn motor primitives in [1].

2.1.3 Movement primitives and attractor landscapes

Movement Primitives have two flavors: *Dynamic* and *Rhythmic Movement Primitives*. The distinction between them is based upon the type of attractor the dynamical systems use. An attractor can be defined as "a set of points to which a dynamical system converges from its many choices of initial points" [11]. If the attractor set is a single point, the attractor is called a *point attractor*, and if the attractor set is a closed trajectory which repeats itself over time, it is called a *limit cycle*. The use of attractors to model equations of trajectories gives the important property of stability to the model, to ensure that the trajectory always settles down to its goal state or its baseline trajectory.

Dynamic Movement Primitives (DMPs) use a point attractor in their dynamical system equation, i.e., they represent actions that have a goal state at which the trajectory ends. Examples of trajectories that can be modeled with DMPs can be an action to throw a ball, or an action to hit a moving ball. *Rhythmic Movement Primitives* (RMPs) use limit cycles in their dynamical system equations to model trajectories. Limit cycles are periodic in nature and allow modeling of periodic trajectories. Examples of trajectories that can be modeled with RMPs can be walking, jumping, masticating food, swimming etc. Since this thesis aims to learn rhythmic actions like walking, we discuss RMPs and its formulation next.

2.1.4 Development of rhythmic movement primitives

A simple and well understood second order dynamics equation — a damped spring model, was chosen by Schaal et al. [1, 5] to model the trajectories. In this thesis, the trajectories being modeled are the joint angles, hence, the equations are formulated with joint angles. RMPs equation formulation have a *canonical system*, which removes the dependence on time of the main dynamics equation, hence, making the system autonomous. The canonical system models the phase of the system as a variable of time, i.e.,

$$\tau \dot{\phi} = 1, \tag{2.1}$$

this allows the system to be parameterized with respect to the time period of the rhythmic trajectory τ [5]. The variable ϕ is the phase of the system, and $\dot{\phi}$ is

its first order time derivative. The second order dynamical equation that models the trajectory is called the *transformation system* as it transforms the dynamical equation's limit cycle to a desired trajectory. The transformation system can be represented as

$$\tau^2 \ddot{\boldsymbol{q}} = \underbrace{\alpha_z(\beta_z(\boldsymbol{g}-\boldsymbol{q})-\tau\dot{\boldsymbol{q}})}_{\text{Attractor function}} + \underbrace{\boldsymbol{\Theta}\boldsymbol{\psi} r}_{\text{Forcing function}}, \tag{2.2}$$

where $\boldsymbol{q}, \dot{\boldsymbol{q}}$ and $\ddot{\boldsymbol{q}}$ are the joint angles of robot and the first and second order derivatives of the joint angles w.r.t. time. Parameters α_z and β_z are timing constants. The time period of the rhythmic action is as mentioned before, represented by τ. The parameter $\boldsymbol{g}$ is the baseline of the rhythmic trajectory. The weight vectors are represented by $\boldsymbol{\Theta}$, and $\boldsymbol{\psi}$ are the nonlinear basis functions. The product $\boldsymbol{\Theta}\boldsymbol{\psi} r$ is called the forcing function, as it modulates the landscape of the attractor, learning $\boldsymbol{\Theta}$ implies learning the forcing function. The nonlinear basis functions are von Mises basis functions defined upon the canonical systems output defined in Eq. (2.1), given by

$$\boldsymbol{\psi} = \exp\big(\boldsymbol{h}\big(\cos\big(\boldsymbol{\phi}-\boldsymbol{c}\big)-1\big)\big), \tag{2.3}$$

where $\boldsymbol{c}$ and $\boldsymbol{h}$ describe the positions and widths respectively of the raster of von Mises functions defined over the space of the canonical system $\boldsymbol{\phi}$. RMPs are based on dynamical systems that are critically damped, i.e., the dynamical systems have a tendency to return to their equilibrium state if not energy is input to them. Therefore, the trajectories generated using them are stable as they do not move away from the equilibrium, if given stable inputs. We now have a mathematical model for the RMPs, next we look at training this model to a desired trajectory.

2.1.5 Training movement primitives

In [6, 12] imitation learning is used to learn an RMP trajectory model. In imitation learning, a desired trajectory for the joint angle, $\boldsymbol{q}_d$ with its first and second order derivatives $\dot{\boldsymbol{q}}_d$ and $\ddot{\boldsymbol{q}}_d$ can be recorded and given as supervised learning examples to learn the forcing function. The cost function for learning the forcing function can be formulated as the difference between the desired trajectory and the modeled trajectory over N measurements as,

$$E = \frac{1}{2}\sum_{k=1}^{N} \Big\| \underbrace{\tau^2 \ddot{\boldsymbol{q}}_{\mathrm{d}}^{k} - (\alpha_z(\beta_z(\boldsymbol{g}-\boldsymbol{q}_{\mathrm{d}}^{k}) - \tau\dot{\boldsymbol{q}}_{\mathrm{d}}^{k}))}_{\text{Demonstrated Trajectory}} - \underbrace{\boldsymbol{\Theta}\boldsymbol{\psi} r}_{\text{Forcing Function}} \Big\|^2, \tag{2.4}$$

from Eq. (2.2). This cost function needs to minimized by learning the weight vectors $\boldsymbol{\Theta}$, which in turn minimizes the difference between the modeled trajectory and desired trajectory, thereby learning the forcing function. The weight vectors $\boldsymbol{\Theta}$ can be learned using single-step solutions like the method of least squares regression or by using multi-step solutions like gradient descent [5]. The period

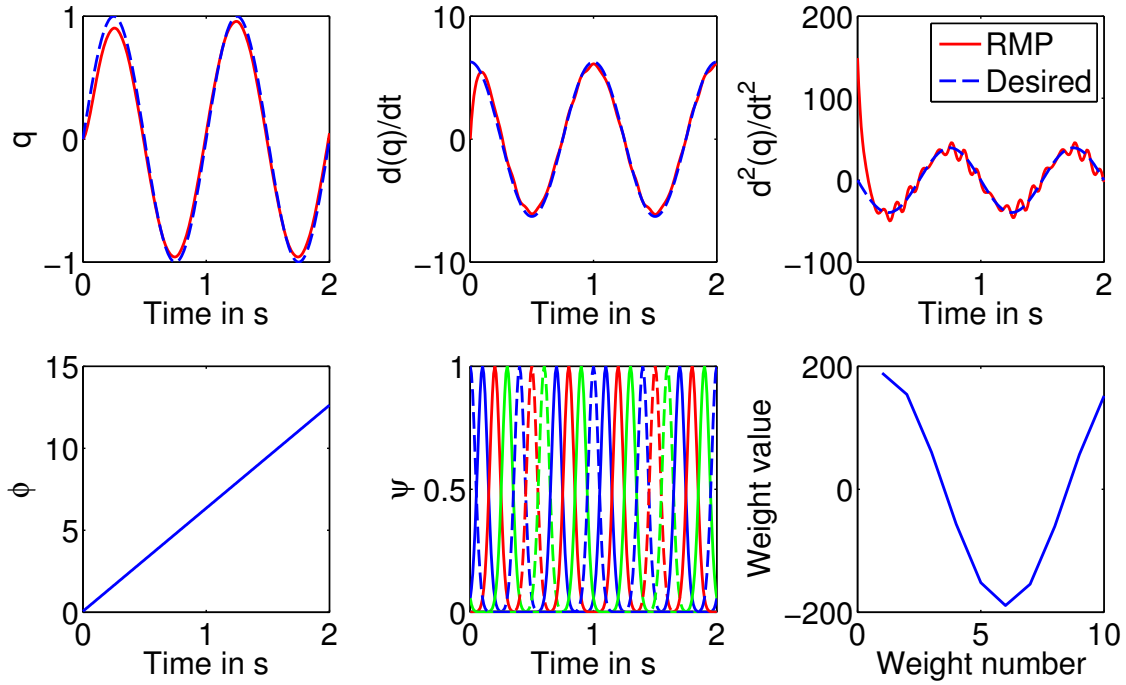


Figure 2.1.: Example trajectory generated by the RMP over two cycles. The parameters have been trained to a desired trajectory of $q(t) = \sin(2\pi t)$. The first row, from left to right, shows the trajectory generated $q(t)$ along with its first and second order derivatives $\dot{q}(t)$ and $\ddot{q}(t)$. The legend of the $\ddot{q}(t)$ subfigure applies to all the subfigures in the first row. The second row, from left to right, shows phase variable ϕ generated by the canonical system, the basis functions ψ, which are 10 per cycle, and the 10 weights trained for these basis functions for each cycle. It can be seen from the first row that the trajectory generated by the RMP overlaps with that of the desired trajectory completely.

of the rhythmic trajectory needs to be evaluated in advance using Fourier analysis [5, 12]. The time constants of α_z and β_z are chosen such that the system is critically damped, i.e., the dynamical system returns to an equilibrium at the end of the action and does not show oscillations. The goal parameter for a rhythmic action can be the mean of the desired rhythmic trajectory or a value that the desired trajectory attains in each cycle. Once given all the timing parameters, the goal state and a desired trajectory the RMP model can be learned for the trajectory. Hence the only parameters that need to be learned a RMP model are the weight vectors $\boldsymbol{\theta}$ of the model.

2.1.6 Generating trajectories using RMPs

After learning the model, the new trajectories can be generated by using Eq. (2.2), where $\ddot{\boldsymbol{q}}$ is calculated with known weight vectors and $\dot{\boldsymbol{q}}$ and $\boldsymbol{q}$ are calculated by numerical integration. A plot of RMP generation is given in Fig. 2.1. It can be seen that this learned model can be easily parameterizes, we can change the

generated trajectory's amplitude by varying the value of r in Eq. (2.2) and we can change the period of the generated trajectory by changing the parameter τ in the Eq. (2.1) and Eq. (2.2). This section detailed the development of the RMP model and its advantages of stability, learning and parameterization. The RMPs model the trajectories that we want to learn. Next we look at how to learn these models.

2.2 Model Learning

Model learning is the name given to the set of learning problems where given a set of input and output data of a system, we learn a representation of the system to predict future outputs. Model learning is a broad field and has large applications in the field of robotics, where we have a robot that needs to manipulate its environment to achieve desired results.

Animals including humans manipulate their environment for everyday tasks. Robots inherently also need to manipulate their environment to be useful. Animals learn to do tasks by learning muscle control, robots also need to learn to control their actuators for tasks. To control a robot we need to know its kinematics model and its dynamical model. The dynamical model gives the relation between applied joint torques and output joint trajectories. The kinematic model gives the relationship between joint angles and the robot end effector's Cartesian coordinates. Previously, these models were calculated using physics-based modeling techniques or hand-crafting them [13]. These calculated models are generally imprecise, as they are can make assumptions for non-linearities. Moreover, hand crafting models requires a lot of effort as well. Instead with machine learning methods these models can be learned precisely based on the observed data, and without as much effort [13]. Thus, machine learning methods are important in model learning for robot control.

A dynamical system can be formulated as follows [13]

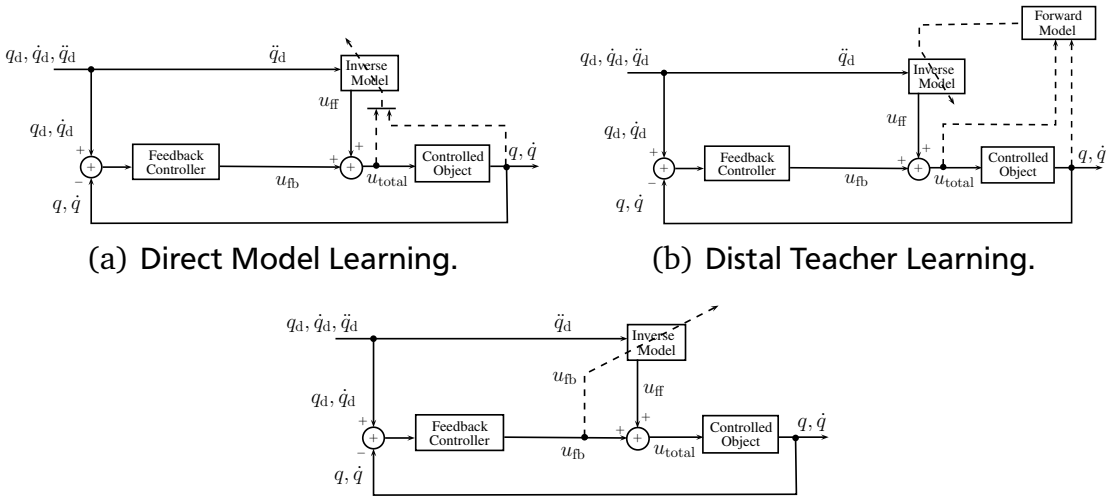
$$\boldsymbol{s}_{k+1} = f\left(\boldsymbol{s}_k, \boldsymbol{a}_k\right) + \boldsymbol{\epsilon}_f, \tag{2.5}$$

$$\boldsymbol{y}_k = h\left(\boldsymbol{s}_k, \boldsymbol{a}_k\right) + \boldsymbol{\epsilon}_y, \tag{2.6}$$

where $\boldsymbol{s}_k$ and $\boldsymbol{a}_k$ are the state of the system and action taken at time k, $\boldsymbol{y}_k$ is the output of the system at time k, f and h represent the state transition and measurement function, and $\boldsymbol{\epsilon}_f$ and $\boldsymbol{\epsilon}_h$ are the noise components which are generally assumed Gaussian [13]. Given such a system the following types of models can be formulated as described in [13]:

- *Forward Models*: These predict the next state $\boldsymbol{s}_{k+1}$ and current output $\boldsymbol{y}_k$ given the previous output or measurement $\boldsymbol{y}_{k-1}$ according to Eq. (2.5). The forward model is needed in problems such as filtering, prediction, optimization etc. where the impetus is to predict or improve the output given a set of inputs. To learn such a model we would need to learn state transition function f and the measurement function h.

- *Inverse Models*: They predict the action required in the current state so as to reach the desired next state. This learning is the opposite to what model



(a) Direct Model Learning.

(b) Distal Teacher Learning.

(c) Indirect Modeling or Feedback Error Learning.

Figure 2.2.: Learning architectures for Inverse modeling: (a) *Direct Inverse Modeling*, the torque $\boldsymbol{u}_{total}$ is the action, and the joint angle $\boldsymbol{q}$ is the output. Regression on both leads to modeling. (b) *Distal Teacher Learning*, the forward model provides the errors to the inverse model for learning. (c) *Feedback Error Learning* model where an inverse model is learned using the feedback error, [7]. The dotted arrows signify learning signals.

learning achieves. An inverse model has applications in the areas of robot control like Inverse dynamics control and computed torque control. In this work we use a learning architecture, Feedback Error Learning, which is traditionally used for Inverse model learning problems. Inverse models need to map the desired next state $\boldsymbol{s}_{k+1}^{\text{des}}$ and output $\boldsymbol{y}_k^{\text{des}}$ to the current action $\boldsymbol{a}_k$ given the current state $\boldsymbol{s}_k$, and this needs the inverse of the state transition function f^{inv}. The mapping of f^{inv} need not be unique, for example the inverse of the square function, i.e., square root is not unique.

- *Mixed Models*: These models are useful when the inverse model is actually non unique, and we need uniqueness when mapping the inverse model. This is achieved by combining constraints in the form of a forward model to make sure that the inverse model is unique in mapping. Mixed modeling approach has applications in Inverse Kinematics, Operational Space Control and Multiple model control. Inverse kinematics and Operational space control have non unique mapping when the robot has redundant degrees of freedom.

- *Multi-step Prediction Models*: When we need to predict the output state of a system n steps in advance these models are useful. Normal forward modeling for n steps would accumulate errors, hence we need a model that takes care of this error accumulation. They are useful in tasks of planning, model predictive control etc.

The objective of this work is to learn trajectories or RMPs, so as to control a robot. This task is closest in definition to an inverse model learning problem, albeit we are trying to learn not the inverse system model, but the trajectories that can be input to an inverse model. This difference would become more clear in the next chapter. Inverse model learning as stated previously is a tricky problem, because it can have one to many mapping from the desired state space to the input action space. Below is a list of architectures that can be used to learn an inverse model of a system:

- *Direct Inverse Modeling*: It is the simplest inverse model learning architecture as shown in Fig. 2.2(a) and has been used in many applications such as inverse dynamics control [13]. The constraint for using Direct Inverse modeling is that the inverse mapping must be unique. If the mapping is unique then standard regression techniques can be used to learn the inverse model. The model is trained directly by measuring the input and output to the system and applying regression methods to learn the model.

- *Distal Teacher Learning*: This learning architecture is useful in learning inverse models with many to one mapping. This is done by teaching particular, desired solutions so as to reduce the learning problem to a unique mapping problem. It uses supervised learning and is goal directed as shown in Fig. 2.2(b). The forward model guides the inverse model to solutions such that the error signal of the forward model to the inverse model are minimized. The forward model can ensure giving unique errors even if the actual inverse model of the system has one to many mapping, hence the forward model acts as a teacher. The learning is goal directed to reduce the error signal of the forward model. Distal Teacher Learning can be potentially unstable and can accumulate errors, but these shortcomings have been overcome for many applications of inverse kinematics modeling [13].

- *Indirect Inverse Modeling*: In problems such as inverse kinematics the mapping of the inverse model need not be unique. To deal with this there are indirect inverse modeling methods like feedback error learning [7] as shown in Fig. 2.2(c). Here the feedback of the system output and desired trajectory are used to learn the inverse model. This learning is goal directed and aims to minimize the feedback by perfecting the inverse model.

The objective of this work is to learn trajectories, by using the outputs of the balancing controllers. This problem statement automatically makes Feedback Error learning the obvious choice to learn the trajectories, using the balance controller's feedback as an error signal. We describe Feedback Error Learning in more detail in the following paragraphs.

2.2.1 Feedback error learning

Feedback Error Learning (FEL) is a biologically inspired idea by Kawato in [7]. Kawato postulated FEL as in inverse model learning method for the cerebellum

to learn motor commands to be passed onto the motor cortex of the cerebrum to produce actions, based on the feedbacks from the reflex arc [7, 14]. After the learning is complete, the inverse model in the FEL architecture does not require a change of neural connections, i.e., when an inverse model is learned by means of FEL its connections remain the same when the learned model is in use, performing its desired task. The learning is also goal directed, to reduce the feedback from the visual task space. Kawato further realized that this architecture can be used for inverse dynamics modeling for robotics in [7]. FEL can learn inverse models of systems with redundant degrees of freedom and this property will be useful when the learning is combined with task space control.

The inverse dynamics modeling architecture of FEL is shown in Fig. 2.2(c). It has input desired trajectories in the form of $\boldsymbol{q}_\mathrm{d}$, $\dot{\boldsymbol{q}}_\mathrm{d}$, and $\ddot{\boldsymbol{q}}_\mathrm{d}$. These are fed into the inverse model to be learned that ideally would generate the precise forward motor torques $\boldsymbol{u}_\mathrm{ff}$. This forward motor torque is summed with feedback motor torque to produce the total torque $\boldsymbol{u}$, and fed into the system to be controlled

$$\boldsymbol{u} = \boldsymbol{u}_\mathrm{ff} + \boldsymbol{u}_\mathrm{fb},$$

where u_fb is the feedback torque generated with a Proportional and Derivative (PD) control law give by

$$\boldsymbol{u}_\mathrm{fb} = \boldsymbol{K}_\mathrm{P}\left(\boldsymbol{q}_\mathrm{d} - \boldsymbol{q}\right) + \boldsymbol{K}_\mathrm{D}\left(\dot{\boldsymbol{q}}_\mathrm{d} - \dot{\boldsymbol{q}}\right), \tag{2.7}$$

with positive gains $\boldsymbol{K}_\mathrm{P}$ and $\boldsymbol{K}_\mathrm{D}$, to ensure that the desired behavior is achieved in the presence of uncertainty and model errors. Learning an inverse model in this case is equivalent to modeling the forward torque u_ff to the system. Consider $\boldsymbol{u}_\mathrm{d}$ to be the desired forward torque to produce the desired trajectory $\boldsymbol{q}_d$. In a supervised learning setup the forward torque can be represented parametrically using basis function as,

$$\boldsymbol{u}_\mathrm{ff} \approx \boldsymbol{f}\left(\boldsymbol{q}_\mathrm{d}, \dot{\boldsymbol{q}}_\mathrm{d}, \ddot{\boldsymbol{q}}_\mathrm{d}, \boldsymbol{\theta}\right) = \boldsymbol{\phi}(\boldsymbol{q}_\mathrm{d}, \dot{\boldsymbol{q}}_\mathrm{d}, \ddot{\boldsymbol{q}}_\mathrm{d})\boldsymbol{\theta}\ , \tag{2.8}$$

where $\boldsymbol{\theta}$ are the weights of the model, which we want to obtain, to learn the model and $\boldsymbol{\phi}(\boldsymbol{q}_\mathrm{d}, \dot{\boldsymbol{q}}_\mathrm{d}, \ddot{\boldsymbol{q}}_\mathrm{d})$ are the basis functions that are defined over input joint angles $\boldsymbol{q}_\mathrm{d}$ and their derivatives $\dot{\boldsymbol{q}}_\mathrm{d}$ and $\ddot{\boldsymbol{q}}_\mathrm{d}$). The forward torque, hence is a linear sum of nonlinear basis functions. To learn the model by regression methods we need an error between the calculated feedback and the desired feedback term

$$E = \frac{1}{2}\sum_{k=1}^{N}\left\|\boldsymbol{e}^k\right\|^2 = \frac{1}{2}\sum_{k=1}^{N}\left\|\boldsymbol{u}_\mathrm{d}^k - \boldsymbol{f}\left(\boldsymbol{q}_\mathrm{d}^k, \dot{\boldsymbol{q}}_\mathrm{d}^k, \ddot{\boldsymbol{q}}_\mathrm{d}^k, \boldsymbol{\theta}\right)\right\|^2\ , \tag{2.9}$$

over N instants of time. To learn the weight parameters $\boldsymbol{\theta}$, we can use any Newton based method, like Gradient descent and change the weight vectors in the direction of diminishing error E. If the inverse dynamics model is precise, i.e., feedforward $\boldsymbol{u}_\mathrm{ff}$ is error free, hence dominant and $\ddot{\boldsymbol{q}}_\mathrm{d} \approx \ddot{\boldsymbol{q}}$, the controller $\boldsymbol{u}_\mathrm{fb}$ will remain dormant most of the time, i.e., $\left\|\boldsymbol{u}_\mathrm{fb}\right\| \approx 0$. Otherwise, the feedback controller will generate torques to make sure that the output of the system follows

the control laws and gives desired outputs. Kawato [7] realized that the feedback of a linear control law can be used as an error signal as

$$\boldsymbol{e} = \boldsymbol{u}_{\mathrm{d}} - \boldsymbol{\Phi}\left(\boldsymbol{q}_{\mathrm{d}}, \dot{\boldsymbol{q}}_{\mathrm{d}}, \ddot{\boldsymbol{q}}_{\mathrm{d}}\right)\boldsymbol{\theta} = \boldsymbol{u} - \boldsymbol{u}_{\mathrm{ff}} = \boldsymbol{u}_{\mathrm{fb}} \tag{2.10}$$

for inverse dynamics learning. Hence, the feedback signal can function as an error signal for the purpose of supervised learning. Stability analysis of FEL has been performed by Kawato in [7] and by Nakanishi and Schaal in [15] and have found the method to provide stable control while learning and thereafter.

We can now state that RMPs for joint trajectories are the model that we want to learn, and FEL is the method chosen to learn them. Next we look at basics of locomotion as the final goal of this thesis is to improve upon gaits online.

2.3 Legged Locomotion in Robotics

Locomotion is an important characteristics of animals that robots try to emulate. Robots initially used wheels for locomotion, but soon the need for using legs for locomotion of robots became apparent. Raibert in [16] gives two important reasons for developing robots with legs. Firstly, there is a need for creating machines that can travel across a difficult terrain. Wheeled robots can only access even and smooth terrains, whereas some robots also need to travel across much harder and uneven terrains. Legged systems can use isolated footholds that give support and traction, but wheeled robots would need continuous paths, which do not always exist. Secondly, legs give active suspension that wheels do not provide, i.e. a legged system decouples the path of its body from the path of the legs, allowing the legs to navigate a rugged terrain keeping its body's trajectory smooth. Fig. 2.3 show the Boston Dynamics' robot "Big Dog" climbing a hill side with snow, lined with trees. This terrain would be impossible to navigate with a wheeled robot carrying a payload, proving the importance of legs in locomotion.

Figure 2.3.: Big Dog climbing a hilly terrain. Image from Boston Dynamics' website - `http://www.bostondynamics.com/robot_bigdog.html`

2.3.1 Gait classification in animals

To understand legged gait we first describe the gaits that animals follow. The first work in studying gaits was done by Eadweard Muybridge [17] when he took a series of stop motion photographs of a galloping horse. From then on various he

went on to compile walking and running behavior of over 40 mammal, including humans.

A *stride* is defined as "the complete walk cycle of leg movements, e.g. for once setting down a foot to next setting down of the same foot" [18]. *Duty factor* is defined within the period of the stride, as the fraction of the duration of a stride when the foot is on the ground [18]. A walk is when the duty factor of a gait is more than 0.5, and a run is when the duty factor is less than 0.5, i.e., in a run both feet are off the ground at certain stages [18].

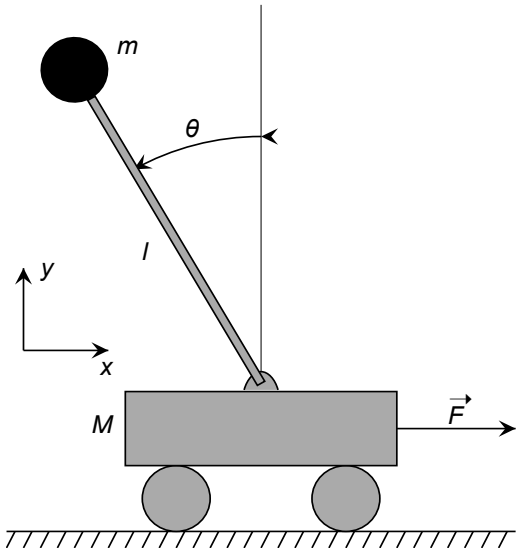
The last criterion gaits are classified upon is whether the gait is symmetrical or asymmetrical. Symmetrical gait is one where left and right feet of each pair (front or back) have equal duty factors and relative phases which differ by half a cycle, within a stride. Examples can be human walk and running, trot of a horse, amble of an elephant, etc. Asymmetric gait is one where the pair of feet have the same relative phase, this can be seen in the gallop of a horse and bounding gaits of deers. Since this work deals with bipedal motion we concern ourselves with symmetric human like gaits.

2.3.2 Gait classification in robots

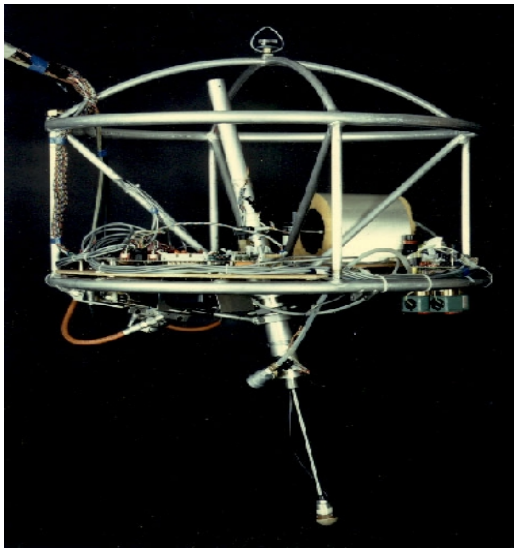
A robot gait can be classified as being *active* or *passive* based on whether the robot has actuation or not [19]. Passive robot gaits do not use actuation and are important for learning gaits that dissipate less energy. Passive robot walkers are usually designed to walk down a slope, by training their trajectories to settle in a limit cycle that dissipates less energy by using the periodicity of the gait [19]. The biped model used in this thesis has an active gait, i.e. it uses actuation, which allows us more control over the robots gait.

Robot gaits on the other hand are also classified based upon the type of stability they have, *static* or *dynamic*. Static gaits can be defined as those gaits which are stable at each instant of time in within the stride, i.e., if the robot is paused at any instant of time within the stride, the robot will have enough feet on the ground so as to have the center of gravity within its broad support base [16]. These gaits were the easiest to achieve in the early stages of legged locomotion. Ralph Mosher at General Electric constructed one of the first human driven static crawlers— The walking truck [16]. The idea of a static gait is that the forward velocity is kept sufficiently low such that it does not affect stability calculations of walker [16].

A dynamic gait on the other hand balances actively, i.e., the velocities and the kinetic energy of the masses determines the behavior of the gait, and not the geometry or the configuration of legs of the device. Dynamic gait is also closer to way humans and most other animals walk or run. Since dynamic gait requires the robot to balance actively, the robot can fall unlike static gaits. The control system needs to make sure that the sum of forces making the robot fall in one direction must be equal to the sum of forces in its opposite direction such that the robot does not fall [16].



(a) Cart and Pole experiment



(b) Final forward trajectory for the torso.

Figure 2.4.: (a) Cart and Pole Diagram take from the Inverted pendulum wikipedia page - `http://en.wikipedia.org/wiki/Inverted_pendulum`. The idea is to move the cart along the sides such that the pendulum is balanced upright at the center. (b) Raibert's one legged hopper, the balance is maintained with a control system regulating height and direction of hops, also the design of the hopper helps in maintaining balance, from MIT's website, `http://www.ai.mit.edu/projects/leglab/robots/3D_hopper/3D_hopper.html`.

2.3.3 Early experiments in active balance

Research in active balance goes back to the problem of balancing an inverted pendulum, or balancing an inverted double pendulum [20]. Cannon [20] worked on balancing an inverted double pendulum on a cart and provided an analysis of different system parameters affecting the equilibrium of the pendulums. The inverted pendulum experiments are since used as primary experiments for learning balance and control. A schematic of the cart and pole problem is shown in Fig 2.4(a), the idea is to balance the pole upright by moving the cart along a line. The work on inverted pendulums was followed by the development of the first dynamic biped walker by Miura and Shimoyama [16], which had three actuators, one for each leg to enable the robot to swing sideways and another that allowed the legs to move w.r.t. its hips. This three actuator configuration is important as it allows a robot without knees to walk, by shuffling. In this thesis we have worked at the problem of learning the balanced trajectory of an oscillating double pendulum first, before moving on to balancing a biped's trajectory, using the double pendulum as a test bed.

Before finishing the section about experiments in robotic gaits a mention must be made about dynamically stable running robots. The first running robot was designed by Matsuoka [16, 21], with four legs that ran with long hops. Matsuoka

also worked on a simple one legged hopper that inspired Raibert to design a one legged hopper capable of running [16]. Raibert's walker had a stance phase, when the robot was out of balance but on the ground and a flight phase when it was airborne. The gait was realized by a continuous hop, a forward velocity of the leg during the stance phase and posture of the the torso while being in the support phase. Raibert's hopper is shown in Fig. 2.4(b). The single legged hopper is an important milestone because it proves the concept of active balance without doubt because there is no statically stable configuration for the robot at any point of time in its trajectory, the robot is always tipping or airborne.

2.3.4 Gaits in biped robots

Biped robots have been traditionally designed with a dynamic gait. This gait can be precomputed and provided to the biped, after which the robot is controlled by tracking this trajectory [12, 22]. There are also other methods present to generate this gait, for e.g. Grizzle [23, 24] uses Poincaré methods to generate future trajectories based upon the criteria of maintaining stability in the walk cycles. The idea is to have the periodic gait to be based on a limit cycle so as to be stable. In this thesis we use the first method of control, where we feed a learned trajectory and control the robot using PD control laws. Since the trajectories we use are generated using RMPs, which use limit cycles, the concern for stability of the generated periodic trajectory is appeased.

In this chapter we discussed discussed the topic of Gaits, their representation with RMPs and FEL as a model learning method. In the next chapter we develop the concept of learning trajectories used FEL with a novel modification to the traditional FEL approach. We look at applying this method to learning balanced trajectories in a biped simulator that we described previously.

3 Feedback Error Learning of Movement Primitives

Feedback error learning is a method developed for inverse model learning. Our problem statement involves learning forward trajectories and not the inverse model. We assume that we already have a satisfactory inverse model to work with. This assumption is can be valid in scenarios where the inverse model is already known, but not all the trajectories that keep the system in motion.

For example consider a biped robot whose inverse model is known, also known is the gait followed by its limbs, now we want to learn the trajectory followed by the torso of the biped in the gait cycle such that the biped would be stable and in balance.

As mentioned before this method helps keep feedbacks of the balance controller low. Therefore, it will improve its gait as the terrain changes.

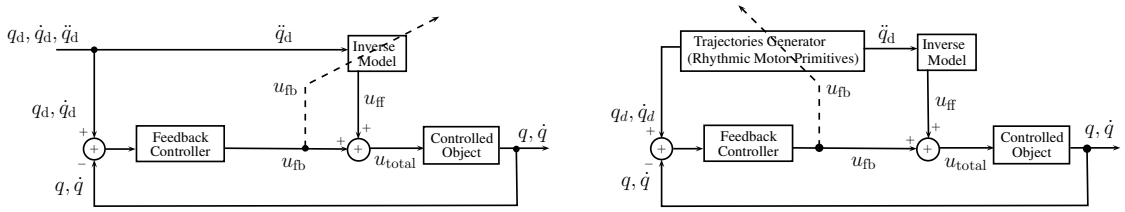
3.1 Development of FEL for Trajectory Learning

An inverse dynamics model takes in desired joint trajectories as input and outputs torques required to be produced by the actuators in the joints to perform the desired joint trajectories. Hence, the desired trajectories that are input to the inverse dynamics model are the forward trajectories of the system, that produce forward torque. Assuming that a robot is a rigid body system, its inverse dynamics equation is given by [25]

$$\boldsymbol{u}_{\text{ff}} = \mathbf{M}(\boldsymbol{q})\ddot{\boldsymbol{q}} + \mathbf{c}(\boldsymbol{q}, \dot{\boldsymbol{q}}) + \mathbf{g}(\boldsymbol{q}) \,, \tag{3.1}$$

where $\boldsymbol{q}$,$\dot{\boldsymbol{q}}$ and $\ddot{\boldsymbol{q}}$ denote joint positions, velocities and acceleration, respectively, $\mathbf{M}(\boldsymbol{q})$ is the inertia matrix, $\mathbf{c}(\boldsymbol{q}, \dot{\boldsymbol{q}})$ denotes centripetal and Coriolis forces, and $\mathbf{g}(\boldsymbol{q})$ denotes the gravity forces and $\boldsymbol{u}_{\text{ff}}$ as mentioned previously denotes forward torque.

The joint angles and their derivatives in Eq. (3.1) are forward trajectories generated by RMPs, according to Eq. (2.2). We want to improve upon the forward trajectories based on the feedback of the balance controller, Fig. 3.1(b) shows the modified architecture. The original FEL architecture is also presented, repeated for comparison in Fig. 3.1(a). We can observe that instead of learning the inverse model the modified FEL learns the forward trajectories. Therefore, there is no desired trajectory to compare our system outputs with but rather a control law which decided the direction of learning. We first describe the equations for the original FEL architecture, and then develop the equations for the modified model using them.



(a) Original feedback error learning. (b) Modified feedback error learning.

Figure 3.1.: (a) Original feedback error learning model where an inverse model is being learned using the feedback error, from [7]. (b) Modified feedback error learning of gaits. The feedback from the the balance controller's feedback output is used for learning the gait and not the inverse model.

Let us assume that the forward torque is defined as per Eq. (2.8), where $\boldsymbol{\theta}$ are the weight vectors of the inverse model to be learned. Combining Eq. (2.8) and Eq. (3.1) we have

$$\boldsymbol{f}\left(\boldsymbol{q}_{\mathrm{d}}, \dot{\boldsymbol{q}}_{\mathrm{d}}, \ddot{\boldsymbol{q}}_{\mathrm{d}}, \boldsymbol{\theta}_{\boldsymbol{h}}\right) = \mathbf{M}(\boldsymbol{q}_{\mathrm{d}})\ddot{\boldsymbol{q}}_{\mathrm{d}} + \mathbf{c}(\boldsymbol{q}_{\mathrm{d}}, \dot{\boldsymbol{q}}_{\mathrm{d}}) + \mathbf{g}(\boldsymbol{q}_{\mathrm{d}}) \,, \tag{3.2}$$

i.e., $\boldsymbol{f}\left(\boldsymbol{q}_{\mathrm{d}}, \dot{\boldsymbol{q}}_{\mathrm{d}}, \ddot{\boldsymbol{q}}_{\mathrm{d}}, \boldsymbol{\theta}_{\boldsymbol{h}}\right)$ is the parameterized form of the inverse dynamics model, and θ is the weight vector to be calculated to learn the inverse model. Minimizing the error in Eq. (2.9) using stochastic gradient descent

$$\begin{aligned} \boldsymbol{\theta}_{h+1} &= \boldsymbol{\theta}_h - \alpha_h \nabla_{\boldsymbol{\theta}} E \\ &= \boldsymbol{\theta}_h + \alpha_{\mathrm{h}} \sum\nolimits_{\mathrm{k}=1}^{\mathrm{N}} \boldsymbol{e}^k \frac{\partial \boldsymbol{f}(\boldsymbol{q}_{\mathrm{d}}^k, \dot{\boldsymbol{q}}_{\mathrm{d}}^k, \ddot{\boldsymbol{q}}_{\mathrm{d}}^k, \boldsymbol{\theta}_h)}{\partial \boldsymbol{\theta}_h}, \end{aligned} \tag{3.3}$$

where α_h is a learning rate or step size, and E is the summed up error $\boldsymbol{e}$ over N measurements as defined in Eq. (2.9). Stochastic gradient descent for a convex error is guaranteed to converge to the optimal solution under mild conditions such as $\sum_{h=1}^{\infty} \alpha_h \rightarrow \infty$ and $\sum_{h=1}^{\infty} \alpha_h^2 < \infty$, see [26]. This learning can also be done in a single step, but we need an online learning method that can keep learning as the surroundings change, i.e., continuous online learning.

Using the result of feedback error learning, i.e., Eq. (2.10), such an update rule would become

$$\boldsymbol{\theta}_{h+1} = \boldsymbol{\theta}_h + \alpha_h \sum\nolimits_{k=1}^{N} \boldsymbol{u}_{\mathrm{fb}}^k \boldsymbol{\Phi}\left(\boldsymbol{q}_{\mathrm{d}}^k, \dot{\boldsymbol{q}}_{\mathrm{d}}^k, \ddot{\boldsymbol{q}}_{\mathrm{d}}^k\right)^T, \tag{3.4}$$

which is known as feedback error learning [7]. Learning and control using feedback error learning for inverse dynamics is stable under the following conditions defined for the gains $\boldsymbol{K}_{\mathrm{P}}$ and $\boldsymbol{K}_{\mathrm{D}}$ as $\left\|\boldsymbol{K}_{\mathrm{D}}\right\|^2 > \left\|\boldsymbol{K}_{\mathrm{P}}\right\|$ [15]. Nevertheless, these gains have to be chosen on trial-and-error basis initially. Since we have discussed FEL for inverse dynamics as shown in Fig. 3.1(a), we now use these results for developing FEL for learning RMPs.

In the modified FEL architecture shown in Fig. 3.1(b) the RMPs are to be learned, and not an inverse model. The RMPs as per Eq. (2.2), can be assumed to be a parametric function of weights $\boldsymbol{\theta}$ and basis functions $\boldsymbol{\psi}$, i.e.,

$$\ddot{\boldsymbol{q}}^k = g(\boldsymbol{\theta}, \boldsymbol{\psi}^k, \dot{\boldsymbol{q}}^{k-1}, \boldsymbol{q}^{k-1}) \,, \tag{3.5}$$

where g is the functional representation of Eq. (2.2). Since the inverse dynamics model is assumed to be known, the forward torque would just be the function of joint trajectories, i.e., $\boldsymbol{u}_{\text{ff}} = \boldsymbol{f}(\boldsymbol{q}_{\text{d}}^k, \dot{\boldsymbol{q}}_{\text{d}}^k, \ddot{\boldsymbol{q}}_{\text{d}}^k)$. Hence the update rule for RMPs can be given as

$$\boldsymbol{\theta}_{h+1} = \boldsymbol{\theta}_h + \alpha_{\text{h}} \sum\nolimits_{\text{k}=1}^{\text{N}} \boldsymbol{u}_{\text{fb}}^k \frac{\partial \boldsymbol{f}(\boldsymbol{q}_{\text{d}}^k, \dot{\boldsymbol{q}}_{\text{d}}^k, \ddot{\boldsymbol{q}}_{\text{d}}^k)}{\partial \boldsymbol{\theta}_h},$$

where parameters $\boldsymbol{\theta}$ are the weights of the rhythmic motor primitives. This implies by using Eq. (3.5) and the application of the chain rule the new weight update rule

$$\boldsymbol{\theta}_{h+1} = \boldsymbol{\theta}_h + \alpha_h \sum\nolimits_{k=1}^{\text{N}} \boldsymbol{u}_{\text{fb}}^k \frac{\partial \boldsymbol{f}(\boldsymbol{q}_{\text{d}}^k, \dot{\boldsymbol{q}}_{\text{d}}^k, \ddot{\boldsymbol{q}}_{\text{d}}^k)}{\partial g(\boldsymbol{\theta}_h, \boldsymbol{\psi}^k, \dot{\boldsymbol{q}}^{k-1}, \boldsymbol{q}^{k-1})} \frac{\partial g(\boldsymbol{\theta}_h, \boldsymbol{\psi}^k, \dot{\boldsymbol{q}}^{k-1}, \boldsymbol{q}^{k-1})}{\partial \boldsymbol{\theta}_h}, \tag{3.6}$$

is formulated. We notice that g is a temporal function, and is not just based upon inputs of weights $\boldsymbol{\theta}$ and basis functions $\boldsymbol{\psi}$. Therefore, an analytical solution of the partial derivative of $\frac{\partial g(\boldsymbol{\theta}_h, \boldsymbol{\psi}^k, \dot{\boldsymbol{q}}^{k-1}, \boldsymbol{q}^{k-1})}{\partial \boldsymbol{\theta}_h}$ is not possible, and it has to be calculated functionally, using the properties of calculus of variations by numerical differentiation. Hence, Eq. (3.6) gives the new update rule for learning RMPs using FEL.

Now we would discuss the type of feedback that we want to use for the learning process. There exist two kinds of feedback controllers, i.e., the feedback controller either tracks the rhythmic trajectory given by

$$\boldsymbol{u}_{\text{track},i} = \boldsymbol{u}_{\text{fb},i} = \boldsymbol{K}_{\text{P},ii} \left(\boldsymbol{q}_{\text{d},i} - \boldsymbol{q}_i \right) + \boldsymbol{K}_{\text{D},ii} \left(\dot{\boldsymbol{q}}_{\text{d},i} - \dot{\boldsymbol{q}}_i \right),$$

alternatively, it acts as a stabilizing controller such as

$$\boldsymbol{u}_{\text{stabilize},j} = \boldsymbol{u}_{\text{fb},j} = \boldsymbol{K}_{\text{P},jj} \left(\boldsymbol{q}_{\text{d},j} - \boldsymbol{q}_i \right) + \boldsymbol{K}_{\text{D},jj} \left(-\dot{\boldsymbol{q}}_j \right).$$

For learning the rhythmic motor primitive, ideally, in a fully actuated system, we realize that we can use $\boldsymbol{e}_j = \boldsymbol{u}_{\text{stabilize},j}$ as an error signal for all stabilizing controllers and define $\boldsymbol{e}_i = 0$ for all other dimensions. This can also be used in the case of over actuated systems where compliant actions are preferred. Moreover, we also realize that this will be a problem in underactuated systems. In underactuated systems, many feedbacks are summed to be given into a single actuator. Thus, to learn one input trajectory we would need to minimize all the sum of feedbacks affecting it. Consequently, in underactuated systems we need to minimize the sum of the stabilizing and tracking feedbacks, this is an important point and will be discussed along with the experiments. In our experiments the robot arm is a fully actuated system and the planar biped is an underactuated system.

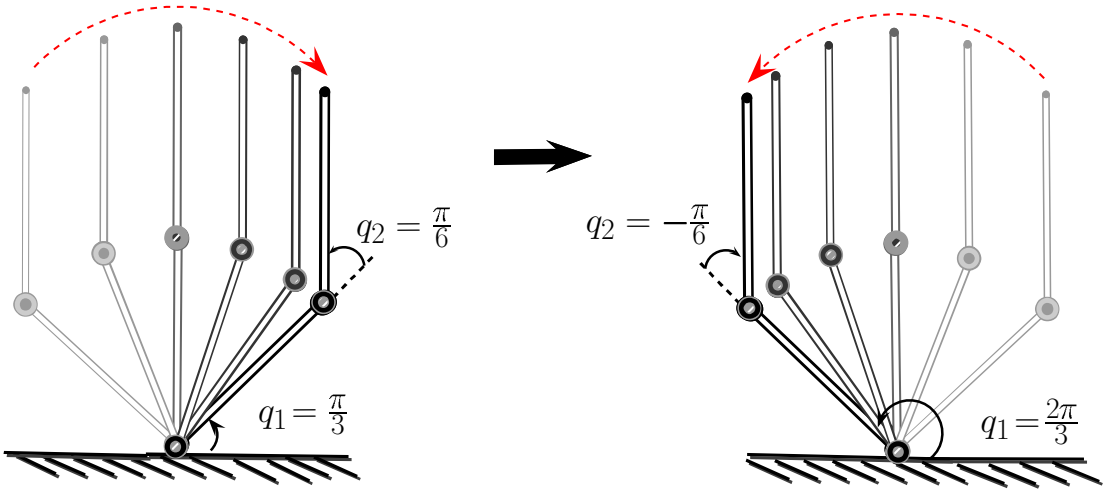


Figure 3.2.: Trajectory cycle for the two link robot manipulator. The lower link is oscillates between $\pm\pi/6$ around the vertical, the upper link has to learn to remain upright, using FEL.

3.2 Experiments to Learn Trajectories using FEL

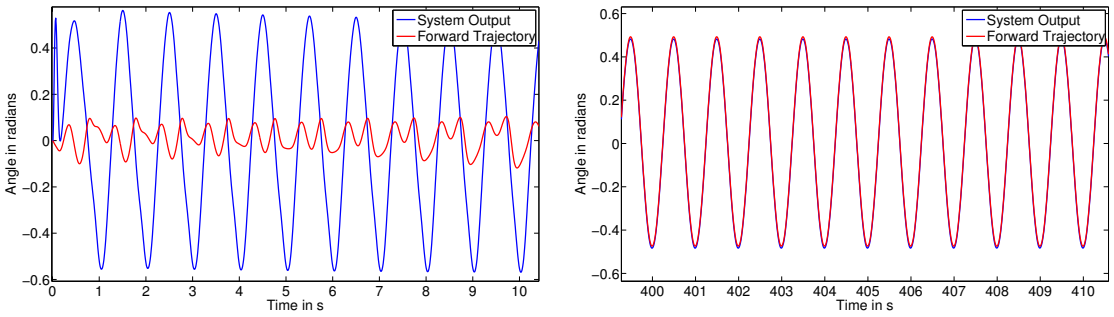
Two experiments are described in this section that use the FEL architecture for learning trajectories modeled by RMPs. The first is a *two link robot manipulator*, whose lower links trajectory is defined and upper link trajectory is to be learned while staying in balance. The second experiment is a *biped robot*, whose torso's trajectory is unknown and needs to be learned from the feedback to stay in balance.

3.2.1 Evaluation on a Two Link Robot Manipulator

The two link manipulator served as a toy example for testing feedback error learning with rhythmic motor primitives. It is similar to a double pendulum, which as previously stated, is used as a toy example for most control learning problems.

Model description

The two link manipulator consists of two links and two actuators as can be seen in Fig. 3.2. The lower link of the manipulator oscillates between a range of angles while inverted, i.e., its mass is above its pivot. The upper link initially has a random feedforward trajectory, which can lead to the manipulator falling in the absence of feedback signals. The manipulator has two actuators: one for each link.



(a) Initial trajectories for upper link.

(b) Final forward trajectory for upper link.

Figure 3.3.: (a) Initial trajectories of the upper link. The dashed line is the system output and the continuous line is the forward trajectory. The forward trajectory is unstable and can make the two link manipulator fall off. The system feedback ensures that this does not happen, hence the system trajectories because of feedback are complete (b) Final trajectories of the upper link, the two trajectories completely overlap proving that the forward gait is completely learned using feedbacks to converge to the control law's criterion.

The Standard two link manipulator

This model is called the standard two link manipulator as each link of the manipulator weighs 1 kg, and is 1 m long. The period of oscillation for the lower link is 1 s. The oscillation in the joint space for the lower link is between $\frac{\pi}{3}$ and $\frac{2\pi}{3}$, i.e., $(\frac{\pi}{2} \pm \frac{\pi}{6})$ rad. This forward trajectory is generated using rhythmic motor primitives [5] with 10 basis functions. 10 basis functions were enough to represent a sinusoidal wave in our case. Adding more basis functions leads to more computations and increases the computation time of the simulations. Thus, we used 10 basis functions. The initial forward trajectory of the upper link q_2 as shown in Fig. 3.3(a), is chosen by random sampling of the weights for the rhythmic motor primitives, also with 10 basis functions. The dynamics of the manipulator were taken from [27], and the equations have the same form as Eq. (3.1). The dynamics equations are detailed in the Appendix A. The sampling frequency for the simulator was chosen 100 Hz as the learning was more stable at this frequency. Increasing the frequency increased the quality of results, i.e., faster and stable learning, but it also increased the computation time.

Feedback used for learning

The control signal for each link is the sum of the torques generated by the inverse model $\boldsymbol{f}$ and the feedback torque generated by the PD control. The feedback torque for the first link measures the PD error between the generated trajectory and the joint angles of the first link. The feedback torque for the second link measures the PD error of the second link of the manipulator from the vertical line

using joint angles. This is the stabilizing torque $\boldsymbol{u}_{\text{stabilize}}$ to be minimized by gait learning,

$$\boldsymbol{u}_{\text{stabilize}} = K_{\text{P2}}(\frac{\pi}{2} - (\boldsymbol{q}_1 + \boldsymbol{q}_2)) + K_{\text{D2}}(0 - (\dot{\boldsymbol{q}}_1 + \dot{\boldsymbol{q}}_2)) \,,$$

where, both the angles $\boldsymbol{q}_1$ and $\boldsymbol{q}_2$ are the system output measured from the horizontal axis and their derivatives are $\dot{\boldsymbol{q}}_1$, $\dot{\boldsymbol{q}}_2$ respectively. The forward trajectory for the first link is $\boldsymbol{q}_{1\text{d}}$ and for the second link is $\boldsymbol{q}_{2\text{d}}$. For a balanced second link we need the second link's forward and system output angle to converge to $\pi/2$ radians from the horizontal and its derivative to be 0 radians per second, i.e., $q^k_{1\text{d}} + q^k_{2\text{d}} = \pi/2$ and $\dot{q}^k_{1\text{d}} + \dot{q}^k_{2\text{d}} = 0$. We use the control laws as described in Eq. (2.7). Parameters K_{P1}, K_{D1}, K_{P2}, and K_{D2} are PD gains for the both the links and they were designed such that $\|K_{D2}\|^2 > \|K_{P2}\|$ for stable control and learning [15].

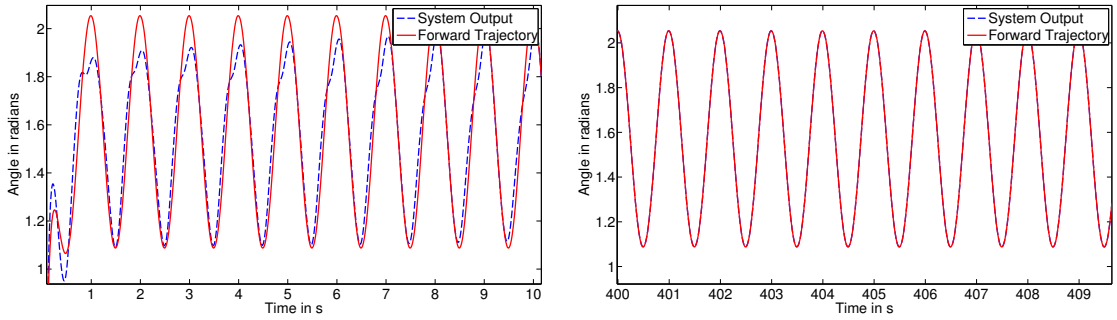
Using FEL to learn trajectories of the upper link

The actuator for the upper link has an initial random feedforward gait that is not balanced, i.e., it can lead to the manipulator falling, especially at the extremes of the lower link's trajectory. However, the control law follows the condition that the second link of the manipulator is always vertical. Therefore, initially the *stability feedback* ensures that the system never falls, generating feedback signals. These feedbacks train the weight vectors of the *feedforward* rhythmic motor primitives using gradient descent, so as to minimize the absolute balancing feedback torque using Eq. (3.6). The goal state $\boldsymbol{g}$ of the RMP from Eq. (2.2), also needs to be learned. This is because the goal state is also chosen randomly to ensure that the trajectory's state is completely unknown. The goal state is also solved using the FEL and Eq. (3.6), with the goal state being one the parameters $\boldsymbol{\theta}$, of the RMPs to be solved. The goal states were observed to perform better when learned slowly, i.e., with lower learning rates when compared to the learning rates of the weight's of the RMP's basis functions.

Using feedback error learning, the upper link's feedforward gait $\boldsymbol{q}_{2\text{d}}$ changes in the direction to reduce the feedback. For this fully actuated system, only the stability feedback, i.e., the second link's feedback, was minimized as this was sufficient for learning a stable gait. minimized as this was sufficient for learning a stable gait.

Results of FEL for the standard two link manipulator

As learning proceeds, the learned feedforward trajectory when passed through the inverse model, and the system, produces outputs that satisfy the control laws and keep the feedback at zero. The convergence to a balanced gait using feedback of the balance controller is shown in Fig. 3.3. Initially the forward trajectory produced was random, but the system trajectories are still perfect because of the feedback control laws. As the learning of the RMPs progressed the feedback was minimized. The balancing feedback torque reduces drastically and is almost equal



(a) Initial trajectories for lower link. (b) Final trajectories for lower link.

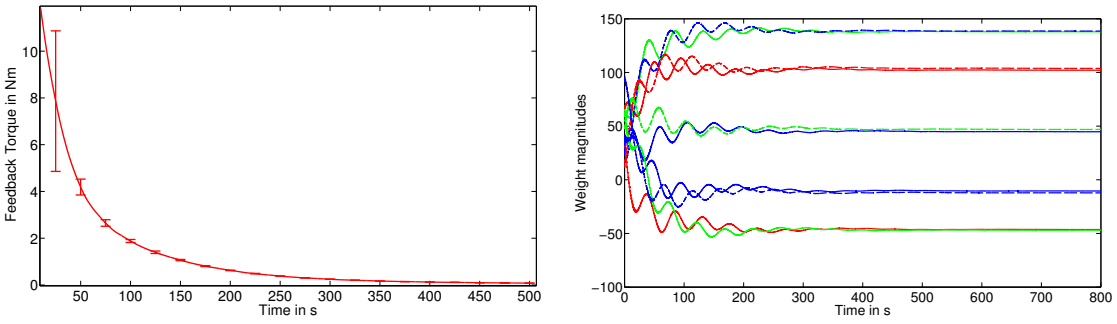
Figure 3.4.: Trajectory followed by the non learning lower link's joint is shown for completeness. (a) Initial trajectories of the lower link, the dashed line is the system output and the continuous line is the forward trajectory, they almost completely overlap, as this trajectory is known. The lower link is not able to follow the desired forward trajectory at some point, this is because of the wrong and unstable forward trajectory of the upper (second) link. (b) Final trajectories of the lower link. The two trajectories completely overlap. As the learning for the upper link is completed, the lower link's system trajectory follows the desired trajectory more easily with zero feedback.

zero after about 400 cycles. The two link model's average absolute feedback torque values for 20 random initial states are shown in Fig. 3.5(a). Fig. 3.5(b) shows the weight convergence for one run out of these 20 runs. The weights for all the 10 basis functions are perfectly settled after 400 cycles, which corresponds to 400 s. In all the experiments in this work the time constants for the RMPs are chosen as, $\alpha_z = 25$ and $\beta_z = \alpha_z/4$. They make sure that the system is critically damped.

For completeness, Fig. 3.4 shows the trajectory followed by the non learning lower link q_1, whose trajectory is known. It can been seen that the initial forward and system trajectory here is almost correct, and does not change even after learning of the second link is complete.

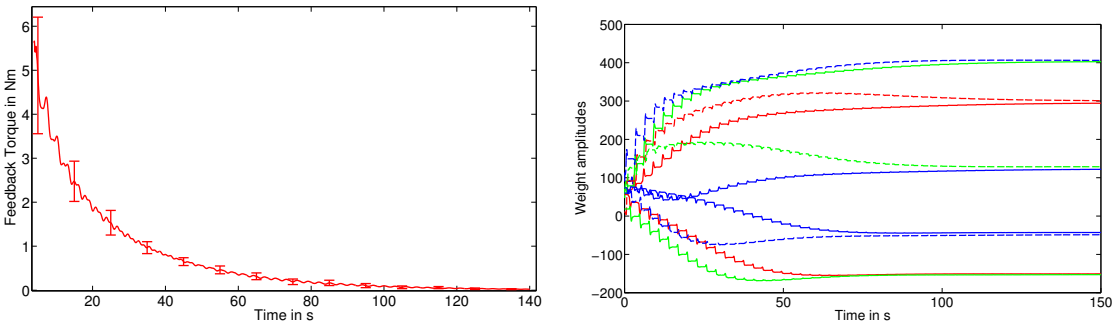
Two link manipulator: Human Arm Model

As a step towards a biped simulator, we evaluated our approach on a two-link manipulator with a weight distribution and frequency closer to that of a human hand, with the lower link acting as *Arm* and the upper link acting the *Forearm*. We set the upper link's weight to 2 kg and the lower link's weight to 2 kg, the length of both to 0.5 m and the period of a arm action cycle to 3 s. This model learned the Forearm's trajectory in 20 cycles, the reason for this quick learning can be a relatively large sampling frequency of 50 Hz. Another reason for this fast learning is also the presence of heavier and shorter links, which result in a more stable system (because of a compact distribution of weight around the actuators) that



(a) Two link Manipulator feedback. (b) Weight convergence for second link

Figure 3.5.: Feedback and weight convergence plots to show the learning of trajectories in 400 cycles. (a) Moving average of the absolute stabilizing feedback (i.e., the feedback for the second link), for the two link robot manipulator, averaged over 20 different initializations. It can be seen that as learning progresses the feedback reduces to zero in the case of the two link manipulator, proving that the forward trajectory is providing the complete required torque. (b) Weight convergence for 10 basis functions over one run. We can see that weights have converged after 400 cycles, which corresponds to 400 s.



(a) Two link Manipulator feedback (b) Weight convergence for second link or Forearm

Figure 3.6.: Feedback and weight convergence plots for the two link manipulator with human arm like weights. (a) Moving average of the absolute stabilizing feedback (i.e., the feedback for the second link), or the human arm like two link robot manipulator, with a cycle period of 3 s averaged over 50 different initializations. It can be seen that as learning progresses the feedback reduces, and becomes nil in the case of the two link manipulator, proving that the forward trajectory is providing the complete required torque. (b) Weight convergence for 10 basis functions over one run, we can see that weights have converged after 20 cycles or about 60 s.

allows large learning rates in gradient descent. The goal state was learned for the RMPs as well. The Fig. 3.6(a), shows reduction in the feedback as predicted by FEL as the learning progresses, and Fig. 3.6(b) shows the convergence of weights within 20 cycles or corresponding to 60 s.

3.2.2 Biped Model's simulation

This section describes gait learning using the modified FEL. We learn the forward stabilizing gait of a biped based on its rhythmic gait and a control law.

Problem description

Previously, there have been methods to learn biped gaits from demonstrations [6, 12] or using reinforcement learning [22, 28], and also with trajectories generated using Poincaré stability analysis [23, 29]. We try to learn a biped's torso's gait using feedback error learning. The biped robot model consists of three links: two legs, a torso, and two actuators, one for each leg to torso joint as shown in Fig. 3.7. The orientation of the measured angles is also shown in Fig. 3.7, the angles are defined keeping the stance leg as the fixed support, and defining the angles w.r.t. the vertical axis. The dynamics model of the three link biped are taken from [23]. The purpose of the simulation is to modify the gait of the torso of the biped while it continues to walk such that the balance controller's feedback is minimized. The biped model from [23] is used as it has a torso unlike other simple biped models or compass walkers. The goal of this experiment is to minimize the feedback required by the torso to remain in balance when its initial trajectory is random. As in the previous section we already have a gait to follow, which is the gait of the lower limbs, and we want to learn the gait of the torso according to a control law.

Biped model used

The biped dynamics model used for this thesis was taken from the work of Grizzle [23]. The model is a three link biped, one for each leg and one for the torso. The model has two actuators, between each leg and the torso.

The biped has only two actuators but five degrees of freedom (one for each link's orientation angle, and two for the x and y coordinate of the hip of the biped), hence is an underactuated system. The biped has a dynamic gait, with one leg swinging and the other resting on the floor providing fixed support much like an inverted pendulum, but with a floating base. The walk cycle of the biped [23] is described in Fig. 3.7. The walk cycle has two phases [23], the swing phase and the impact phase. A walk cycle is complete when a leg finishes one stance and one swing leg phase.

During the swing phase one leg acts as a stance leg over which the biped pivots and swings forward. The inverse dynamics dynamical model of the biped is ap-

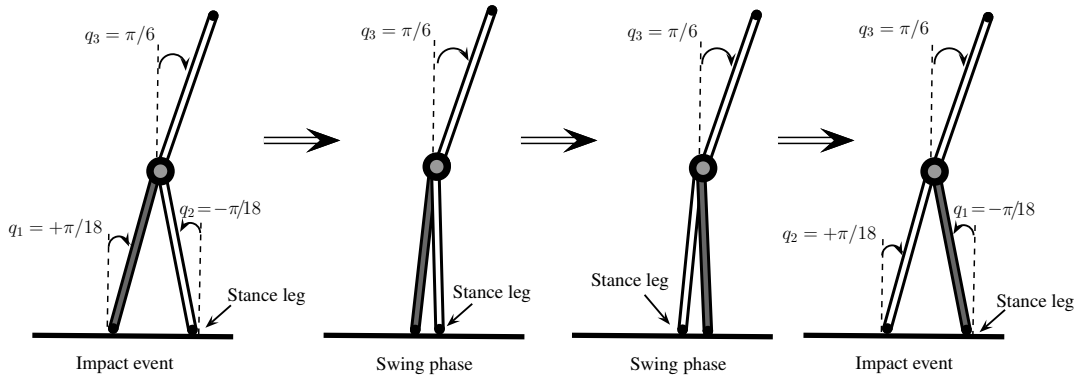


Figure 3.7.: Part of a walk cycle of a biped, starting from an impact phase, with the unshaded leg being the stance leg and the shaded leg swinging forward. The swing phase continues till the shaded leg reaches its terminal joint angle value, in our case the joint angle swing was from $+\pi/18$ to $-\pi/18$. At the end of swing phase the impact phase begins and the stance legs change as shown in the figure.

plicable in this phase and it is described in detail in Appendix B. The dynamics equation has a general form as described by [23]

$$\boldsymbol{B}\boldsymbol{u} = \mathbf{M}(\boldsymbol{q})\ddot{\boldsymbol{q}} + \mathbf{c}(\boldsymbol{q}, \dot{\boldsymbol{q}}) + \mathbf{g}(\boldsymbol{q}) \,, \tag{3.7}$$

where $\boldsymbol{q} = (q_1, q_2, q_3)$ denote joint angles, and $\dot{\boldsymbol{q}}$ and $\ddot{\boldsymbol{q}}$ are the velocities and acceleration, respectively of these joints. The inertia matrix is given by $\mathbf{M}(\boldsymbol{q})$, $\mathbf{c}(\boldsymbol{q}, \dot{\boldsymbol{q}})$ denotes centripetal and Coriolis forces, $\mathbf{g}(\boldsymbol{q})$ denotes the gravity forces, $\boldsymbol{u}$ is the forward torque, and $\boldsymbol{B}$ is the gain matrix that also couples the torques in case of underactuated systems. The orientation of the joint angles is visible in the Fig. 3.7. Note that this equation is same as the one described for a general robot in Eq. (3.1), with the exception of $\boldsymbol{B}$, as this equation is a general dynamics equation for rigid bodies. The complete dynamics equations with variable values are described in the Appendix B.

The impact phase is occurs when the swinging leg lands back on the ground. As soon as the swinging leg lands on the ground it becomes the new stance leg, and the old stance leg becomes the new stance leg. The joint angles are described based on the stance leg, so the joint angles and velocities are interchanged, i.e., if the initial states of joints and joint velocities is $\boldsymbol{X}^- = [q_1, q_2, q_3, \dot{q}_1, \dot{q}_2, \dot{q}_3]$, the after impact new state would be $\boldsymbol{X}^+ = [q_2, q_1, q_3, \dot{q}_2, \dot{q}_1, \dot{q}_3]$.

Setup of the biped

The biped's dynamics equation is of the same form as Eq. (3.1). The sampling rate of the simulator is 200 Hz. There are three joint angles to be considered, each of which uses rhythmic motor primitives. We used 16 basis functions per cycle. The number of basis functions is relatively high as more basis functions

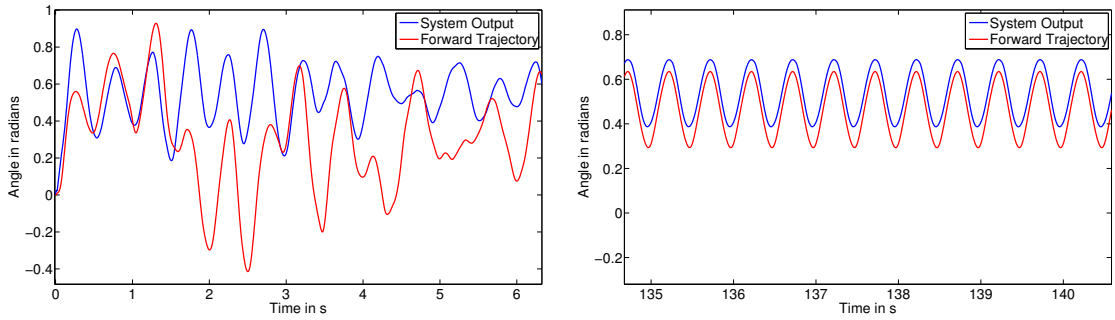
would mean that the trajectory would be more precisely defined and also more precisely trained. The weight and size of the biped are chosen analogous to that of a human. The torso weighs 40 kg, each of the legs weighs 15 kg, and the hip weighs 10 kg. The torso and limbs are 1 m long. The walking rate is assumed to be two steps a second or a complete walk cycle in 1 s. A walk cycle consists of pivoting the whole body on one limb called the stance leg and pushing the other limb, the swing leg forward. And then, switching the stance and the swing legs and repeat. A walk cycle is finished when a leg finishes one stance and one swing leg phase. As mentioned previously this is a dynamic gait where the impact stage is instantaneous.

The forces on a leg during the stance and the swing phase are completely different. The forces are defined only with respect to the stance leg as it is a fixed support, and the swinging leg and the torso, form the moving parts (or links) of this system. The dynamical model of the biped used is defined in terms of the stance leg as a fixed support and the torso and the swing leg being in motion attached to this fixed support, until the switch. The switching of stance legs is assumed to be instantaneous in the impact stage. In our simulations, we have not assumed any special force model for the impact stage. There is a simple switch in trajectories and forces between the old stance leg and the new stance leg. All the angles for the model are taken in the uniform coordinate system from the vertical, measuring in the clockwise direction as shown in Fig. 3.7. Hence, if the before-impact state is $\boldsymbol{X}^- = [q_1, q_2, q_3, \dot{q}_1, \dot{q}_2, \dot{q}_3]$, the after-impact state would be $\boldsymbol{X}^+ = [q_2, q_1, q_3, \dot{q}_2, \dot{q}_1, \dot{q}_3]$. Along with these angles their corresponding force equations are also switched as the other leg is now the fixed support, i.e., the stance leg. The stance leg is clearly shown by an arrow under the surface in Fig. 3.7. An example walk cycle is displayed in Fig. 3.7.

Feedbacks used for FEL

As discussed previously the biped system has three links resulting in 5 degrees of freedom with 2 actuators. Hence, it is an under-actuated system. Our first intuition was to reduce only the torso's balancing feedback as it was our required stabilizing control feedback $\boldsymbol{u}_{\text{stabilize}}$, therefore making it the cost function. The trajectory learned using this cost function leads to an increase in the feedbacks for the other two joint angles, as three joint angles are described between two actuators, as shown in Fig. 3.7. Consequently, we need a better cost function that does not lead to a wrong gait being learned, in case of underactuated systems like the biped.

The cost function chosen was the sum of squares of the three feedbacks, one for each joint angle. The cost function is minimized w.r.t. the weights of the rhythmic motor primitives for the torso's joint using the learning rule as per Eq. (3.6). In the biped gait experiment we force the torso to be at $\pi/6$ radians to the vertical axis, this is our stabilizing criterion. The initialization of the torso's forward trajectory is random, the stabilizing feedback makes sure that even with this random gait the torso does not fall over. The feedbacks therefore are based on this stabilizing criterion for the torso link and regular trajectory tracking condition for the other



(a) Initial forward trajectories for the torso.

(b) Final forward trajectory for the torso.

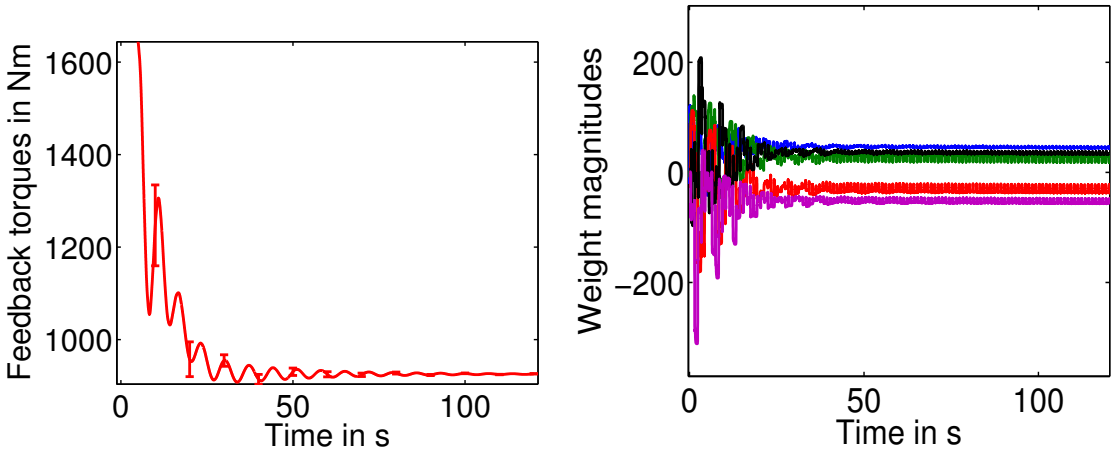
Figure 3.8.: (a) Initial forward trajectories for the torso: The dashed line is the system output and the continuous line is the forward trajectory. (b) Final forward trajectory for the torso: the two trajectories almost overlap proving that the forward gait is learned using feedbacks to converge to the sum of the control laws requirement. The convergence is not perfect because of the under-actuated system, however the periodic structure of the trajectories is the learned perfectly.

two links using the PD control in Eq. (2.7), and the sum of these feedbacks gives us the new cost function.

Results

It can be seen in the output results of Fig. 3.8 that the torso has learned the gait relatively well. The lower limbs maintain a gait of $-\pi/18$ to $+\pi/18$. As previously mentioned we minimize all the three squared feedback torques with respect to the torso's RMP weight parameters to achieve the best possible solution. As the sum of the feedbacks can not be reduced to zero, the upper torso does not perfectly converge to the required limit of $+\pi/6$, as can be seen in Fig. 3.8. The convergence in the two link manipulator case was perfect to the desired gait as the link to be stabilized had an actuator only for balance, which is not the case for the biped. Hence, a perfect output according to the control law is difficult to achieve in the biped, but our method also ensured that all the tasks are fulfilled at best without breaking down the system. Fig. 3.8 shows the forward gait being learned from a random initial forward gait of the torso. The learned gait has the same periodic structure, but differs in the baseline by a small margin.

The plot of reduction of summed absolute feedback torque is given in Fig. 3.9(a). It can be seen that the feedback is reduced considerably from the start time, because the gait has been learned. It can be seen that there is a considerable amount of learning in the first 40 s as the feedback reduces the most here and then reaches a minimum. After learning, the sum of the absolute feedback torques is almost halved from the initial value, and the standard deviation is almost zero. We have used 16 basis functions which are too many to plot, hence Fig. 3.9(b) shows the convergence of the first 5 weights. The goal state was learned for the RMPs as well. The weights converge within the first 60 cycles or



(a) Biped feedback in Nm. (b) Biped weight convergence

Figure 3.9.: (a) Moving average of the sum of feedbacks for the biped robot for a single run. Here the feedback does reduce and reaches a minimum when the learning stops. (b) The convergence plot of the first 5 weights amongst the 16 basis functions actually used. The weights converge within the first 60 cycles and the trajectory is almost learned within 100 cycles.

60 s correspondingly, after which the trajectory is almost completely learned and sum of feedbacks reaches its minimum.

This chapter dealt with our development of the modified FEL algorithm and the results obtained by using this method in learning balancing forward trajectories of a two link model and a biped's torso. The results show that this method can be used in learning forward trajectories and that this can be a viable solution in cases of online learning scenarios where gait changes while executing gaits might be required.

4 Conclusion and Future Work

In this thesis we described a novel method to learn balanced forward trajectories using feedback error learning. Our method allows us to learn trajectories online, while executing stable trajectories, using the stabilizing feedback of a rhythmic behavior. It learns trajectories without the need of complicated value functions or numerous supervised examples. Moreover, the modified FEL method also allows us to use low feedback gains to control our robots, in turn making the robots safe for use around humans.

We tested our method on two example problems: learning the forward balancing trajectory in a two link manipulator and gait adaptation in a biped robot. Our method gave impeccable solutions in both these example tasks. The method learned the forward trajectory flawlessly in the case of a fully actuated model, i.e. the two link manipulator, and with a small bias in case of an underactuated model, i.e., the biped robot. The bias in case of the underactuated model is the result of the minimization of all the feedback torques (instead of the stabilizing torque alone), which can not be completely minimized. In both the cases feedback torques were reduced drastically and learning of the forward trajectory was observed.

4.1 Future work

Our method currently functions in the joint space, i.e., joint space trajectories are learned and task space trajectories are derived from them using kinematics, even feedbacks are being calculated in the joint space. It would be easier to describe trajectories to be performed in the task space. This can be done using the framework of Operational Space Control. Using operational space control can allow us to use lower gains and still achieve a smooth control performance in the task space. We currently have a formulation to combine the modified FEL with operational space control, but learning in this scenario is proving to be difficult. We need to investigate this idea further. If the combination of modified FEL works with operational space control, then in future our method can help learning compliant control of manipulators with redundant degrees of freedom.

Bibliography

[1] A. Ijspeert, J. Nakanishi, and S. Schaal. Learning attractor landscapes for learning motor primitives. In *NIPS*, 2003.

[2] J. Peters and S. Schaal. Policy gradient methods for robotics. In *IROS*, 2006.

[3] J. Kober and J. Peters. Learning motor primitives for robotics. In *ICRA*, 2009.

[4] D. Pongas, A. Billard, and S. Schaal. Rapid synchronization and accurate phase-locking of rhythmic motor primitives. In *IROS*, 2005.

[5] S. Schaal, J. Peters, J. Nakanishi, and A. Ijspeert. Learning movement primitives. In *International Symposium on Robotics Research*, 2004.

[6] J. Nakanishi, J. Morimoto, G. Endo, G. Cheng, S. Schaal, and M. Kawato. Learning from demonstration and adaptation of biped locomotion. In *Robotics and Autonomous Systems*, number 2–3, 2004.

[7] M. Kawato. Feedback-error-learning neural network for supervised motor learning. *Advanced Neural Computers*, 1990.

[8] F. A. Mussa-Ivaldi. Modular features of motor control and learning. *Current Opinion in Neurobiology*, 9:713–717, 1999.

[9] G. Schoner and J. A. S. Kelso. Dynamic pattern generation in behavioral and neural systems. *Science*, 239:1513–1539, 1988.

[10] G. Schöner and C.M.P. Santos. Control of movement time and sequential action through attractor dynamics: A simulation study demonstrating object interception and coordination. In *SIRS 2001*, 2001.

[11] J. W. Milnor. Attractor. *Scholarpedia*, 1(11):1815, 2006.

[12] J. Nakanishi, J. Morimoto, G. Endo, G. Cheng, S. Schaal, and M. Kawato. A framework for learning biped locomotion with dynamic movement primitives. In *Humanoids*, 2004.

[13] D. Nguyen-Tuong and J. Peters. Model learning for robot control: a survey. *Cognitive Processing*, 12(4):319–340, 2011.

[14] M. Kawato and H. Gomi. A computational model of four regions of the cerebellum based on feedback-error learning. *Biological Cybernetics*, 68:95–103, 1992. 10.1007/BF00201431.

[15] J. Nakanishi and S. Schaal. Feedback error learning and nonlinar adapative control. In *Neural Networks*, number 10, 2004.

[16] M. H. Raibert. *Legged robots that balance*. Massachusetts Institute of Technology, Cambridge, MA, USA, 1986.

[17] E. Muybridge. *Animals in Motion*. Dover Publications, 1957.

[18] R. M. Alexander. The gaits of bipedal and quadrupedal animals. *International Journal of Robotics Research*, 3(2):49–59, 1984.

[19] A. Goswami, B. Thuilot, and B. Espiau. A study of the passive gait of a compass-like biped robot: Symmetry and chaos. *International Journal of Robotics Research*, 17:1282–1301, 1998.

[20] J. F. Schaefer and R. H. Jr. Cannon. On the control of unstable mechanical systems. *International Federation of Automatic Control*.

[21] K. Matsuoka. A mechanical model of repetitive hopping movements. *Biomechanisms*, 5:251–258, 1980.

[22] J. Morimoto and C. A. Atkeson. Minimax differential dynamic programming: An application to robust biped walking. 2003.

[23] J. W. Grizzle, G. Abba, and F. Plestan. Asymptotically stable walking for biped robots: Analysis via systems with impulse effects. *IEEE Transactions on Automatic Control*, 46(01), 2001.

[24] E. R. Westervelt, J.W. Grizzle, and D. E. Koditschek. Hybrid zero dynamics of planar biped walkers. *IEEE Transactions on Automatic Control*, 48:42–56, 2001.

[25] J. J. Craig. *Introduction to Robotics: Mechanics and Control*. Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA, 2nd edition, 1989.

[26] L. Bottou. Online algorithms and stochastic approximations. In David Saad, editor, *Online Learning and Neural Networks*. Cambridge University Press, Cambridge, UK, 1998.

[27] T. Yoshikawa. *Foundations of Robotics*. MIT Press, 1990.

[28] M. P. Deisenroth, R. Calandra, A. Seyfarth, and J. Peters. Toward Fast Policy Search for Learning Legged Locomotion. In *IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2012.

[29] J. Morimoto and C. G. Atkeson. Nonparametric representation of an approximated poincaré map for learning biped locomotion. *Auton. Robots*, 27(2):131–144, 2009.

A Two Link Robot Manipulator

This section details the Two link Manipulator's simulator shown in Fig A.1. First we describe the inverse dynamics equations for this system taken from [27].

$$\begin{aligned} u_1 = & \left(m_1 l_{g1}^2 + I_1 + m_2\left(l_1^2 + l_{g2}^2 + 2l_1 l_{g2} C_2\right) + I_2\right)\ddot{\theta}_1 \\ & + \left(m_2\left(l_{g2}^2 + l_1 l_{g2} C_2\right) + I_2\right)\ddot{q}_2 - m_2 l_1 l_{g2} S_2\left(2\dot{q}_1\dot{q}_2 + \dot{q}_2^2\right) \\ & + m_1 \hat{g} l_{g1} C_1 + m_2 \hat{g}\left(l_1 C_1 + l_{g2} C_{12}\right), \end{aligned} \tag{A.1}$$

$$\begin{aligned} u_2 = & \left(m_2\left(l_{g2}^2 + l_1 l_{g2} C_2\right) + I_2\right)\ddot{\theta}_1 + \left(m_2 l_{g2}^2 + I_2\right)\ddot{q}_2 \\ & + m_2 l_1 l_{g2} S_2 \dot{q}_1^2 + m_2 \hat{g} l_{g2} C_{12}, \end{aligned} \tag{A.2}$$

where u_i is the forward torque for the i^{th} link, m_i is the mass of the i^{th} link, l_i is the length of the i^{th} link, l_{gi} is the distance of center of mass from the i^{th} joint for the i^{th} link, I_i is the moment of inertia for the i^{th} link calculated by $I_i = m_i l_i^2/3$, q_i, $\dot{q}_i$ and $\ddot{q}_i$ are the joint angles and their first and second order derivatives for the i^{th} joint and $\hat{g}$ is the acceleration due to gravity. S_i and C_i are $\sin(\theta_i)$ and $\cos(\theta_i)$ respectively and S_i and C_{ij} are $\sin(\theta_i + \theta_j)$ and $\cos(\theta_i + \theta_j)$.

Since, multiple configurations of the two link manipulator were experimented with here two configurations would be discussed, with their chosen system parameters.

A.1 Two Link Manipulator: Standard Model

We describe the values of the system parameters of the standard manipulator with the Table A.1. It is standard because the masses and lengths are 1 kg and 1 m. Training this model was harder as it is potentially more unstable than the arm configuration discussed next. It needed a high sampling frequency of 100 Hz, this is because the pendulum's center of mass for each link was relatively far which increased the instability of the system while training. The training itself was initially done using trajectories represented as a linear function of basis functions alone, and not as an RMP. This was done ensure that the parameters of feedback gain and learning rates are right before training the RMP, because RMPs are relatively harder to train rather than basis functions. RMPs have the extra parameter of the goal state to be learned that has different learning rates than the other weights of the RMPs. The goal states were observed to perform better when learned slowly, i.e., with lower learning rates.

Some snapshots of the visualization of the simulator are shown in Fig. A.2. The Fig. A.2 shows half an oscillation cycle of the two link manipulator before and after the learning is complete. Initially the double link's forward trajectory and

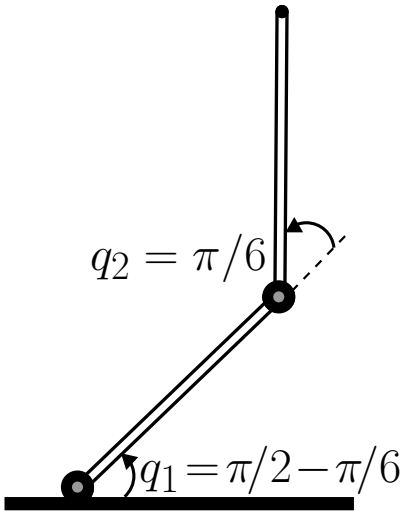


Figure A.1.: Model for the two link robot manipulator in the balanced state

System Parameter	Value with units
m_1	1 kg
l_1	1 m
l_{g1}	0.5 m
m_2	1 kg
l_2	1 m
l_{g2}	0.5 m
$\hat{g}$	9.8 m/s^2
Sampling Frequency	100 Hz
Oscillation Frequency	1 Hz

Table A.1.: System parameters for the standard two link manipulator experiment

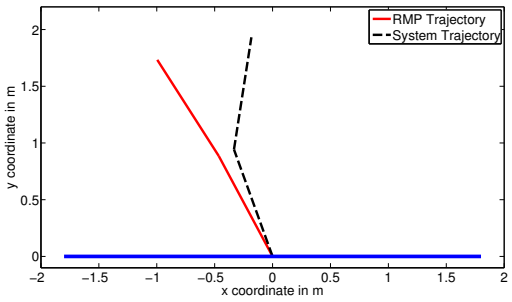
System Parameter	Value with units
m_1	2 kg
l_1	0.5 m
l_{g1}	0.25 m
m_2	2 kg
l_2	0.5 m
l_{g2}	0.25 m
$\hat{g}$	9.8 m/s^2
Sampling Frequency	50 Hz
Oscillation Frequency	1/3 Hz

Table A.2.: System parameters for the arm link two link manipulator experiment

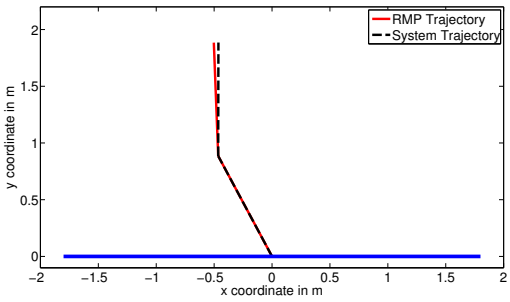
the trajectory executed is completely different, as the executed trajectory is being corrected by the control laws for stability. The trajectories converge as learning is complete, making the feedback zero, and driving the two link manipulator based on forward torque alone.

A.2 Two Link Manipulator: Human Arm Model

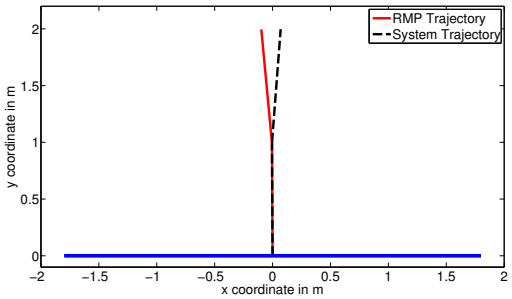
This configuration of the two link manipulator is much closer to a human arm, with similar mass and lengths. The system parameters are given in the Table A.2. This model is considerably slower in completing its oscillation compared to the standard two link model. It is heavier and the lengths of the links are much shorter than that of the standard two link model. This mass and length distribution makes this arm like model compact and easier to control that the standard model. Hence, it learns at a much faster rate than the standard model and the learning is complete within the first 20 cycles, which is about 60 s. This model was also trained initially with basis functions to ensure that the feedback gains were precise, and then it was switched over to RMPs as the final goal is to learn RMP modeled trajectories. The Fig. A.3 shows the snapshots of the simulator for the two link arm model. The trajectory of the arm model was an up and down motion, as seen in the Fig. A.3, this was done to just to test the arm in different configurations. The Fig. A.3 shows a total overlap of the forward and the system trajectory after learning.



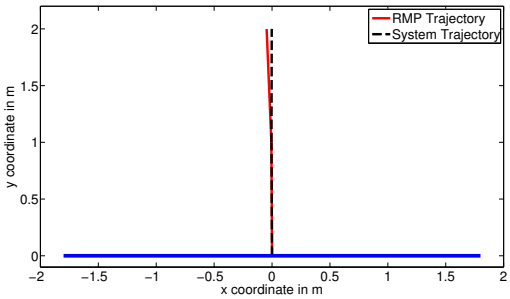
(a) Initial Trajectory 1.



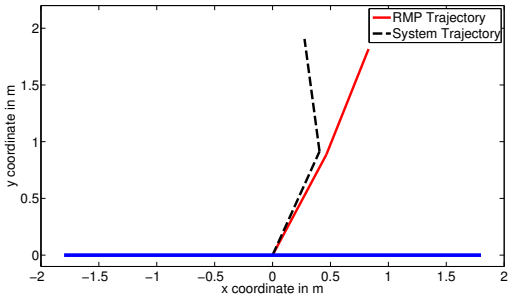
(b) Final Trajectory 1.



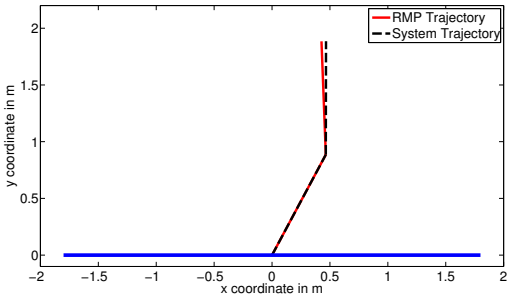
(c) Initial Trajectory 2



(d) Final Trajectory 2

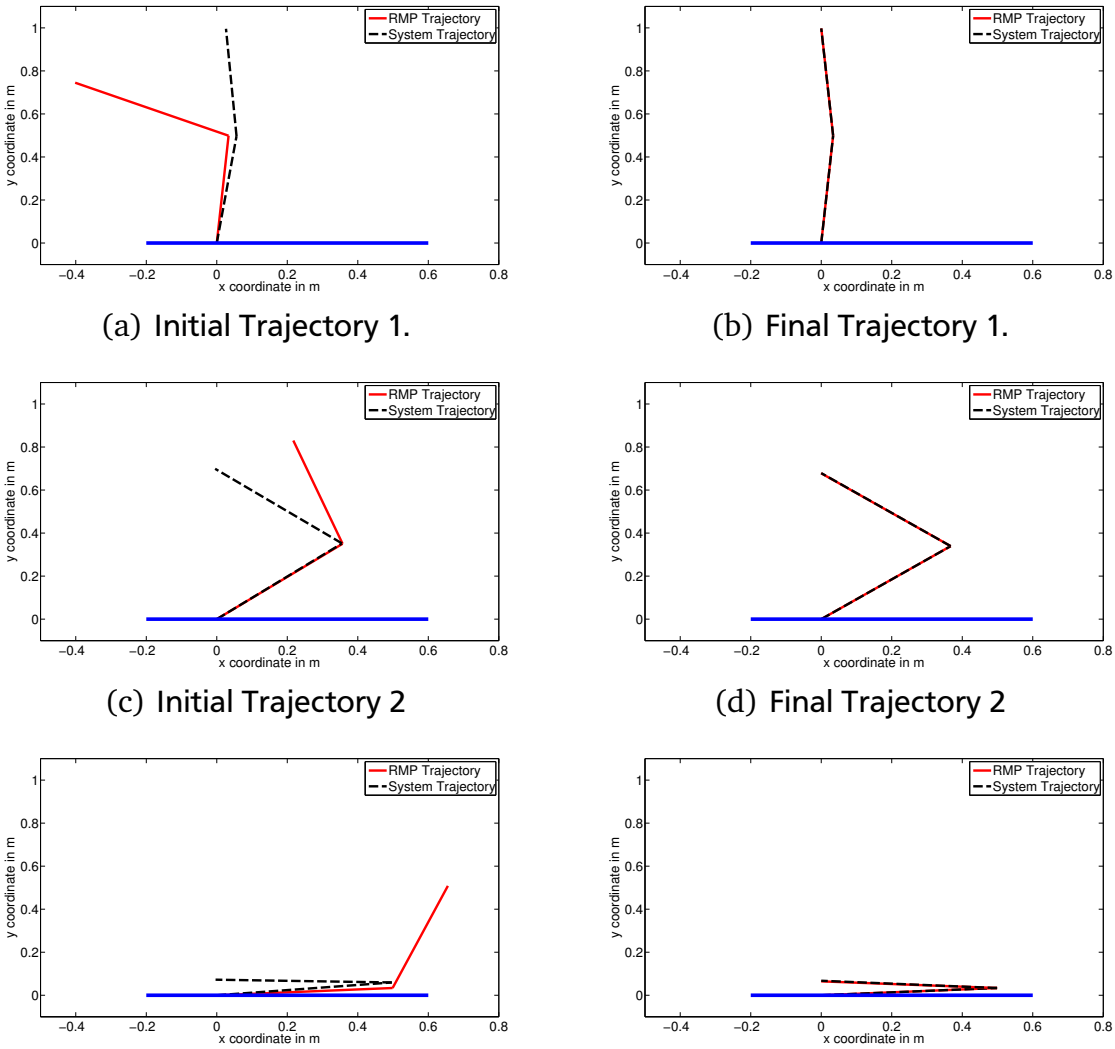


(e) Initial Trajectory 3



(f) Final Trajectory 3

Figure A.2.: Simulator visualization for the experiment. The oscillation of the two link manipulator before learning the forward trajectory learning is shown be the first column (a), (c) and (e). It can be seen that the RMP trajectory or the forward trajectory of the two link manipulator is not stable and not following the control law. After learning the forward trajectories are stable and follow the control law precisely. These learned trajectories are shown in the second column (b), (d) and (f). The trajectory of the second link is converged to that of the first link, which reduced the feedback to zero.



(a) Initial Trajectory 1.

(b) Final Trajectory 1.

(c) Initial Trajectory 2

(d) Final Trajectory 2

(e) Initial Trajectory 3

(f) Final Trajectory 3

Figure A.3.: Simulator visualization for the two link experiment modeled after a human arm. The oscillation of the two link manipulator before learning the forward trajectory learning is shown be the first column (a), (c) and (e). It can be seen that the RMP trajectory or the forward trajectory of the two link manipulator is not stable and not following the control law. After learning the forward trajectories are stable and follow the control law precisely. These learned trajectories are shown in the second column (b), (d) and (f). The trajectory of the second link is converged to that of the first link, which reduced the feedback to zero. The learning takes about 20 cycles or 60 s to complete.

B Biped Simulator

The biped simulator was taken from Grizzle's work in [23]. This model simple biped walker with just 3 links, two legs and a torso. The orientation of the angles of the biped is given in Fig. B.1. The inverse dynamics equation of the biped is given by

$$\boldsymbol{Bu} = \mathbf{M}(\boldsymbol{q})\ddot{\boldsymbol{q}} + \mathbf{c}(\boldsymbol{q}, \dot{\boldsymbol{q}}) + \mathbf{g}(\boldsymbol{q}) \,, \tag{B.1}$$

where $\boldsymbol{q} = (q_1, q_2, q_3)$ denote joint angles, and $\dot{\boldsymbol{q}}$ and $\ddot{\boldsymbol{q}}$ are the velocities and acceleration, respectively of these joints. The inertia matrix is given by $\mathbf{M}(\boldsymbol{q})$, $\mathbf{c}(\boldsymbol{q}, \dot{\boldsymbol{q}})$ denotes centripetal and Coriolis forces, $\mathbf{g}(\boldsymbol{q})$ denotes the gravity forces, $\boldsymbol{u}$ is the forward torque, and $\boldsymbol{B}$ is the gain matrix.

The inertia matrix $\mathbf{M}(\boldsymbol{q})$ is given by

$$\mathbf{M}(\boldsymbol{q}) = \begin{bmatrix} (\frac{5}{4}m + M_{\mathrm{H}} + M_{\mathrm{T}})r^2 & -\frac{1}{2}mr^2c_{12} & M_{\mathrm{T}}rlc_{13} \\ -\frac{1}{2}mr^2c_{12} & \frac{1}{4}mr^2 & 0 \\ M_{\mathrm{T}}rlc_{13} & 0 & M_{\mathrm{T}}l^2 \end{bmatrix} , \tag{B.2}$$

where m is the mass of the legs, M_{H} is the mass of the hip, M_{T} is the mass of the torso, r is the length of the legs of the biped and l is the length of the torso of the biped. $c_{ij} = \cos(\theta_i - \theta_j)$ and $s_{ij} = \sin(\theta_i - \theta_j)$. The Coriolis forces matrix is given by

$$\mathbf{c}(\boldsymbol{q}, \dot{\boldsymbol{q}}) = \begin{bmatrix} 0 & -\frac{1}{2}mr^2s_{12}\dot{\theta}_2 & M_{\mathrm{T}}rls_{13}\dot{\theta}_1 \\ \frac{1}{2}mr^2s_{12}\dot{\theta}_1 & 0 & 0 \\ -M_{\mathrm{T}}rls_{13}\dot{\theta}_1 & 0 & 0 \end{bmatrix} , \tag{B.3}$$

where $\dot{\theta}_i$ is the time derivative of joint angle θ_i. The gravity matrix $\mathbf{g}(\boldsymbol{q})$ is given by the expression

$$\mathbf{g}(\boldsymbol{q}) = \begin{bmatrix} -\frac{1}{2}g(2M_{\mathrm{H}} + 3m + 2M\mathrm{T})r\sin(\theta_1) \\ \frac{1}{2}gmr\sin(\theta_2) \\ \frac{1}{2}gM_{\mathrm{T}}l\sin(\theta_3) \end{bmatrix} , \tag{B.4}$$

and the gain matrix $\mathbf{B}$ is given by the expression

$$\mathbf{B} = \begin{bmatrix} -1 & 0 \\ 0 & -1 \\ 1 & 1 \end{bmatrix} . \tag{B.5}$$

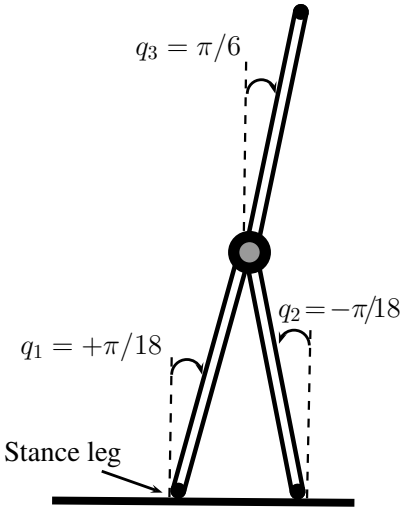
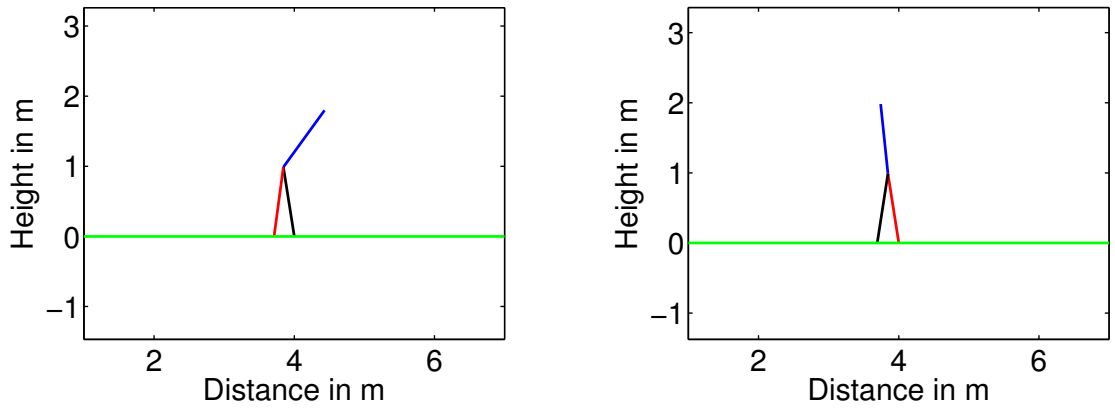


Figure B.1.: Model for the Biped robot in the balanced state.

System Parameter	Value with units
m	15 kg
r	1 m
l	1 m
M_{T}	40 kg
M_{H}	10 kg
$\hat{g}$	9.8 m/s^2
Sampling Frequency	200 Hz
Walk cycle Frequency	1 Hz

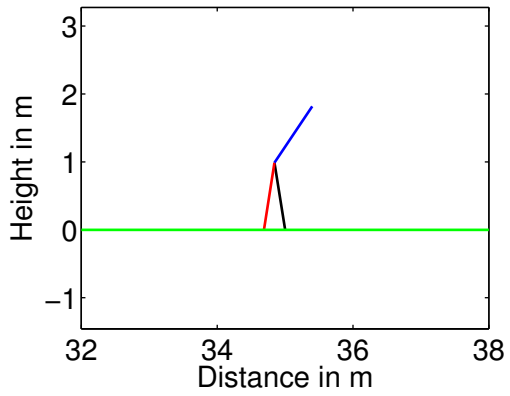
Table B.1.: System parameters for the biped experiment

The values of these system parameters were chosen to be close to that of the human body. The Table B.1 gives the values of the system parameters of the biped model. Fig. B.2 shows the visualization of the biped simulator. It can be seen that the initial trajectory of the system and the forward torso trajectories are not similar at all in Fig. B.2(a) and Fig. B.2(b) respectively. The forward trajectory is random and seems unstable with the torso arching backwards in Fig. B.2(b) and the system trajectory of the torso is controlled by a control law that keeps the torso in balance. The control law force the torso to be at $\pi/6$ radians to the vertical axis. This stable trajectory is learned by the RMPs completely after about 60 cycles corresponding to 60 s. The second row of plots in Fig. B.2 compares the trajectories after learning, with Fig. B.2(c) being the system output and Fig. B.2(d) being the forward trajectory given by the RMPs.

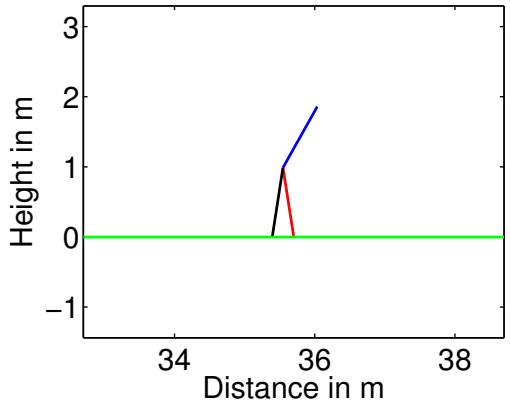


(a) Initial System Trajectory.

(b) Initial Forward Trajectory .



(c) Final System Trajectory



(d) Final Forward Trajectory

Figure B.2.: Simulator visualization for the biped robot. The first row shows the biped using (a) initial System Trajectories and (b) the Forward RMP Trajectories, at the same instant of time. The torso's position in the two figures are very different. The System Trajectories because of the feedback produce a stable orientation of the torso which was defined to be at $\pi/6$ to the vertical. Whereas, the forward trajectories are untrained thus giving unstable torso orientation, trying to arch backwards. After learning in about 120 s the biped's (c) System Trajectory (d) and RMP Forward Trajectory are almost equal at the same instant of time. This shows the modified FEL'S potential to learn trajectories online.