MANCHESTER 1824
The University of Manchester

Jumping Dynamics of Animals and Robots

A thesis submitted to the University of Manchester for the degree of
Doctor of Philosophy
in the Faculty of Science and Engineering

2021

Conor Zachary Andrew Marsh
Department of Mechanical, Aerospace and Civil Engineering

Contents

Word Count: 53,313

List of figures

List of tables

Nomenclature

The units used in this work are all S.I. units, e.g. metres, seconds, kilograms, radians etc.

Algebraic Notation

$\ddot{a}$	Double dot denotes the second derivative of a with respect to time
$\dot{a}$	Dot denotes the derivative of a with respect to time
$\frac{\partial x}{\partial y}$	Denotes the partial derivative of x with respect to y
det$(\boldsymbol{A})$	The determinant of the matrix $\boldsymbol{A}$
N$(\boldsymbol{A})$	The null space of the matrix $\boldsymbol{A}$
proj$_{\boldsymbol{b}}\boldsymbol{a}$	Projection of the vector $\boldsymbol{a}$ onto $\boldsymbol{b}$
$\boldsymbol{A}$	Bold upper case letter denotes a matrix
$\boldsymbol{A}^{+}$	The Moore-Penrose inverse (pseudo-inverse) of the matrix $\boldsymbol{A}$
$\boldsymbol{A}^{-1}$	The inverse of the matrix $\boldsymbol{A}$
$\boldsymbol{A}^{T}$	The transpose of the matrix $\boldsymbol{A}$
$\|\boldsymbol{a}\|$	Euclidean norm (length or magnitude) of the vector $\boldsymbol{a}$
$\boldsymbol{a}$	Bold lower case letter denotes a vector

Symbols

$\boldsymbol{J}$	Jacobian matrix of a system
π	Decision making policy
$\boldsymbol{q}$	Vector of generalised joint angles
$\boldsymbol{x}$	Vector of Cartesian body position and orientation
F	Force
g	Acceleration due to gravity, 9.81 ms^{-2}
I	Inertia
l	Length
m	Mass

Abstract

Jumping is useful as a precursor for flight and arboreal locomotion, as well as locomotion through intermittent terrain. This thesis aims to answer the questions: What insights can be gained through the application of predictive modelling methods to novel contexts in jumping dynamics problems, and how should the fitness of different predictive modelling techniques be assessed?

This thesis uses the metrics: predictive power, computational cost, and intellectual investment, to assess predictive modelling methods for use in jumping research.

Analysis of model equations is rarely found applied to multiple degree of freedom jumping models in the literature. This work explores such applications of analysis and determines that the method is capable of providing insights into low and high complexity models.

The work identifies static optimisation for the resolution of second order kinematic redundancy and reinforcement learning as predictive modelling methods which are not currently used in jumping research. The results of this work imply that static optimisation could be made viable for providing insights into jumping dynamics problems. However, the static optimisation method is demonstrated to produce results which are comparable to experimentally measured data found in the literature and may be a useful method in jumping applications. The method is limited in that it does not accommodate the physical constraints of jumping systems.

Deep Deterministic Policy Gradients, a reinforcement learning method, is shown to also violate physical constraints. The method is demonstrated to be unreliable when applied to jumping dynamics problems with 32% of optimisation runs failing to achieve any positive rewards.

Dynamic optimisation is assessed on its predictive power, computational cost, and intellectual investment and is deemed to require considerable intellectual investment to use.

As well as insights into the methods used to solve jumping problems, this work provides technical contributions of: 1) demonstration of a novel method used to determine the dynamic balancing capability of a jumping system and 2) empirical demonstration of a second order kinematic simulation method with results comparable to those found in the literature.

Declaration of originality

I hereby confirm that no portion of the work referred to in the thesis has been submitted in support of an application for another degree or qualification of this or any other university or other institute of learning.

Copyright statement

i The author of this thesis (including any appendices and/or schedules to this thesis) owns certain copyright or related rights in it (the "Copyright") and s/he has given The University of Manchester certain rights to use such Copyright, including for administrative purposes.

ii Copies of this thesis, either in full or in extracts and whether in hard or electronic copy, may be made *only* in accordance with the Copyright, Designs and Patents Act 1988 (as amended) and regulations issued under it or, where appropriate, in accordance with licensing agreements which the University has from time to time. This page must form part of any such copies made.

iii The ownership of certain Copyright, patents, designs, trademarks and other intellectual property (the "Intellectual Property") and any reproductions of copyright works in the thesis, for example graphs and tables ("Reproductions"), which may be described in this thesis, may not be owned by the author and may be owned by third parties. Such Intellectual Property and Reproductions cannot and must not be made available for use without the prior written permission of the owner(s) of the relevant Intellectual Property and/or Reproductions.

iv Further information on the conditions under which disclosure, publication and commercialisation of this thesis, the Copyright and any Intellectual Property and/or Reproductions described in it may take place is available in the University IP Policy (see `http://documents.manchester.ac.uk/DocuInfo.aspx?DocID=24420`), in any relevant Thesis restriction declarations deposited in the University Library, The University Library's regulations (see `http://www.library.manchester.ac.uk/about/regulations/`) and in The University's policy on Presentation of Theses.

Acknowledgements

I would like to thank my supervisors Prof. Bill Crowther and Dr. Ben Parslew for all of the support and guidance they have provided, both academically and in life. The patience they both had for me was phenomenal and greatly appreciated.

I am grateful for the support and encouragement of my parents, Judy Forrister and Andrew Marsh. They always made me feel like I could achieve anything.

I would also like to thank Prof. Mike Barnes for his help and guidance early in the work while I found my footing in the world of research.

Thank you to all the researchers in the F46 and D46 labs in Sackville St. building for all the talks, support and the lunch time gaming.

Chapter 1

Introduction

Legged systems are able to walk, run, and jump, enabling them to traverse terrains of varying heights and gaps. Such capabilities are particularly desirable for search and rescue robots, as well as systems designed to explore harsh terrains including caves and extra-terrestrial lava tubes. The ability to jump can provide a system with means to cross gaps in terrain as well as providing sufficient velocity for a flight take-off in the case of birds. Research into jumping systems and the motions they produce may also provide insight to the field of athletics, aiding with physiotherapy, performance, and prosthetics. A jump may be divided

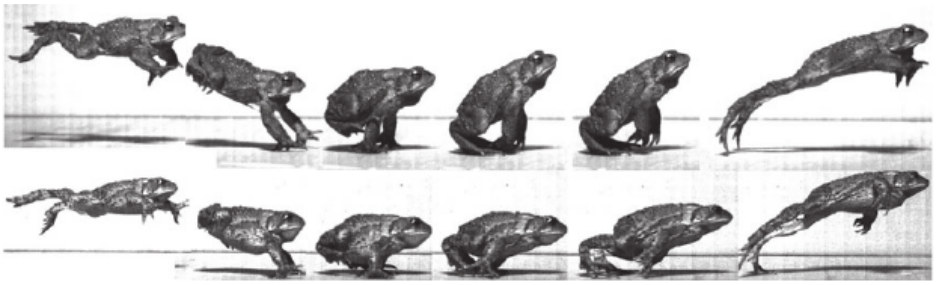


Figure 1.1: Video frames of a jumping toad (*Rhinella marina*) taken from work published by Reilly et al. [1]. The frames show the progression of jumping stages from flight (left) to landing (left-centre) to push-off (right-centre) to flight again (right).

into sequential stages, which are: preparation, push-off, flight, and landing. The take-off event is defined between push-off and flight as the moment the system breaks contact with the substrate it is jumping from. Similarly, the touch-down event occurs after the flight stage, when the system makes contact with a surface. The scope of this work is concerned with the push-off stage and the take-off event. The flight stage may consist of ballistic motion of the system, or may be extended by aerodynamic features such as wings, in which case it may be considered as a separate motion from the preceding jump. A system's ability to withstand touch-down and landing may impose requirements on push-off and take-off. When landing, a jumping system may use elastic storage mechanisms to reuse some of the energy from the previous jump to perform another jump. This case of continuous hopping generally ties better to running than one-off jumping and is modelled using the spring-loaded inverted pendulum [2]. Such intricacies of jumping are not considered here.

With peak forces that are often many times greater than the weight of the system, jumping systems require effective structural and mechanical design to both produce and withstand such forces. Biological systems with articulated legs, such as humans and birds, have rich and complex configuration spaces; often too rich to exhaustively search when determining ideal

motions for the system. Yet many of these systems demonstrate a mastery of balance and control when executing dynamic jumping motions. As such, some researchers study the exemplary motions produced by biological systems [3]–[10]. To properly measure the motions of living subjects generally requires an extensive experimental process [11]. Other researchers use predictive modelling methods, such as dynamic optimisation, to determine motions using models of jumping systems [12]–[19]. While the kinematic and dynamic models used in both of these approaches are generally similar, the methods used and insights gained vary. This work investigates the use of predictive modelling methods in solving jumping problems using planar, segmented models of jumping systems ranging from 1 to 5 degrees of freedom. The investigated methods include: analysis of model equations, static optimisation of kinematics, dynamic optimisation (aka trajectory optimisation), and reinforcement learning. Model-free methods from reinforcement learning are identified in this work as potentially useful methods for jumping motion data procurement which are rarely found in jumping literature.

Predictive modelling involves the use models of a system to produce data for that system. This provides an alternative to measurement of data directly from a system of interest. Methods of predictive modelling, such as optimisation, have the potential to reveal high performance behaviours for jumping systems. These methods The implementation and use of predictive modelling methods may be hindered by the intellectual investment or potentially large amounts of computational resources required in their use. Other reasons for their infrequent use may be that their predictions are brittle, or dependent on the models used rather than the system itself. The following predictive modelling methods are investigated:

1. analysis of model equations,
2. static optimisation,
3. dynamic optimisation,
4. reinforcement learning.

The methods are assessed in the context of jumping dynamics problems using the following criteria:

- Predictive Power - How flexible the method is, the range and utility of the results it provides, and the limits in model complexity for the method.
- Computational Cost - The time it takes for a commercial, high spec desktop computer to carry out the method.
- Intellectual Investment - The domain knowledge required from the user of the method, in both the problem and the method.

1.1 Research Problem and Questions

The use of predictive modelling techniques for solving jumping dynamics problems is well established, yet some predictive modelling methods are not used in jumping literature. Many predictive modelling methods are complex and difficult to apply without expert knowledge, and it is difficult for non-experts to make rational choices on what methods to apply.

The questions which arise from this problem are:

1. What insights can be gained through the application of predictive modelling methods to novel contexts in jumping dynamics problems?
2. How should the fitness of different predictive modelling techniques be assessed?

1.2 Thesis Aim and Objectives

The aim of this thesis is to provide a critical assessment of the use of predictive modelling methods in jumping dynamics research.

The objectives of this work are to:

1. investigate the application of predictive modelling methods, which are not already used in jumping literature, to jumping problems
2. assess the predictive power, computational cost, and intellectual investment of predictive modelling methods
3. facilitate the decision making process of researchers wanting to use predictive modelling methods through the comparison of common methods
4. demonstrate best practice for using a variety of predictive modelling methods through a series of case studies from biomechanics and robotics

1.3 Thesis Outline

Literature from jumping research which utilised predictive modelling methods is reviewed. The predictive power, computational cost, and intellectual investment of the methods are discussed. Patterns in the usage of the methods are also identified. The literature review reveals uncommon usages for some predictive modelling methods which this thesis intends to investigate.

Chapter 3 provides formalisation of an analysis of model equations method found in the literature [17], [20] and demonstrates the novel application of the method to jumping systems models with multiple degrees of freedom. The chapter also uses analysis of model equations to derive the novel concept of a dynamic balance ellipsoid.

Following this, Chapter 4 investigates the use of static optimisation for the resolution of kinematic redundancy in the novel application of jumping, which was found to be unused in jumping research in the literature review. The work uses a model of the guinea fowl (*Numida meleagris*) found in the literature [6]. Chapter 5 explores the use of dynamic optimisation for predictive modelling using the case study model used in Chapter 4. Chapter 6 investigates the application of reinforcement learning for predictive modelling of jumping systems.

The methods studied in each of the technical chapters are also assessed on their predictive power, computational cost, and intellectual investment.

1.4 Research Methods in Jumping Dynamics

Jumping is a behaviour studied in the fields of biomechanics, robotics, and optimal control. Each field of study typically seeks to obtain different insights into jumping and so use different methods. This section provides a brief overview of the methods used in each field.

Biomechanics research primarily focuses on explanatory modelling of existing biological systems [11]. This generally involves collection of experimentally measured data from biological systems. The data are then used alongside models of the system to explain how those behaviours might be achieved. For example, studies seek to determine whether a jumping motion is possible to achieve using only muscles, or whether additional features, such as elastic storage, are required [6]. Explanatory modelling is a strong focus of biomechanics. Such an endeavour requires collection or synthesis of motion data and may benefit from predictive modelling methods, providing the data they produce can be validated.

Robotics and optimal control studies typically use optimisation to produce desirable motions of systems through open or closed loop control [21]. These fields often use detailed mathematical notation which can be time consuming to interpret. Closed loop control is the general focus in robotics and optimal control as this enables adaption of the system to errors and noise. In the endeavour of robust control, many methods are designed to consider various eventualities of a system and may prove excessive in dynamic motion studies, unless the system's control strategy is of particular interest. Another focus of robotics includes design, often related to manipulators and the space in which they are fixed during operation. Manipulators are usually articulated robotic systems, similar in topology to arms and legs. Their dynamic equations of motion are often very similar to those used in jumping studies and methods used in manipulator design may also be applied in jumping research.

Some research in biomechanics is tending towards "model-free" methods, which do not require knowledge of the model equations. Model-free methods may require less intellectual investment as the researcher is not required to derive or understand the equations of motion for models of the systems they are studying. Reinforcement learning contains a set of model-free optimisation methods which may be suitable in such applications [22]. While most reinforcement learning methods are limited to discrete decision processes, some state of the art applications of reinforcement learning have been demonstrated in continuous control appli-

cations [23]. Reinforcement learning uses the Markov Decision Process framework, in which state transitions are defined as probabilities rather than the rates of change used in dynamics. Accommodation to these probabilistic transitions often leads to a probabilistic solution, i.e. the resulting controller samples its control actions from an optimised probability distribution. Examples of state of the art stochastic policy methods include Trust Region Policy Optimisation [24], Proximal Policy Optimisation [25], and Soft Actor-Critic [26]. This work considers the Deep Deterministic Policy Gradient method [23] which is derived from the deterministic policy gradient proposed by Silver [27] as it is one of the few reinforcement learning methods which produces a deterministic policy while also accommodating continuous control actions.

1.5 Predictive Modelling

Predictive modelling methods use models of a system to determine possible behaviours or performances of the system. The ability to accurately predict behaviours of a system can allow for research of animals without the need to experimentally measure their behaviours. Predictive modelling also enables research into systems which do not exist, including extinct animals and hypothetical systems. There is a catch: validation of predictions generally involves comparison with a ground truth, which, in the case of animals, means experimentally measuring behaviours to compare the predicted behaviours against. In some contexts, it may be sufficient to validate the steps which lead to a prediction and trust that those steps combine to also create a valid prediction.

It is common practice to obtain example motions of a model from experimentally measured data and to then use optimisation on an inverse dynamics model to predict actuation efforts used by the system [3], [4], [6]–[8], [11]. When experimental data are unavailable, the generation of jumping motion data is typically done using dynamic optimisation methods from optimal control theory on forward dynamics models. Dynamic optimisation uses a scalar objective, such as jump height, to determine the optimal motion of the model [9], [12], [15], [16], [28]. These two methods of data collection, experimental and predictive, may be considered as alternate methods for obtaining ideal jumping motions of a given system [11].

In robotics, predictive modelling is typically used to design systems and in development (and implementation) of controllers. In design, parameters of the model are optimised to maximise metrics such as the dexterity, working space, or maximum acceleration of a manipulator [29]; or the agility of a jumping robot [30]. Dynamic optimisation generally uses a forward dynamics model of the system and attempts to find control (force and torque) inputs which shape the dynamics into desirable behaviours. Model predictive control is a controller implementation which performs dynamic optimisation to select the next control action for the system over regular intervals during operation. At each interval the optimised control is applied and another optimisation is performed in a sliding window fashion. The optimised trajectory typically looks further ahead in time than the optimisation intervals occur.

The methods described above require models of the system. Model-free methods, such as

reinforcement learning or genetic algorithms, work with data collected from a system instead of the model equations. This is significant as model-free methods may be applied to real-world systems which may be too complex to model and simulate. Similarly, model-free methods may be applied to simulations without the user being required to be aware of the model and equations used in the simulation, this includes dynamic simulation applications such as GaitSym [31] and OpenSim [32].

Predicting how a system might behave does not directly provide an explanation for the behaviour and gives no notion of how the system's performance may be improved. However, the predicted behaviour may be used as data in explanatory modelling, which seeks to find such relationships between a system parameters and its performance. In this work, predictive modelling methods are considered for finding feasible, or optimal, motions of jumping systems. The following methods are investigated:

1. Analysis of model equations,
2. Static optimisation,
3. Dynamic optimisation,
4. Reinforcement learning.

When deciding which of the above methods to use, one must take multiple factors into consideration. Predictive modelling methods vary in the models they may be applied to. It is uncommon to find analysis of model equations for systems with many degrees of freedom; behaviours of such systems are generally obtained by experimentally measuring data or through optimisations which require significant computational resources. Some methods of predictive modelling are flexible in the sense that they may be used to determine many factors of a system's behaviour, while others, such as static optimisation, are limited to a smaller subset of behaviours.

Jumping, dynamics, biology, algebra, and optimisation are all different subjects. While experts may have diverse pools of knowledge and experience to draw from, the time and resources required to study all of these subjects should be considered when deciding to pursue research into jumping dynamics using predictive modelling methods. Each of the methods require different levels of domain knowledge, both for the implementation and use of the method. For example, model-free methods, such as reinforcement learning, are feasible to apply to a black box system or simulation, requiring no knowledge about the workings of the system (although such knowledge may still be beneficial in practice).

1.6 Definitions and Assumptions

This section intends to clarify the terminology and assumptions used in this work.

1.6.1 Planar Jumping System

The complexity of model necessary for a problem depends on many factors. It is preferable to select the minimum order model while maintaining a suitable level of precision and accuracy in replicating the system. A considerable reduction in the order of jumping models can be made by looking at only planar motion of the system. A good approximation of bipedal locomotion can be made by looking at movement in the sagittal plane as most systems move both their bodies and legs predominantly in this plane during jumping [33], the same strategy may be applied to jumping systems.

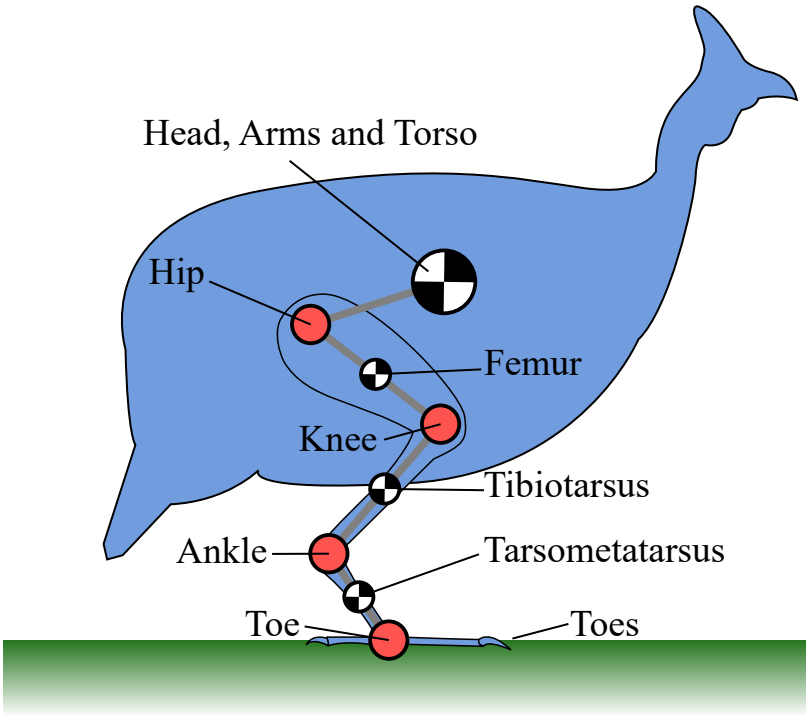


Figure 1.2: Example planar jumping model of *Numida meleagris*. Each segment is treated as a rigid body with mass and inertia. Segments are connected by revolute joints (orange). The diagram is for illustrative purposes only and is not drawn to scale. Centre of mass indications are drawn smaller for leg segments than the body but are not indicative of relative mass or inertia.

The planar models in this work consist of segments connected by revolute joints. Each joint represents a degree of freedom in the model as shown in figures 1.2 and 1.3. The inertial frame of reference for all models is aligned such that gravity acts in the negative vertical, y, direction. The ground is assumed to be flat and aligned with the horizontal, x, direction. The origin for all models is set to the point at which the leg, or foot, meets the ground coinciding with the position of the first revolute joint as shown in figure 1.3. Each degree of freedom in a model is described by the angle made by the segment attached to the joint and the horizontal ground plane. The angles of each joint are defined relative to the horizontal inertial axis i.e. the ground angular displacement. Many examples in manipulator literature define angles using the angular displacement of a segment relative to the previous adjacent segment, meaning that, when the angle is zero (or an integer multiple of π), the two segments are aligned. The use of angles which are relative to the inertial frame produces a simplified set of equations of motion compared to the use of relative joint angles used in robotics as no fictitious forces need

to be included in the equations. The definition of angles using adjacent segments is necessary when angles are measured by sensors inside the robot, such as encoders. In biomechanics, the measurement of angles in motion capture is usually performed by a set of cameras in the inertial frame, thus allowing an inertial angle to be defined relative to the ground.

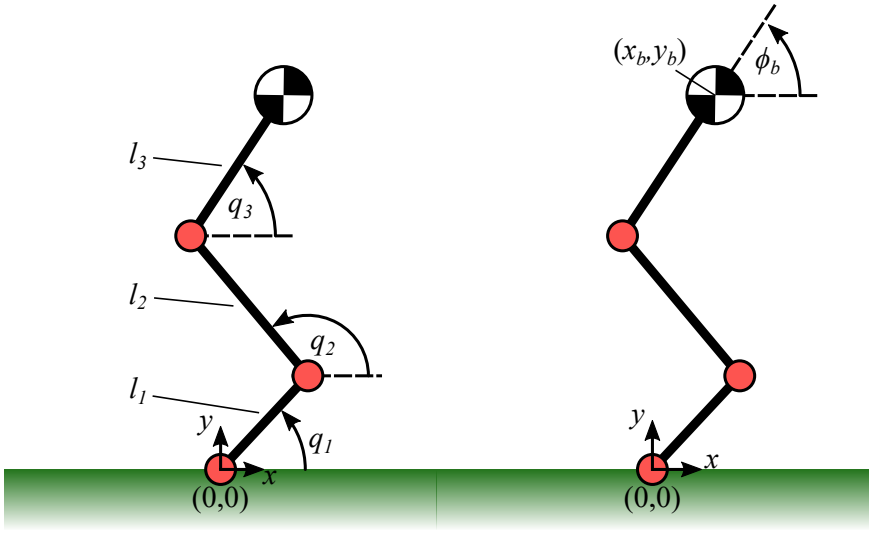


Figure 1.3: (Left) The joint angles and segment lengths for an example 3 degree of freedom model and (right) the body pose of an example 3 degree of freedom model.

The joints in a model are numbered in ascending order, starting with the joint connecting the leg to the ground and working along the leg towards the final joint connecting to the body as shown in figure 1.3. The angles of the joints are given by the vector $\boldsymbol{q}$ while $\dot{\boldsymbol{q}}$ and $\ddot{\boldsymbol{q}}$ represent the joint velocities and accelerations respectively. The lengths of each segment are given by the vector $\boldsymbol{l}$. The generalised coordinates of a model are the minimal set of coordinates which fully describe the model's pose. For all models in this work the generalised coordinates are the vector of joint angles, $\boldsymbol{q}$. The complete configuration, or state, of the model is given by the joint angles and velocities, $\boldsymbol{q}$ and $\dot{\boldsymbol{q}}$. While the generalised coordinates describe the pose of the entire model, they are not easily understood and quickly become intractable when plotted as time series, especially for systems with many degrees of freedom. It is more intuitive to look at the motion of the body, which is attached to the open end of the chain, or leg. The pose of the body, $\boldsymbol{x}$, provides the body's horizontal and vertical position, x_b and y_b and orientation ϕ_b as shown in figure 1.3. The segment lengths, joint angles and body pose for a model with n degrees of freedom are:

$$\boldsymbol{l} = \begin{bmatrix} l_1 \\ l_2 \\ \vdots \\ l_n \end{bmatrix}, \quad \boldsymbol{q} = \begin{bmatrix} q_1 \\ q_2 \\ \vdots \\ q_n \end{bmatrix}, \quad \boldsymbol{x} = \begin{bmatrix} x_b \\ y_b \\ \phi_b \end{bmatrix}. \tag{1.1}$$

1.6.2 Defining a Jump

This section provides a definition of a jump. Jumping is an atypical method of locomotion as it is not a cyclic motion like walking, running or flying. Note that this work is not concerned

with cyclic jumping motions, also known as hopping. As previously mentioned, this work considers the push-off stage and take-off event of a jump. For this explanation, consider a jumping system standing on flat ground with gravity acting down, perpendicular to the ground as shown in figure 1.4. The preparation stage involves the system moving into an ideal state from which to begin the high effort push-off stage, this typically entails a crouching motion. The push-off stage is when the system applies a force greater than its weight to the ground, receiving a reaction force which accelerates the system's centre of mass. As previously mentioned, the jump concludes with the system breaking contact with the ground. While jumps may be performed from many surfaces, it is discussed in this work with reference to the ground. It is assumed that the ground is solid and does not deform during the jumping motion. Many sources in the literature define the point of take-off in a jumping motion as the moment when the foot breaks contact with the ground [12], [17], [20], [34]. By assuming

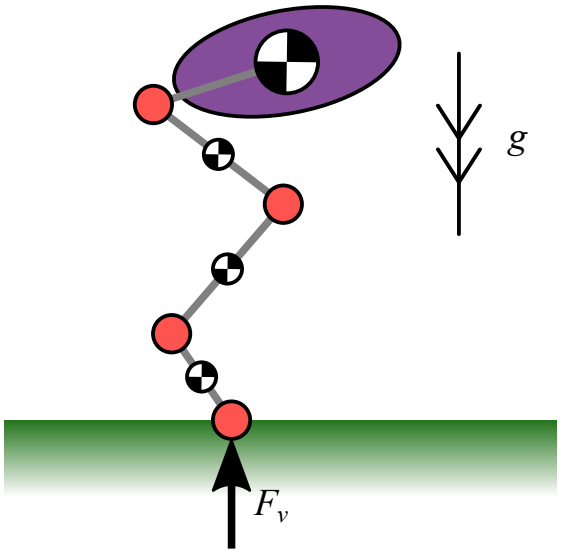


Figure 1.4: Diagram of a planar jumping model standing on flat ground with gravity acting down, perpendicular to the ground. The vertical ground reaction force, F_v, is labelled.

that the system is unable to grip or hold onto the ground, the breaking of contact is defined by the vertical ground reaction force, F_v. Without means of pulling on the ground, if $F_v \leq 0$ then the system must break contact with the ground and take-off occurs. This may also be considered in terms of the accelerations, $\ddot{y}$, and masses, m, of each segment in the model:

$$F_v = \sum_{i=1}^{n} m_i(\ddot{y}_i + g) \tag{1.2}$$

where n is the number of segments in the model and g is the acceleration due to gravity (9.81 ms^{-2}). Breaking contact with the ground does not guarantee the motion to be a jump; it is possible for the leg to be contracted with sufficient force that the foot is pulled towards the body. Should this occur while the body of the system is stationary or moving towards the ground, then the system would proceed to fall instead of jump. To differentiate these motions from jumps, a further criteria is proposed: the centre of mass of the system must have some velocity in the opposite direction to gravity at the point of take-off, as shown in figure 1.5. This is included to exclude motions such as diving from the definition of a jump.

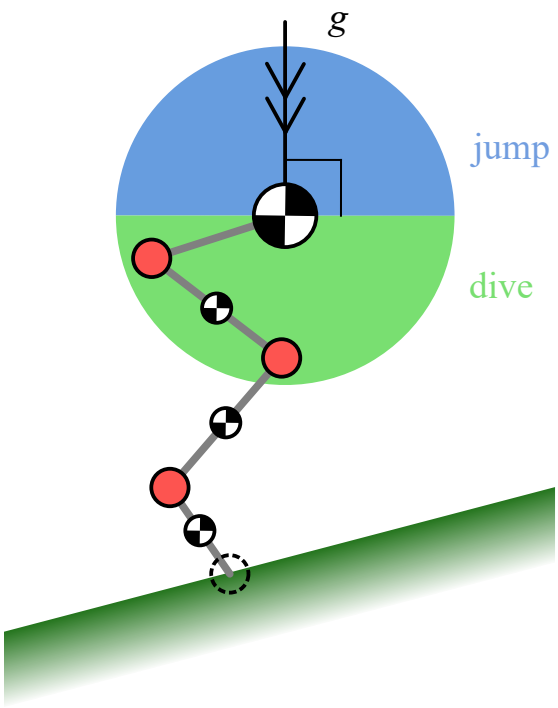


Figure 1.5: Diagram demonstrating the different motions which occur upon breaking contact with the ground. The coloured circle around the system's centre of mass represents the direction of the velocity of the centre of mass. Any motion involving velocity with some component opposite to the gravitational field direction is considered a jump.

The ground reaction force is a notoriously difficult aspect to model for any ground-based locomotion problem. In jumping the ground reaction force could be modelled using a spring-damper or as the constraint force in a revolute joint. The spring-damper model requires tuning for the specific model and motion exhibited, but allows the simulation to proceed after contact with the ground is broken. Whereas the revolute joint model will apply a negative ground reaction force, allowing the system to pull on the ground. This means that the revolute joint model must measure the ground reaction force and halt the simulation when the vertical force becomes ≤ 0. In this work it is considered sufficient to model contact with the ground using a revolute joint as this requires no additional steps in deriving the model.

The horizontal constraint force of the revolute joint connecting the system to the ground may be used to evaluate the friction forces acting in the system. In order to consider realistic motions, the horizontal ground reaction force must not exceed the limits of friction. Some jumping studies do not consider friction [12], [17], [20]. This is likely because friction depends on the normal force, which in jumping is the vertical ground reaction force. As the system approaches the point of take-off, the vertical ground reaction force approaches zero and thus the friction force will also. Therefore it may suffice to only consider the vertical ground reaction force in such applications.

Take-off and flight are typically considered separately from the jumping motion. In this work, and many others [12], [14]–[17], [20], [34], [35], jump height is considered as the height gained by the centre of mass of the system during the flight state. This is generally modelled by assuming ballistic motion of the system with no effects of air resistance, thus the velocity

of the system at take-off defines jump height as:

$$h = \frac{v_{\text{TO}}^2}{2g}. \tag{1.3}$$

Where h is the jump height, v_{TO} is the velocity at take-off and g is the acceleration due to gravity ($9.81ms^{-2}$).

1.6.3 Stability and Balance

The term "stability" used in reference to dynamic systems in the context of optimal control generally refers to Lyapunov's definition of stability which is that an equilibrium state of a system is stable if the system, when started within a certain distance from the equilibrium, will not move greater than a finite distance from the equilibrium at any time in the future [36]. "Stability" is often used to describe a structure's resistance to falling over, which follows the definition given by Lyapunov. However, these definitions are not applicable to systems undergoing acceleration, as the definition of an equilibrium point is that the rate of change of the state of the system is equal to zero. Some works discuss stability of jumping systems as the ability of the system to jump without falling over [34], however, in a context where optimal control theory is used often, it is important to ensure that the terms used are unambiguous. Thus, in this work the system's resistance to falling over is referred to as the "balance" of the system. A system may be in static balance if the system does not fall over while remaining stationary and dynamic balance describes a system which is moving while not falling over [37].

1.6.4 Redundancy in Models

This section looks to clarify the definitions of redundancy in jumping systems. A common notion of redundancy relates to the muscles of a system; in models there are generally more muscles actuating a jumping system than there are degrees of freedom and thus the muscles are redundant. This redundancy means that there are an infinitely many combinations of muscle forces which may produce a given motion of the model. This work considers a separate form of redundancy in jumping systems: the kinematics of the body and leg's motions. For a planar model, the body has 3 degrees of freedom. Any model which includes a leg having more than 3 joints will therefore be redundant in moving the body. This redundancy of the leg allows for "self motion" in which the leg may control the body while also meeting other objectives at the same time.

The forward kinematics of a model define the relationship from joint properties to body prop-

erties:

$$\begin{aligned}\boldsymbol{x} &= f(\boldsymbol{q}) \\ \dot{\boldsymbol{x}} &= f(\boldsymbol{q}, \dot{\boldsymbol{q}}) \\ \ddot{\boldsymbol{x}} &= f(\boldsymbol{q}, \dot{\boldsymbol{q}}, \ddot{\boldsymbol{q}}).\end{aligned}$$

Inverse kinematics describe the reverse:

$$\begin{aligned}\boldsymbol{q} &= f(\boldsymbol{x}) \\ \dot{\boldsymbol{q}} &= f(\boldsymbol{x}, \dot{\boldsymbol{x}}) \\ \ddot{\boldsymbol{q}} &= f(\boldsymbol{x}, \dot{\boldsymbol{x}}, \ddot{\boldsymbol{x}}).\end{aligned}$$

This kinematic redundancy may be taken advantage of during jumping motions, this is discussed further in Chapter 4.

1.7 Kinematics of Jumping Systems

The kinematics of a jumping system are used in Chapter 4. This section provides clarification of the necessary terms.

Kinematics describe the motion of a system without consideration of the forces and torques which cause such motion. In the planar jumping models of this work kinematics are used to relate the positions, velocities and accelerations of the degrees of freedom, typically relating the motion of joints to the motion of the body of the system. Kinematics are straightforward to derive and provide a useful tool for synthesis of jumping motions, whereas dynamic models require an additional step in determining the forces and torques required to produce desired accelerations. In this work, static optimisation is applied to kinematic models of jumping systems to produce jumping motions. This process uses the mapping from joint angular accelerations to the acceleration of the body which is defined by the Jacobian matrix of the system and the time rate of change of the Jacobian matrix. Both entities are derived in this section along with general kinematic definitions used in this work.

The Jacobian matrix, $\boldsymbol{J}$ (hereon referred to as the "Jacobian"), relates the joint angular velocities to the body velocity:

$$\dot{\boldsymbol{x}} = \boldsymbol{J}\dot{\boldsymbol{q}} \tag{1.4}$$

where $\boldsymbol{J}$, a $m \times n$ matrix, is defined as:

$$\boldsymbol{J} = \begin{bmatrix} -l_1 \sin(q_1) & -l_2 \sin(q_2) & \dots & -l_n \sin(q_n) \\ l_1 \cos(q_1) & l_2 \cos(q_2) & \dots & l_n \cos(q_n) \\ 0 & 0 & \dots & 1 \end{bmatrix}. \tag{1.5}$$

For the planar models in this work, the Jacobian has m rows, corresponding to the m degrees of freedom of the body: horizontal and vertical translation and rotation about the normal of the plane. When considering only the translation of the body, the Jacobian will have 2 rows. The n columns of the Jacobian correspond to the n degrees of freedom, or joints, in the leg.

The kinematic accelerations, or second order kinematics, are found by taking the derivative of the velocities with respect to time:

$$\ddot{\boldsymbol{x}} = \boldsymbol{J}\ddot{\boldsymbol{q}} + \dot{\boldsymbol{J}}\dot{\boldsymbol{q}}. \tag{1.6}$$

The time rate of change of the Jacobian, $\dot{\boldsymbol{J}}$, is defined as:

$$\dot{\boldsymbol{J}} = \begin{bmatrix} -l_1\dot{q}_1\cos(q_1) & -l_2\dot{q}_2\cos(q_2) & \dots & -l_n\dot{q}_n\cos(q_n) \\ -l_1\dot{q}_1\sin(q_1) & -l_2\dot{q}_2\sin(q_2) & \dots & -l_n\dot{q}_n\sin(q_n) \\ 0 & 0 & \dots & 0 \end{bmatrix}. \tag{1.7}$$

The product $\dot{\boldsymbol{J}}\dot{\boldsymbol{q}}$ represents the centripetal acceleration of the body due to rotational velocities of the leg segments. Note that equation (1.6) is a kinematic relation between joint and body accelerations. The equations contain no information about the torques causing the accelerations.

1.8 Dynamic Equations of Motion for Jumping Models

The generalised dynamics of an articulated system are presented in different forms depending on the school of research. This section provides clarification of the general form of equations of motion used in this work. All derivations of equations of motion use Lagrangian mechanics, examples of which may be found in various engineering mechanics text books or lecture series. Lagrangian mechanics is chosen over Newtonian mechanics as its algorithmic nature makes it more easily applied to systems with multiple degrees of freedom as well as being especially suitable for use in computer algebra system implementations. The equations of motion for the models used in this work are provided in B.

The equations of motion for a dynamic system are presented in their general matrix form by collecting the joint angle, velocity, and acceleration terms. It is common practice to present the equations of motion in their inverse dynamic form, where the net forces at each degree of freedom are found from the positions, velocities and accelerations. The following equations show some of the conventions used to represent the dynamics in their general form.

$$\boldsymbol{H}\ddot{\boldsymbol{q}} \quad + \boldsymbol{C}\dot{\boldsymbol{q}} \quad + \boldsymbol{\tau}_g \quad = \boldsymbol{\tau} \tag{1.8a}$$

$$\boldsymbol{M}(q)\ddot{\boldsymbol{q}} + \boldsymbol{C}(q,\dot{q})\dot{\boldsymbol{q}} + \boldsymbol{\tau}_g(q) = \boldsymbol{\tau} \tag{1.8b}$$

$$\boldsymbol{A}(q)\ddot{\boldsymbol{q}} \; + \boldsymbol{B}(q)\dot{\boldsymbol{q}}^2 \; + \boldsymbol{C}(q) = \boldsymbol{\tau}(\boldsymbol{q},\dot{\boldsymbol{q}}) \tag{1.8c}$$

The inertia matrix is represented by either $\boldsymbol{A}$, $\boldsymbol{H}$ or $\boldsymbol{M}$. The centripetal and Coriolis terms

are represented by $\boldsymbol{C}\dot{\boldsymbol{q}}$, $\boldsymbol{C}(q,\dot{q})\dot{\boldsymbol{q}}$ or $\boldsymbol{B}(q)\dot{\boldsymbol{q}}^2$, where $\dot{\boldsymbol{q}}^2$ represents the element-wise product of $\dot{\boldsymbol{q}}$ with itself. $\boldsymbol{\tau}_g$, $\boldsymbol{\tau}_g(q)$ and $\boldsymbol{C}(q)$ are the vectors of torques acting at each joint due to gravity. $\boldsymbol{\tau}$ is the vector of externally applied torques at each joint. $\boldsymbol{\tau}$ may be considered as the torques provided by motors at each joint, as it often is in manipulator work. $\boldsymbol{\tau}(\boldsymbol{q},\dot{\boldsymbol{q}})$ is used as shorthand to represent the various torques at joints due to muscle or tendon models included in some biomechanics work. $\boldsymbol{\tau}$ and $\boldsymbol{\tau}(\boldsymbol{q},\dot{\boldsymbol{q}})$ are interchangeable.

Equations (1.8a) and (1.8b) are generally found in robotics literature [21] with (1.8b) only being used when the joint angle dependencies are relevant to show in the equations. Equation (1.8c) is found in older biomechanics work (1990s) [12], [15], it is unclear what notation is used in recent studies as dynamic models are typically constructed in software and their equations are not explicitly presented in published papers.

This work uses equation (1.8a) for its conciseness. A further condensed form of the dynamics will be used when equations become long:

$$\boldsymbol{H}\ddot{\boldsymbol{q}} + \boldsymbol{b} = \boldsymbol{\tau} \tag{1.9}$$

where

$$\boldsymbol{b} = \boldsymbol{C}\dot{\boldsymbol{q}} + \boldsymbol{\tau}_g. \tag{1.10}$$

All forms of equation (1.8) provide the inverse dynamics of the model. The forward dynamics are found by inverting the inertia matrix of the model:

$$\ddot{\boldsymbol{q}} = \boldsymbol{H}^{-1}\Big(\boldsymbol{\tau} - \boldsymbol{C}\dot{\boldsymbol{q}} - \boldsymbol{\tau}_g\Big). \tag{1.11}$$

In a well defined model, the inertia matrix is symmetric and positive definite and therefore always invertible. If the model is not properly defined, then the inertia matrix will not be invertible. Examples of ill-defined models are provided in the following sections where relevant.

1.9 Computer Algebra Systems

The process of deriving the equations of motion for a model can be automated and completed by a computer algebra system, such as MatLab's symbolic toolbox [38]. (Similar systems such as Wolfram Alpha and SymPy are also available). This work makes use of the symbolic toolbox in MatLab for derivations of equations of motion, kinematic equations and constraint equations for various models. A computer algebra system can be used to compute the time derivatives of the segment positions to obtain the full set of kinematic equations for a model. Using the kinematics, the equations for kinetic and potential energies are trivial to obtain. Finally, the partial derivatives which comprise the Lagrangian may also be computed using a computer algebra system.

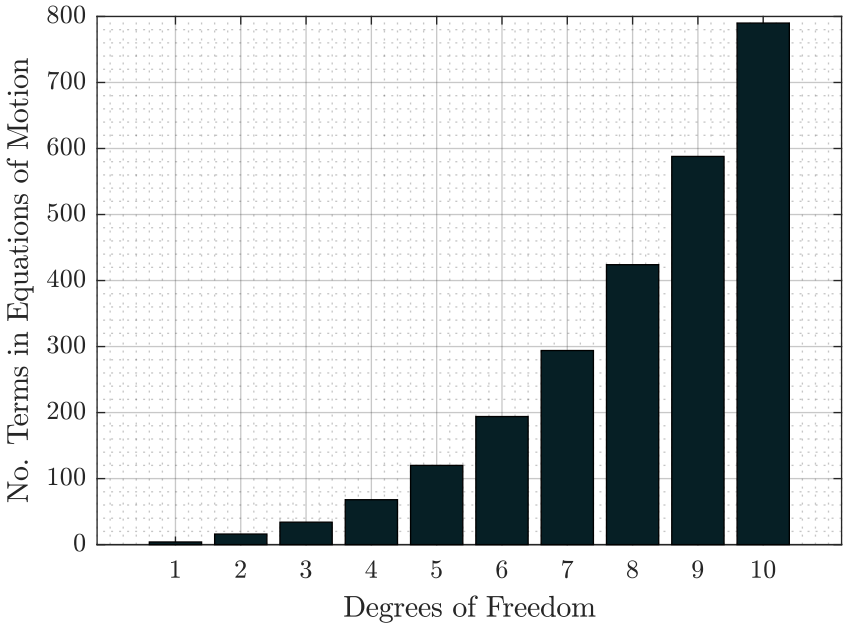


Figure 1.6: The number of terms in the equations of motion for planar articulated leg models. The models consider each segment as a rigid body with mass and rotational inertia. Joint angles are defined relative to the inertial frame to minimise the number of terms in each set of equations.

1.10 A Note on Model Complexity

The "complexity" of a model may refer to many aspects of the model. In this work, complexity is considered to be dependent on the number of internal interactions of the model. Internal interactions influence one another, leading to non-linear behaviours. Adding degrees of freedom to jumping models increases the number of interactions and therefore the complexity of the models. Such models quickly reach levels of complexity which are time consuming for researchers to comprehend. Thus, complex models generally require a change in the scope of research objectives from general system behaviours to specific trajectories and motions; features which are agnostic of parameters in the system to features which are parameter dependant.

To demonstrate this dilemma, the symbolic toolbox in MatLab [38] was used to derive the equations of motion for a range of planar leg models. All angles in the models are defined relative to the inertial frame to ensure no fictitious forces are included in the equations. The number of terms in the equations were counted in MatLab and the results presented in figure 1.6. Each term may be approximately related to an interaction within the model, such as centripetal acceleration, thus, more terms in the equations of a model means more components to comprehend for the researcher. The derived equations are presented in B.

1.11 Models used in this work

This section will discuss the 1, 2, and 4 degree of freedom planar jumping models used in this work. The following chapters of this thesis will refer to this section for context of the models used. The dynamic equations of motion for each model were derived using Lagrangian mechanics implemented in MatLab's symbolic toolbox. The equations of each model are included in B.

A common model found in biomechanical jumping literature comprises 4 segments: foot, shank, thigh and HAT (Head Arms and Trunk/Torso) [6], [12], [14], [16]. Each segment in the model is considered as a rigid body with uniform density and rotational inertia. The 4 degree of freedom model is generally considered sufficient to capture the main motions of a jumping system, however, it does not consider contributions from the torso, arms/wings, or the head of the system. Models with fewer degrees of freedom are considered here as they provide insights into the general motion of the system's centre of mass with the 2 degree of freedom model incorporating the non-linearity of an articulated leg.

1.11.1 One Degree of Freedom Model

The 1 degree of freedom models in this work, as shown in figure 1.7, consider a single rotational degree of freedom between the system and the ground. The model is equivalent to an inverted pendulum. The system may be abstracted as either: an inertial rod as in [17] and [20], a point mass body [34], or a rigid body with rotational inertia which is not based on a rod shape. Examples of the inertial body case for single degree of freedom models are rare as it is not necessary. However, 4 degree of freedom models which use the inertial body case include [12], [15], [16]. In the case of the point mass and rigid bodies, the body is attached to the ground via a massless, rigid link. In each case the system is connected to the ground by a revolute joint located where the foot of the system would be. The system is parameterised by its mass m, inertia I, and the length of the link l. The configuration of the model is defined by the angle, q, between the rod/link and the ground, and the angular velocity $\dot{q}$.

The inertial rod model considers the system's inertia about the joint connecting the leg to the ground. This is unlikely to be a readily obtained value, one is more likely to find the inertias

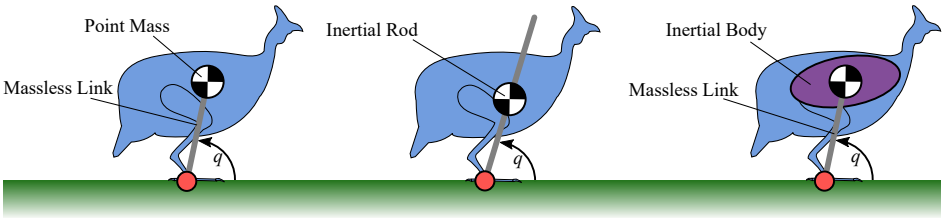


Figure 1.7: The single model variations. (Left) The point mass on a massless link, only considers the rotational inertia of the system about the joint between the link and the ground. (Centre) The inertial rod considers both the rotational inertia of the system about the joint and the centre of mass of the system. (Right) The inertial body is an extension of the inertial rod which does not assume the body to have the inertia of a thin rod.

of each body segment about their respective centres of mass [39] (note: the source cited here contains graphic images and may be unsettling for some readers). The rigid body model has rotational inertia about the point where the link connects to the body at the centre of mass, this is considered as the body's inertia, I_b. The inertia of the system about the joint connecting the link to the ground, I_j, is then obtained using the parallel axis theorem:

$$I_j = I_b + ml^2$$

where m is the mass of the system and l is the length of the link. The benefit of considering the system as an inertial body connected to a massless link is that the inertia of the system remains the inertia of the body, whereas treating the system as an inertial rod imposes an abstraction on the system's inertia. While this abstraction may be trivial, the resulting model and equations are not immediately related to properties of the system.

The system is actuated by a torque applied to the rod. With no foot, the source of the torque may be considered as two thrusters attached to the free end of the rod which point perpendicularly to the rod. With a foot, the torque might be applied by a motor at the joint connecting the rod to the foot.

1.11.2 Two Degree of Freedom Model

Adding a second rotational degree of freedom to a jumping model doubles the number of parameters required to fully describe the system. Additionally, the second degree of freedom introduces internal interactions to the system wherein the motion of the two segments are coupled. The 2 degree of freedom model comprises two rigid segments with lengths l_1 and l_2, masses m_1 and m_2, and inertias I_1 and I_2. The segments are connected together by a revolute joint with angle q_2 and externally applied torque τ_2. The lower segment is attached to the ground by a second revolute joint with angle q_1 and externally applied torque τ_1. This is dynamically equivalent to an inverted double pendulum and is a classic example of a system capable of chaotic motion.

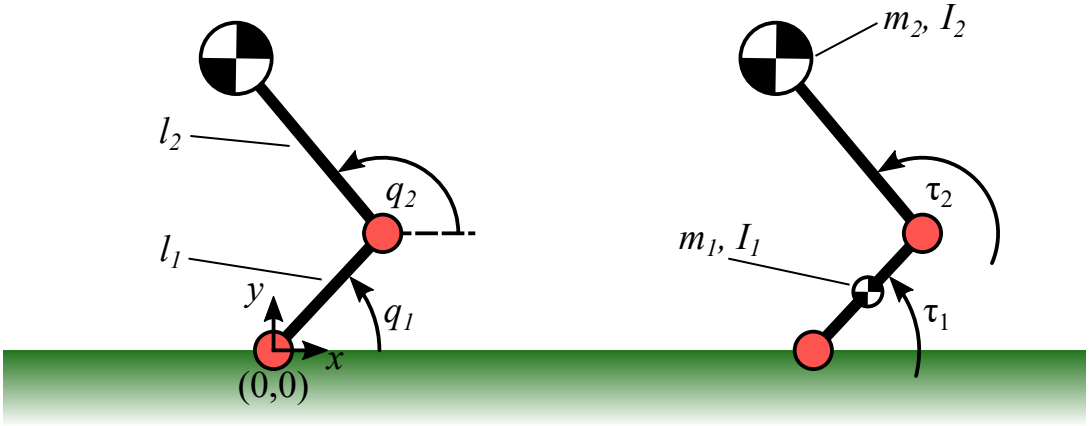


Figure 1.8: The two degree of freedom system, treating the second link as the body of the system. The first segment is treated as an inertial rod while the second segment is considered as a massless link connecting a rigid body to the first segment.

1.11.3 Four Degree of Freedom Model

The model used most often in jumping applications is the 4 degree of freedom model. In this work, the first three segments are treated as inertial rods while the final segment of the model, which represents the head arms and torso of the system, is considered as a rigid body connected to the third segment by a massless link. This gives more design control over the body's inertia. 4 degree of freedom models of humans consider the foot, shank, thigh as the first three segments. 4 degree of freedom models of birds consider the tarsometatarsus, tibiotarsus and the thigh as the first three segments. Most 4 degree of freedom models consider the head, arms, and torso as the fourth segment.

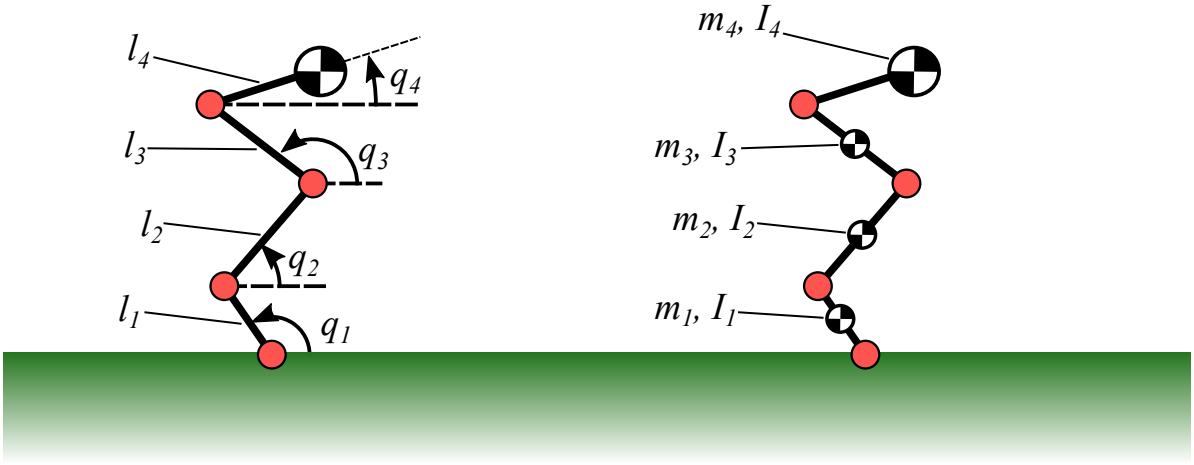


Figure 1.9: The four degree of freedom system, considering either the foot, shank and thigh or the tarsometatarsus, tibiotarsus and thigh as the first three segments. The fourth segment generally comprises the head arms and torso.

1.12 Summary

This chapter introduced the context of this work using predictive modelling methods in the study of dynamic jumping systems. The aims and objectives of the work were stated and discussed. The chapter also provided clarifications and definitions relevant to the thesis including the presentation of diagrams for the models used.

Chapter 2

Literature Review

This work considers the application of predictive modelling methods to jumping systems for motion data procurement. This literature review first presents the planar, articulated models used in jumping research for context. Next, the review explores the application of predictive modelling methods to such models in the jumping literature, as well as predictive modelling methods which are not found in the jumping literature. Each method is discussed in relation to the metrics described in Chapter 1:

- Predictive Power - How flexible the method is, the range and utility of the results it provides, and the limits in model complexity for the method.
- Computational Cost - The time it takes for a commercial, high spec desktop computer to carry out the method.
- Intellectual Investment - The domain knowledge required from the user of the method, in both the problem and the method.

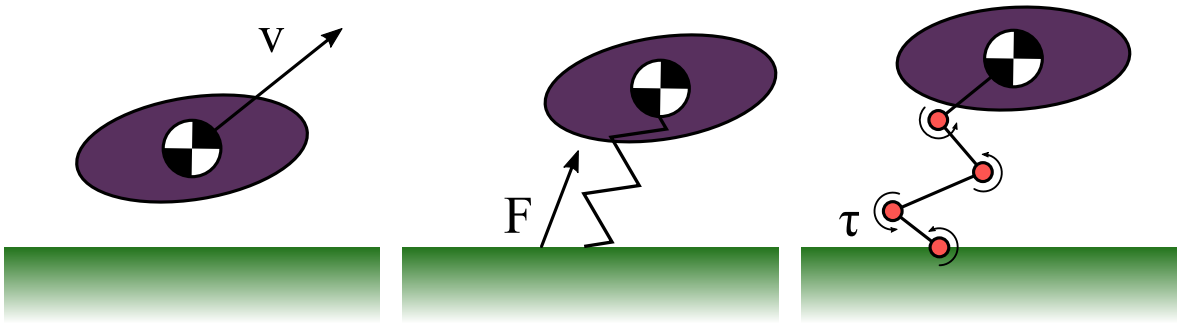
Predictive modelling methods found in robotics and optimal control literature that are not commonly found in jumping literature are presented and discussed regarding their potential application to jumping problems.

2.1 Jumping Models

This section categorises jumping models found in the literature by the number of degrees of freedom they have. Degrees of freedom may be considered as an indication of model complexity; the more degrees of freedom a model has, the more complex it may be considered to be. A method's predictive power depends in part on the range of model complexities that method is applicable to. The methods applied to the models are discussed briefly in this section with more detailed discussion in the following sections.

The models used to describe jumping systems in the literature often share the same equations. For example, a 4 degree of freedom planar model is used by Henry et al. to model the jumping motions of *Numida meleagris* [6] and by Pandy to model the jumping motions of humans [12]. The two models differ considerably in the size and mass of the links used. The size of a system correlates to the speed at which its muscles move, with smaller animals tending to

have faster moving muscles [40]. Due to this complex relationship between speed and size, it can be difficult to use the results from a particular set of model parameters with another set of parameters of a different scale.



(a) Single Degree of Freedom Model (b) Low Degree of Freedom Model (c) High Degree of Freedom Model

Figure 2.1: Jumping models found in the literature (a) A single degree of freedom model represents the entire system as either a point mass or a rigid body. Motion is typically modelled using traditional ballistic motion equations. (b) Low complexity models include ≤ 2 degrees of freedom and often include actuator models, such as muscles or tendons. The state space of a low complexity model is generally possible to present in graphs. (c) High complexity models include > 2 degrees of freedom and are generally beyond the limits of graphing representation.

2.1.1 Single Degree of Freedom Models

Single degree of freedom point mass and rigid body models are typically used in energetics studies of jumping systems and in modelling the flight stage of jumping systems. Many jumping studies consider jumping systems as rigid bodies during the flight stage of the jump, using ballistic equations of motion to determine the path of the system [12], [14]–[17], [20], [34], [35].

In their book, "Principles of Animal Locomotion" [33], Alexander considered the kinetic energy of point mass systems to determine the energetic requirements of motions, including jumping. The models used minimal descriptors and degrees of freedom, allowing Alexander to find the energetic cost of transport for a jumping system through analysis of model equations.

Parslew et al. [34] considered a jumping system as single rigid body with a massless leg connecting the body to the ground, and used analysis of the model equations to determine regions of dynamic balance for jumping systems in general.

Roberts et al. [17], [20], considered a single, torque actuated rigid rod as representing a jumping system. The rod was assumed to be connected to the ground at one end by a revolute joint. The torque propelled baton is a single degree of freedom system. However, the relationship between the linear ground reaction forces and the rotational motion of the system is non-linear and can become challenging to conceive. This issue is exasperated when further degrees of freedom are included in jumping models used in other studies [12], [14]–[16]. Roberts et al.

analyse the model equations and derive the boundaries for regions in state space for which the optimal control strategy is readily deducible.

2.1.2 Low Complexity Models

Low complexity models have up to 2 degrees of freedom and may include models of actuators, such as muscles or motors. A common model of muscles is the Hill type model [41]. The modelling of actuators is not directly considered in the scope of this work.

Alexander derived a two degree of freedom jumping model to investigate the ideal take-off conditions for athletes performing high and long jumps [42]. The system is treated as a single rigid body with revolute joints at the knee and at the point of contact between the leg and the ground. The model also included a muscle acting as an actuator at the knee joint.

Roberts and Marsh [43] investigated jumping in bullfrogs using a low complexity muscle model. Roberts and Marsh assessed the performance of a single muscle with various characterisations of a series elastic element and variable mechanical advantage of the lever arm representing the leg of the system. Their comparison demonstrated that jumping performance can be enhanced by inclusion of a series elastic element and by variation of the leg's mechanical advantage.

The spring loaded, inverted pendulum [2] is a widely adopted model in locomotive studies. It is however, much more prevalent in running and continuous hopping studies [19] than it is in jumping studies. The jumping robot SALTO is an example of an implementation which uses the spring loaded, inverted pendulum model for continuous hopping control in a jumping robot [30], [44]. SALTO's leg is a linkage system designed to apply a force through the centre of the mass of the robot [45], ensuring that minimal moments are produced on the body during the push-off stage. This method of jumping requires additional control of the orientation of the system in order to direct a single jump, which has recently been implemented using a reaction wheel on the robot [46].

Aguilar et al. demonstrate a similar model of a linearly actuated jumping robot with a spring attached to the foot [47]. Aguilar et al. use a comprehensive method of numerical analysis in their work to demonstrate two distinct jumping strategies: a stutter jump, involving a small jump before performing the maximum height jump, and a single jump involving a counter-movement and maximum height jump in a single motion. The stutter jump result lies in a grey area between continuous hopping and single jumping and is not a method considered in this work.

2.1.3 High Complexity Models

High complexity models consist of > 2 degrees of freedom and may include models of actuators. A particularly common planar model of jumping systems comprises 4 degrees of freedom and is used in the simulation of human jumping motions [12]–[14], [16], primates

[35], and birds [6], [34]. It is a common assumption that the legs of a jumping system are the primary driver of jumping motions, leading to the popular decision that the Head, Arms and Torso (HAT) may be lumped together and treated as a single rigid body in the model [12]–[16], [48].

High complexity models are also frequently used in inverse dynamics applications [3], [4], [6]–[8], in which experimentally measured motion data are used to determine muscle forces acting throughout the motion [11]. Other studies use experimentally measured data to provide target motions for dynamic optimisations [5], [9], [10]. This is particularly useful when the data are incomplete or when data were obtained with reference to a different model of the system, making inverse dynamics difficult.

Kargo and Rome [49] investigated the effects of degrees of freedom in jumping frog models by simulating jumping motions with models of increasing degrees of freedom. Their predictive modelling method used 1000 forward dynamic simulations with randomly generated torque vectors held constant for each jump.

2.2 Analysis of Model Equations

Analysis of model equations is typically applied to single degree of freedom models, such as [13], [17], [20], [33], [34]. The number of parameters required to completely describe the model's configuration, referred to as the "state space", increases as the degrees of freedom and number of actuators increase. The increase in the number of parameters leads to an increase in the "volume" of the state space. This dilemma was termed "the curse of dimensionality" by Bellman [50] and is a hindrance to many methods of study.

Results of analysis of model equations in the literature are generally applicable to a range of systems. For example, Alexander demonstrates the energetic cost of transport associated with a massive foot through analysis of a two body jumping system [33]. This result applies to any jumping system with feet and gives a relationship between the cost of transport and the fraction of system mass belonging to the foot. Such analysis requires the derivation of a suitable model which requires intellectual investment both in the derivation process and in the background knowledge of algebra, calculus, and mechanics required to perform the derivations. The result of the analysis has implications for general jumping systems as it provides an equation into which specific parameters may be input.

Alexander moves from analysis to numerical methods with the introduction of a second degree of freedom in their high and long jump work [42]. The increased complexity of multiple degree of freedom models makes their analysis difficult and time consuming to perform, even for computer algebra systems. The results of such analysis are difficult to communicate due to their high dimensionality and multiple internal interactions. A similar method was used by Roberts and Marsh [43] in their investigation using a low complexity muscle model. Roberts and Marsh used a brute force optimisation to assessed the performance of their model. A range of configurations were tested and the best were selected for discussion. This approach

arguably requires less intellectual investment to solve the problem than an analytical approach would, however, the brute force method requires more computation than analysis of model equations. It also remains uncertain whether the results apply to jumping systems in general or are only relevant to the parameters used in the study.

Parslew et al. [34] derived the constraints of body accelerations required to maintain dynamic balance during jumping by analysing the equations of motion for a single degree of freedom model. The derived constraints are applicable to any jumping system which intends to keep its foot flat on the ground; the result is not specific to a given parameterisation of the model. The notion that the foot cannot grip the ground is a common consideration and is often used to determine the point at which a jumping system takes off [17], [20], [34]. The analysis performed by Parslew et al. provides a reusable set of equations which may be applied to different parameterisations of jumping systems.

The analysis performed by Roberts et al. [17], [20] seems to require some prescience of the system's dynamic behaviour as some boundaries are derived using statements and notions of system properties and behaviours. For example, one boundary line in the work describes the point at which the system is no longer able to pull itself backwards towards the ground and is forced to fall forwards. This method of derivation is likely not applicable to multiple degree of freedom models with non-linear dynamics as the number of internal interactions increases and the behaviours of the system become difficult to anticipate.

The number of parameters required to completely describe a model's configuration increases as the degrees of freedom and number of actuators increase, meaning that the "volume" of the configuration space grows exponentially as the model's fidelity is increased. This dilemma was termed "the curse of dimensionality" by Bellman [50]. A common response to this dilemma is demonstrated in the different methods used by Alexander with their walking [13] and jumping models [42]. The two models are similar in that they both treat the system as a single rigid body, however, the leg of the walking model is represented by a single massless link while the jumping model includes a second degree of freedom at the knee as well as a muscle actuator model. With the increase in model complexity, Alexander turns to the use of numerical methods to study the jumping model as opposed to the algebraic analysis they used for the walking model.

The numerical analysis performed by Aguilar et al. [47] involved a sweep of the parameters describing the behaviour of the actuator. The system was started from a fixed state and maximum jump heights for each set of parameters were recorded. This use of numerical analysis to investigate a range of parameters and determine their optimal values is common with low complexity and high complexity models [16], [42], [43], [47].

The work done by Kargo and Rome [49] used a random search in the external torque space, selecting a random torque vector which was then applied to the system for the duration of the jump. While the probability of randomly selecting an exact value when choosing random continuous numbers is zero, choosing a vector of 5 values where each value is accurate to within 10% of a given range has a probability of $0.2^5 = 0.00032$. Thus, 1000 samples

may be insufficient in choosing an accurate torque vector. This probability would be much smaller for the models with additional degrees of freedom. A different predictive modelling method may have been better suited to the study by Kargo and Rome. The use of constant torques, as opposed to varied torques, applied throughout the jump may have also hindered the performance of each of the models.

2.2.1 Discussion of Assessment Criteria

The quality of results produced through analysis of model equations is high. Such studies often demonstrate the process applied to a case study and provide results for a particular system of interest, however, the studies also provide equations which other researchers may use in studies with different systems. Thus, the range and utility of results for analysis of model equations are considered to be excellent. It is identified that analysis is typically only used with single degree of freedom systems.

Computational cost is generally not presented for studies which utilise analysis of model equations. This is likely because the derivations and equation manipulations are often carried out by the author without the use of a computer algebra system. Because of this, it is assumed that analysis of model equations may be conducted with minimal computational cost.

It is difficult to assess the intellectual investment put into the analysis found in the literature as there are generally multiple authors involved in each study, each of whom will have their own domain knowledge and skill set. Instead, this work uses case studies to assess the general domain knowledge required to complete studies using each of the methods of interest.

2.3 Static Optimisation

This work investigates the use of static optimisation for the resolution of kinematic redundancy, specifically using the acceleration of the system. Kinematic optimisation of this style was originally proposed for control of robotic systems [51] in 1980, and later developed for use with redundant manipulators [52] (1983). The method uses the second order kinematic equations of a model to simulate its motion. Kinematic methods are generally much lower in computational requirement than dynamic methods [53].

The use of kinematics in robotic control is not a new concept; it was explored throughout the 1980s for manipulator control [52], [54]. An articulated leg and a manipulator are very similar systems; both consist of segments connected by rotary joints and are generally used to transport an object attached to the end of the segment chain. In manipulation this object is termed the “end effector” while in jumping this is typically the “body” of the system. The major differences between the two applications are that manipulators are usually fixed to the ground and perform repetitive tasks in the same work space, while jumping systems seek to break contact with the ground and transport themselves to a different space. Despite these differences, the motions of interest of jumping systems tend to occur while the system is in

contact with the ground and so methods which are useful for manipulator applications may also be useful in the context of jumping.

Kinematic optimisation for redundancy resolution in manipulator control seems to be rarely used. This is likely due to the fact that the method does not exhibit "cyclicity", meaning that the method does not provide a direct relationship between the system's joint configuration and the body position [21], [52]. Motions which follow a closed path for the end effector are not guaranteed to follow a closed path in joint space. This is not necessarily an issue in locomotion, particularly jumping problems, where the motion of the system is typically not cyclic.

A numerical implementation of kinematic control was performed by Sellers [35] to transfer data from one jumping model to another. Sellers interpolated joint angles from a starting state to a desired end state and adjusted the joint angle trajectories to ensure the body followed a straight path. This required sampling to determine the appropriate joint motions and may prove difficult to scale to systems with more degrees of freedom.

Richards and Porro [55] used kinematic models and quaternion interpolation to simulate jumping motions in frogs using high complexity models. Richards and Porro claim their method to be novel because to their knowledge "it provides one of the simplest sets of mathematical rules that predict realistic limb motion in the absence of detailed physical, anatomical and physiological constraints". Static optimisation for second order kinematic redundancy resolution may provide a similar method in terms of simple mathematical rules for predicting limb motions in the absence of the aforementioned constraints.

2.3.1 Discussion of Assessment Criteria

Static optimisation as a tool for predictive modelling is not common in the literature for jumping dynamics. As such, this work intends to assess the predictive power of the method in this application.

The computational cost of static optimisation is considered to be low as the optimisation requires a matrix inversion as opposed to iterative gradient descent.

The intellectual investment of static optimisation for the resolution of kinematic redundancy will be investigated in this work. Existing works in the literature, which derive and use the method, make heavy use of complex mathematical notation. This may make the method difficult to adopt for researchers not already experienced in the field of robotics or optimal control.

2.4 Dynamic Optimisation

Dynamic optimisation is used in a variety of ways, including optimisation of joint torque trajectories [16] or muscle excitations [15]. Some studies also use experimentally measured

data to provide target motions for dynamic optimisations [5], [9], [10], which is particularly useful when existing data is incomplete or when data was obtained with reference to a different model of the system, making inverse dynamics difficult.

The fact that not all biomechanical studies utilise simulated models and dynamic optimisation in particular may be due to a few factors, including:

- Empirical results demonstrated that dynamic and static optimisation yield the same results when determining muscle forces acting in a system [56]
- The impossibility of perfectly reproducing a real world system using a simulated model
- An intimate knowledge of both dynamics and optimal control is required to implement dynamic optimisation, tools which facilitate the process are scarce [57]
- Other, "model-free" optimisation methods require only state information about a dynamic system in order to work, they do not require thorough knowledge of the equations of motion to implement, and dynamic simulation software are common [31], [32]

Anderson and Pandy's work demonstrating that static and dynamic optimisation may ultimately provide the same information for a given model can be seen as an argument in favour of using dynamic optimisation, however, for researchers already using static optimisation it is clarification that the intellectual investment required to change their optimisation method to dynamic optimisation would bring no improvement to results.

Selbie and Caldwell [16] used dynamic optimisation to determine the onset times of the torques acting at each joint in a 4 degree of freedom human leg model, with the objective to maximise jump height. Their study looked at the torque trajectories for a variety of starting configurations of the system. The maximum obtained jump heights were relatively close for all starting positions, however, the onset times varied considerably. The variation in torque onset times implies variation of actuator co-ordinations, as proposed by Selbie and Caldwell, but it also implies variation in the amount of work done by the actuators of the system. This is reflected in the variation of horizontal displacement in the optimal jumps with some attempts travelling further than others. Angular displacement of the system is not reported, so actual variations in the system's kinetic energy at take-off cannot be properly determined. Selbie and Caldwell's work is in contrast with most other works involving 4 degree of freedom models for maximum jump height studies as they directly consider the external torques applied to the model, whereas most works consider a muscle-based actuator [9], [12], [15], [28].

The current state of the art in dynamic optimisation for jumping was demonstrated by Bishop et al. [58]. Bishop et al. demonstrated their optimisation using a musculoskeletal model of the elegant-crested tinamou, *Eudromia elegans*, comprising 9 segments and 26 degrees of freedom [59]. Bishop et al. assumed that many of the degrees of freedom were not necessary in producing the jumping motions under investigation, and so constrained the model including the assumption that the two legs moved with bilateral symmetry. After constraints and assumptions, the model was reduced to 9 degrees of freedom. Direct collocation was used as

the dynamic optimisation method, seeking the optimal time derivatives of muscle activations and tendon forces, with a nominal solution obtained in a reported 29 minutes. The work goes on to present the results of 18 more optimisations for a sensitivity analysis, leading to an estimated 551 minutes (9 hr 11 min) of computing time for the study (not counting optimisations which were may have been run during development and testing). This is significantly less computation time than was reported by Anderson and Pandy, 22 years prior, in their optimisation study which used similar fidelity models [15]. Anderson and Pandy reported 1800 hours (2.5 months) of CPU time to obtain a single set of results. It is noted here that the processor used by Bishop et al. was a 2.4 GHz CPU while Anderson and Pandy's CPU was 0.18 GHz. Adjusting for the different computation rates, Anderson and Pandy's optimisation may be approximated as taking 15.5 hours on a 2.4 GHz CPU. This comparison is not strictly fair and is only a rough figure, but it demonstrates that modern computing power enables the use of complex optimisation methods which were previously limited to researchers with access to supercomputers.

2.4.1 Discussion of Assessment Criteria

The implications of the results produced by dynamic optimisation are limited to the system parameterisation used in the optimisations. This limitation is a trade off for the increase in model complexity which the method is applicable to; dynamic optimisation is most often found in the literature applied to high complexity models.

The computational cost of iterative optimisation methods such as dynamic optimisation is generally high. Two examples in the literature [15], [58] reported using between 9 and 16 hours of CPU time with a 2.4 GHz processor. On the other hand, the work by Bishop et al. [58] mentions that a single optimisation took 29 minutes which is a reasonable amount of computational time. However, for the early stages of a study, where rapid prototyping may be a requirement, 29 minutes may become a significant cost to using the method.

The intellectual investment for dynamic optimisation is generally large. Research papers which include dynamic optimisation typically introduce multiple mathematical notations and concepts. The methods work over an entire trajectory instead of at a single step, this requires consideration of more factors during their implementation and use than a static optimisation method would require.

2.5 Reinforcement Learning

New reinforcement learning algorithms are typically published by means of benchmark comparison with existing algorithms. One popular set of benchmark problems is OpenAI Gym [60], which includes a set of dynamic control problems implemented in the MuJoCo simulation environment [61], including rigid body models of an articulated leg and a humanoid. Many efforts have shown apparent success on these, and similar, problems [23]–[26], [62]–

[65].

At first glance, many reinforcement learning benchmarks demonstrate clear improvements to the performance of the new algorithm over previous iterations. Upon closer investigation, the use of 3 trials for such stochastic optimisation methods [25] raises questions about the accuracy of representation provided in the work. Reinforcement learning algorithms comprise many different parameters and sub-methods within which may contribute to a given performance on a given task. Such parameters are difficult to compared due to the stymieing computational expense of executing a single trial of a reinforcement learning algorithm.

Reinforcement learning methods are not commonly applied in research of jumping systems. However, reinforcement learning has been used to optimise controllers for quadrupedal robots using data collected from simulated models of the robots [66], [67]. The practice of optimising a controller, or policy, is much more involved than a dynamic optimisation; a controller should behave optimally in any state that the system can enter, whereas a dynamic optimisation is generally only concerned with a specific trajectory through the state space. The benefit of optimising a controller is that the system's behaviour is then defined for many scenarios and can be used to gain insights about many behaviours of the system. The difference in the scope of the methods becomes apparent in differing convergence times for dynamic optimisation and reinforcement learning methods, with the latter often taking months to converge [68].

Some jumping motions are found in benchmark tests, such as OpenAI Gym [60]. However, the motions and models of these benchmarks are not extensively validated and so are not considered in this work.

Similar to the biomechanical methods of collecting experimental data and fitting models to the data, Peng et al. [10] and Peng et al. [69] demonstrate the use of reinforcement learning methods to control dynamic models of various systems, including humans, robots and dragons. The controllers are optimised to follow experimentally measured motions including back flips, front flips and jump kicks. The models used range from 30-79 degrees of freedom. The motions provided in [10] are provided as motion capture data, describing the motions of segments. Whereas the motions provided in [69] are video frames. A table describing the number of samples required per motion is presented in [10]. Required samples range from 44-191 million. Supposing the forward dynamics take 0.1 ms [70] to compute, that is a minimum of 1 hour and 10 minutes of computation time to simulate the dynamics while generating the samples. The actual run time of the methods will be much greater than this as a feed forward of the neural network representing the control policy must be performed to determine the control signal for each sample generation to ensure the data collected is on-policy. Then, gradient descent must be performed on the policy every so many samples to iteratively improve the policy. This implies that the methods are likely to take considerable computational time to complete and may not be effective tools for researchers with access to limited computational power.

2.5.1 Discussion of Assessment Criteria

Reinforcement learning methods optimise a policy instead of a trajectory. This means that the results they provide have greater utility and re-usability than methods such as dynamic optimisation as a policy may be used to generate behaviours from a range of starting states. A policy also provides a mapping from state to control action which could provide insights into how the controller operates, i.e. which states are most influential to the selection of actions. While the policy does provide more information than a trajectory would, the policy is still dependent on the parameterisation of the system which was used during the optimisation; a different system would likely use a different policy. Reinforcement learning methods are generally applicable to high complexity models with many degrees of freedom and they are model free methods. The predictive power of reinforcement learning methods is good but limited to the parameters used.

The computational cost of reinforcement learning methods is significantly high. Reinforcement learning methods typically use random sampling to explore the problem. This process of interaction with the system means that reinforcement learning methods often require thousands and even millions of samples to achieve sufficient performance. Most reinforcement learning methods are on policy and so require samples to be collected by the current iteration of the policy being optimised. This prevents the re-use of collected data and means that every optimisation run must collect new samples from the simulation during optimisation.

The intellectual investment of reinforcement learning methods varies. Implementations of many state of the art reinforcement learning methods are available to use in environments such as Python and MatLab. As tools, the methods are not dissimilar from dynamic optimisation in their implementation for a user. However, the complexity of reinforcement learning becomes an issue if the tools require tuning or fail to provide adequate results. Hyper-parameters of reinforcement learning methods are complex and not always intuitive to tune. The algorithms often have multiple components working together which may need to be tuned as well. Reinforcement learning and optimal control share many features and are derived from the same background, however, they often use different notation and jargon and can seem very different at a surface level.

2.6 Thoughts and Conclusions

Analysis of model equations has the best predictive power of the results investigated in this literature review. The process of analysis produces equations which can be used by other researchers to test systems with different parameters to any case studies used in the initial analysis study. It was observed that analysis is generally applied to single degree of freedom models and is rarely seen applied to low or high complexity models. As such, this work intends to investigate and assess the application of analysis to low complexity models to determine the issues related to such practice.

This review found no work in the jumping literature which used static optimisation to resolve second order kinematic redundancy in jumping systems. Work in robotics was found which used the method to study similar models to those used in the jumping dynamics literature. In light of these findings, this work will explore the application of static optimisation to a jumping problem and assess the method in its predictive power, computational cost, and intellectual investment.

Dynamic optimisation was identified as a commonly used method in jumping research. The application of dynamic optimisation to jumping problems will not be a novel contribution of this thesis. However, the assessment of the method's application is included here so that the method may be properly compared with the other methods considered in this work as no literature was found which described the method's predictive power or intellectual investment.

In a similar case to static optimisation, reinforcement learning was not found to be applied in jumping research. Some case studies were found where jumping motions were included in benchmarking tests yet little focus was made into their validation. Reinforcement learning provides a set of model free methods which are applicable to high dimensional, continuous dynamics problems. Because of this, a state of the art reinforcement learning method, namely Deep Deterministic Policy Gradients (DDPG), is investigated in this work through a case study. The method is assessed on its predictive power, computational cost, and intellectual investment.

Chapter 3

Model Analysis Methods

Applications of model analysis to single degree of freedom models in the literature demonstrate the method's utility in obtaining general insights into a system and its behaviours, as opposed to insights which are specific to a given set of model parameters. For example, Alexander [33] uses analysis of model equations to show that an increase in foot mass increases the energetic cost of transport for a jumping system. While the algebraic equations which result from analysis methods may be applied to a broad range of parameterisations, their application is generally limited to low complexity models, often to single degree of freedom models. This chapter investigates the application of analysis of model equations to models which have multiple degrees of freedom, by using a computer algebra system to alleviate the algebra and calculus involved in the analysis.

Analysis of the equations of low and high complexity models was not found in jumping literature. As such, the work in this chapter work explores the application of analysis methods to low complexity models to determine what insights may be obtained. The analysis builds on existing work done by Roberts et al. [17], [20]. This work was selected due to the practically algorithmic nature of the method described. This allowed for the method to be automated and applied in a more general way. Another objective of the work in this chapter is to assess the predictive power, computational cost, and intellectual investment of analytical methods for jumping dynamics studies.

This chapter also provides a novel technical contribution with the concept of the dynamic balance ellipsoid, which may be used to determine the feasible accelerations of a jumping system during dynamic jumping motions. The dynamic balance ellipsoid is applicable to systems of arbitrary degrees of freedom and is limited by the intellectual investment induced by the numerous concepts required in its derivation.

3.1 Computational Cost and Computer Algebra Systems

Computer algebra systems provide the means to manipulate mathematical equations, including algebraic and calculus operations. In this work, the symbolic toolbox of MatLab [38] was used to carry out model derivations following a popular branch of analytical mechanics: Lagrangian mechanics. This use of a computer algebra system transfers some of the intellectual effort of the analysis method to computational cost. While it is difficult to measure

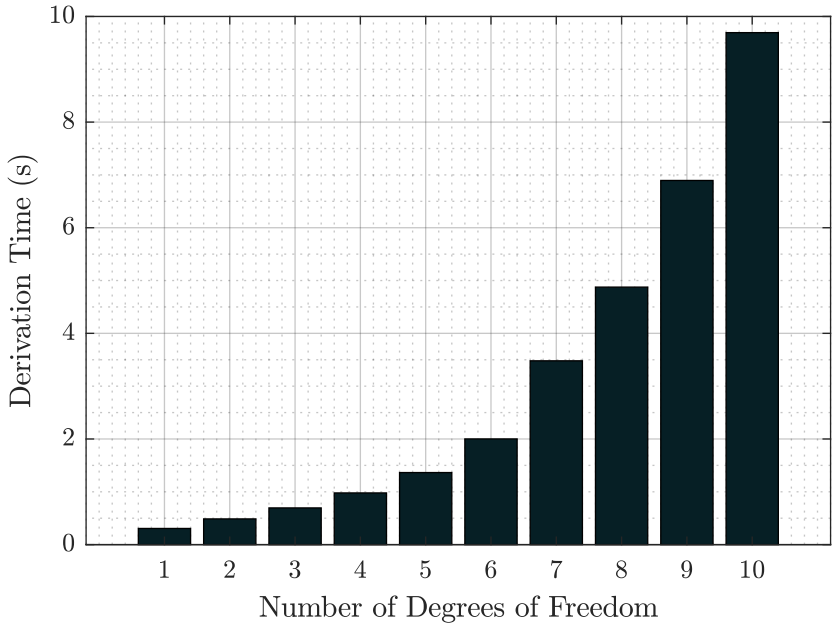


Figure 3.1: Time taken to derive the equations of motion for increasing degree of freedom planar jumping models. Derivations were performed using the symbolic toolbox in MatLab. Reasonable effort was made to optimise the code.

the time it takes for a person to learn and apply the mathematical techniques required to derive dynamic models using analytical mechanics, it is possible to measure the time taken for a computer algebra system to carry out the computations. Figure 3.1 shows the time taken for such derivations to be completed. With the derivations all taking less than 10 seconds to complete, it is clear that the computational cost of such analysis is minimal for low complexity models, however, the intellectual investment is not entirely diminished as the user of a computer algebra system must understand enough mathematics to program the system and validate the results. Furthermore, the user must invest time into learning to program the system. Figure 3.1 shows an exponential increase in the computational cost with the number of degrees of freedom in the model. The trend is approximated as:

$$t_c = 0.22e^{0.38\,n}. \tag{3.1}$$

Where t_c is the computation time of the derivation for a model with n degrees of freedom.

3.2 State Space Analysis

The state space of a system (known as phase space in some optimal control literature) is the set of all configurations which the system may have. The dynamics of a system may be used to describe how the state of the system changes depending on the system's state and the applied control action. By analysing a system's state space, one may gain insights into the behaviours of that system in a generally applicable sense. Providing a description of a system's behaviour

which applies to the entire state space generally requires the system to have linear dynamics. The articulated leg models considered in this work do not exhibit linear dynamics. This section presents an example of state space analysis applied to a 1 degree of freedom model from the literature, and then applies the methodology to a 2 degree of freedom model.

3.2.1 1 Degree of Freedom Model

In order to describe the behaviour and control requirements of a 1 degree of freedom jumping system, Roberts et al. divide the state space of the system into regions, with each region having a concise description of the optimal controls required to produce a maximum height jump of the system. In this work, potential alterations to the method used by Roberts et al. are identified which enable the method to be applied to models with multiple degrees of freedom.

The equation of motion for the 1 degree of freedom model used by Roberts et al. is:

$$ml^2\ddot{q} + mgl\cos(q) = \tau. \tag{3.2}$$

The forward dynamics form is:

$$\ddot{q} = \frac{\tau}{ml^2} - \frac{g}{l}\cos(q). \tag{3.3}$$

Roberts et al. divide the state space of the system into regions which can be concisely explained. Properties of a system which can be defined relative to the state and control inputs of the system, such as the vertical ground reaction force, or vertical velocity of the body, may be used to define regions in the system's state space. Regions may reduce the "size" of the state space by neglecting irrelevant states, or may reduce the dimensions of the considered states by effectively removing degrees of freedom from the model. Roberts et al. define a region in the state space within which the system is guaranteed to take-off, breaking contact with the ground instantly. This region is defined by the vertical ground reaction force falling below zero:

$$F_v \leq 0. \tag{3.4}$$

This definition requires the assumption that the system cannot grip the ground to pull itself down. The vertical ground reaction force of the 1 degree of freedom model is defined as:

$$F_v = m\left(-l\dot{q}^2\sin(q) + \frac{\cos(q)}{ml}\tau + g\sin^2(q)\right). \tag{3.5}$$

Roberts et al. point out that a positive torque will always act to increase F_v in the range $0 < q \leq \frac{\pi}{2}$. They do not explicitly explain that this observation is made by looking at the derivative of F_v with respect to τ:

$$\frac{\partial F_v}{\partial \tau} = \frac{\cos(q)}{l}. \tag{3.6}$$

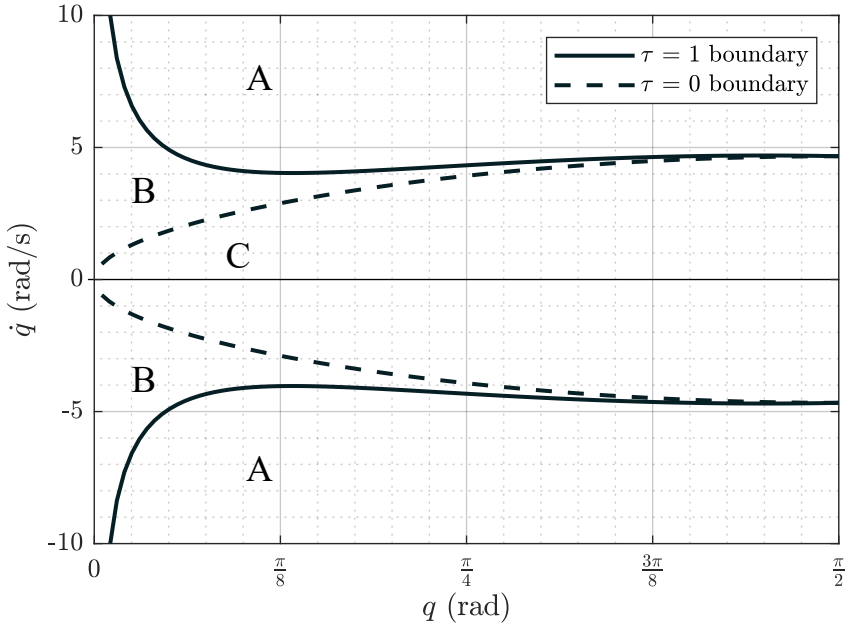


Figure 3.2: Regions of state space for 1 degree of freedom model defined by $F_v \leq 0$. In region (A) the system will break contact immediately with the ground no matter what torque is applied (within the limits of τ_{max}). In region (B) the system is able to jump by applying a smaller than maximum torque. In region (C) the system is unable to break contact with the ground. Model parameters used: $l = 0.45\,m$; $m = 1.5\,kg$; $g = 9.81\,ms^{-2}$; $\tau_{\text{max}} = 1\,Nm$.

Derivatives in models are useful in determining the effects of control inputs on a given metric or property. In this case, if $\frac{\partial F_v}{\partial \tau} \geq 0$ then an increase in applied torque would increase the ground reaction force and allow the system to maintain contact with the ground at higher angular velocities.

In order to successfully jump, the system must enter a region of the state space in which the ground contact force becomes zero. Upon entering the region, the system will break contact with the ground causing the dynamics of the system to change, thus, only the boundary of this region is of particular interest. Assuming that the system starts outside of the region and is able to produce a positive vertical ground reaction force, the boundary is defined by:

$$F_v = 0. \tag{3.7}$$

Roberts et al. find the boundary of the region by substituting equation (3.7) into (3.5) to eliminate $\ddot{\boldsymbol{q}}$, the result is solved to make $\dot{q}$ the subject:

$$\dot{q} = \pm\sqrt{\frac{\cos(q)}{ml^2\sin(q)}\tau + \frac{g}{l}\sin(q)} \tag{3.8}$$

Figure 3.2 shows the regions for negative angular velocity which Roberts et al. do not include. Roberts et al. propose that one optimal control strategy for the system is to fall to the ground and then apply maximum torque from there. However, while falling towards the ground, the system may break contact with the ground and thus must use control torque to prevent this. This is a technicality as breaking contact with the ground while falling to the ground may not be an issue, however, it is reported here for completeness. The remaining boundary

derivations by Roberts et al. are neglected in this work as they represent specific behaviours of the 1 degree of freedom model which are not obviously relevant to higher degree of freedom models.

Lollipop Experiment

An empirical demonstration of a single degree of freedom system breaking contact with the ground as it falls is presented here. A real-world system which is comparable to the single degree of freedom model is a lollipop. The mass of a lolly stick is much smaller than the head of the lolly and so the lollipop may be approximated as a point mass on the end of a massless rigid rod. The experiment cannot perfectly replicate the torque controlled baton model as the no-slip assumption is not easily replicated in reality. The horizontal friction force, F_h, acting between the ground and the lolly stick depends on the vertical reaction force:

$$F_h = \mu F_v \tag{3.9}$$

where μ is the coefficient of static friction between the stick and the ground. As the vertical reaction force approaches zero, the horizontal friction force also approaches zero. This causes the lollipop to slip before it breaks contact with the ground.

Figure 3.3: Purely demonstrative example frames from a recording of a lollipop falling from an almost vertical position. The frames show the lollipop fall, slip and then break contact with the ground. It is to be noted that the rough surface played a big part in the "success" of this experiment as the bumps in the surface likely prevented significant horizontal motion when slipping occurred.

3.2.2 Multiple Degree of Freedom Model

This section demonstrates the application of state space analysis to a 2 degree of freedom planar jumping model. The model, as described in Section 1.11.2, comprises two links with masses m_1 and m_2 and inertias I_1 and I_2. All derivations and mathematical manipulations were performed using the symbolic toolbox in MatLab [38]. The equations of motion, forward dynamics, and vertical ground reaction force are provided in Appendix A. The partial derivative of the vertical ground reaction force with respect to the external torque applied at joint 1 is:

$$\frac{\partial F_v}{\partial \tau_1} = \frac{P}{Q} \tag{3.10}$$

where

$$P = 2\,l_1 \Big({l_2}^2\,{m_2}^2\,\cos(q_1) + I_2\,m_1\,\cos(q_1) + 2\,I_2\,m_2\,\cos(q_1) \\ - {l_2}^2\,{m_2}^2\,\cos(q_1 - 2\,q_2) + {l_2}^2\,m_1\,m_2\,\cos(q_1) \Big)$$

$$Q = 4\,{l_1}^2\,{l_2}^2\,{m_2}^2\,\sin^2(q_1 - q_2) + m_1\,{l_1}^2\,{l_2}^2\,m_2 \\ + 4\,I_2\,{l_1}^2\,m_2 + I_2\,m_1\,{l_1}^2 + 4\,I_1\,{l_2}^2\,m_2 + 4\,I_1\,I_2.$$

The partial derivative of the vertical ground reaction force with respect to the external torque applied at joint 2 is:

$$\frac{\partial F_v}{\partial \tau_2} = \frac{R}{S} \tag{3.11}$$

where

$$R = l_2\,m_2 \Big(4\,I_1\,\cos(q_2) + 2\,{l_1}^2\,m_2\,\cos(q_2) - {l_1}^2\,m_1\,\cos(2\,q_1 - q_2) \\ - 2\,{l_1}^2\,m_2\,\cos(2\,q_1 - q_2) \Big)$$

$$S = 4\,{l_1}^2\,{l_2}^2\,{m_2}^2\,\sin^2(q_1 - q_2) + m_1\,{l_1}^2\,{l_2}^2\,m_2 \\ + 4\,I_2\,{l_1}^2\,m_2 + I_2\,m_1\,{l_1}^2 + 4\,I_1\,{l_2}^2\,m_2 + 4\,I_1\,I_2.$$

The denominators, Q and S, of equations (3.10) and (3.11) are strictly positive given that all of their terms comprise mass, inertia and link length parameters with the exception being a squared sin function, which is also strictly positive. The numerators of equations (3.10) and (3.11) do not hold such properties; they depend on the joint angles, q_1 and q_2. Thus, a general relationship between the applied torque and the ground reaction force cannot be deduced here. To overcome this, the second segment in the model is constrained to move relative to the first, this effectively reduces the model to a single degree of freedom and means that the system's configuration is defined by only the first joint. The application of this constraint reduces the dimension of the state space of the model to the same as the state space for a 1 degree of freedom model. However, the behaviours and properties of the model are a hybrid of the two models; the inertia of the constrained 2 degree of freedom system changes with state as the two links move relative to each other, capturing the inertial effects of the leg's motion. This use of constraints also affects the information obtainable through the model: an external torque is only applicable at the first joint, the external torque applied at the second joint is defined by the constraints and cannot be varied independently of the torque at the first joint.

The second joint in the system is constrained to move relative to the first by:

$$q_2 = aq_1 + c. \tag{3.12}$$

The velocity and acceleration of the second joint become:

$$\dot{q}_2 = a\dot{q}_1 \quad \text{and} \quad \ddot{q}_2 = a\ddot{q}_1. \tag{3.13}$$

The constraint parameters a and c describe the motion of joint 2 relative to joint 1. This fully defines the kinematic motion of the system and so the parameters may be selected to produce a desired motion in the system. This could be done by defining a start and final state of the system, $\boldsymbol{q}_s$ and $\boldsymbol{q}_f$, and finding the values of a and c which produce those states:

$$a = \frac{q_{2f} - q_{1f}}{q_{2s} - q_{1s}} \tag{3.14}$$

$$c = q_{2s} - aq_{1s}. \tag{3.15}$$

Alternatively, the values may be selected through inspection by plotting the pose of the model and adjusting a and c as needed.

3.2.3 Method

To demonstrate the constraint method, the values for a and c were chosen as:

$$a = -1 \quad \text{and} \quad c = \pi.$$

This gives a reasonable looking motion of the system in the range of $\frac{1}{8}\pi \leq q_1 \leq \frac{1}{2}\pi$ as demonstrated in figure 3.4. The actual parameters selected for the model in this process will not change the method's application, the resulting description of the system is likely to vary based on the chosen parameters, however, the means of determining such a description would remain the same. After the method has been implemented, it is a case of inputting a different set of parameters into the resulting equations to determine the characteristics of the different system.

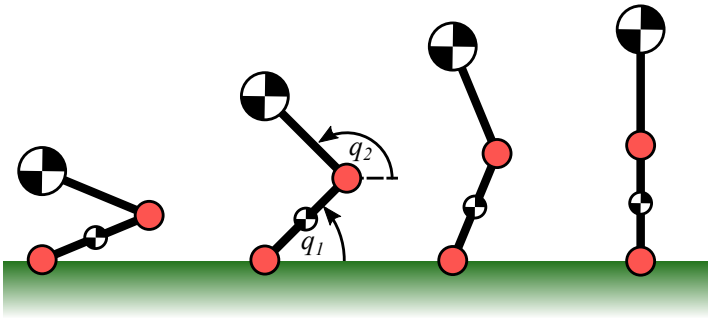


Figure 3.4: Jumping motion of the constrained system starting from $q_1 = \frac{1}{8}\pi$. The angle q_2 is constrained to be $q_2 = -q_1 + \pi$. Both links have length $0.225\,m$. The values of the constraint parameters were selected to produce a plausible vertical jumping motion.

The remaining model parameters were selected arbitrarily as: $l_1 = 0.225\,m$; $l_2 = 0.225\,m$; $m_1 = 0.75\,kg$; $m_2 = 1\,kg$. Any parameter values could be selected at this stage as the equations for the ground reaction force of the model was derived and is available in Appendix A. The equation of motion for the system was derived using Lagrangian mechanics and the regions in which the system breaks contact with the ground were identified using equation (3.7). For the given system parameters, the ground reaction force becomes:

$$F_v = \frac{0.534\cos(q)\Big(0.101\sin(2q)\dot{q}^2 + \tau - 5.24\cos(q)\Big)}{0.101\cos(2q) + 0.113} - 0.534\dot{q}^2\sin(q) + 17.2 \quad (3.16)$$

The derivative of F_v with respect to the applied torque, τ, is:

$$\frac{\partial F_v}{\partial \tau} = \frac{0.534\cos(q)}{0.101\cos(2q) + 0.113}. \quad (3.17)$$

Equation (3.17) is positive for all values of q and so the method of Roberts et al. may be applied in deriving the boundaries of this parameterised system. For any value of q, an increase in τ will lead to an increase in F_v, thus the minimum and maximum torques may be used to define the ground contact breaking capabilities of the system. For this example, the torque is bounded as $0 \leq \tau \leq 1$.

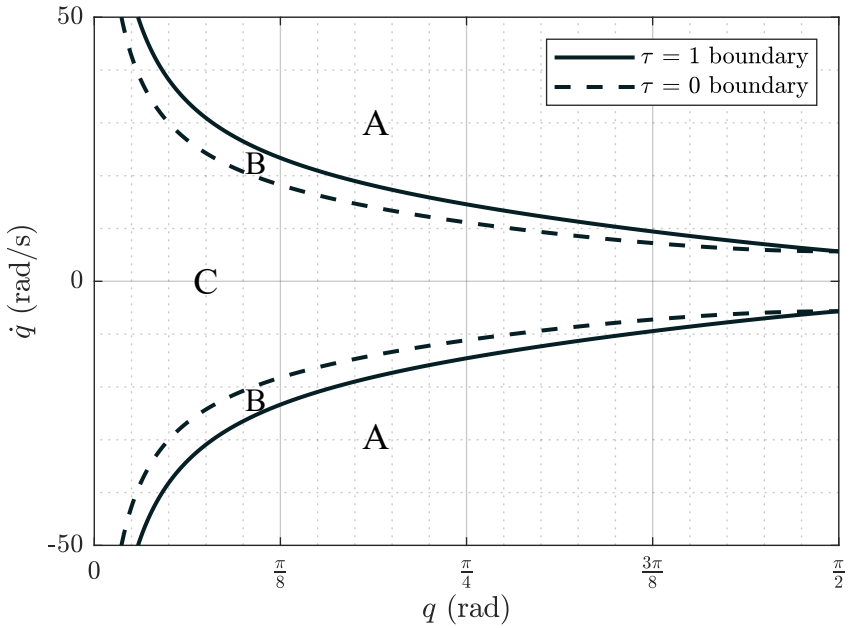


Figure 3.5: The regions of the constrained 2 degree of freedom system which define the system's capability to break contact with the ground. The second joint angle of the system is constrained to move relative to the first as: $q_2 = -q_1 + \pi$. The regions are: (A) the system will break contact with the ground no matter what, (B) the system may break contact with the ground by applying sufficient torque within the limits of the system, (C) the system has no means of breaking contact with the ground. For this system, both boundaries have the asymptote $q = 0$.

3.2.4 Stuck to the Ground

The boundary plot 3.2 shows that the 1 degree of freedom system is able to break contact with the ground from an arbitrary joint angle $q_1 > 0$. This is not the case for the constrained 2 degree of freedom system. Due to the way the motion of the second segment is constrained to the first, it is possible for some sets of constraint parameters to render the system unable to produce a ground reaction force $F_v \leq 0$ for certain joint angles. A second set of constraint parameters were used to demonstrate this effect more clearly:

$$a = -1 \quad \text{and} \quad c = 0.8\pi.$$

The motion of this system is shown in figure 3.6. The system is constrained to move in a similar way to the vertical jumping system.

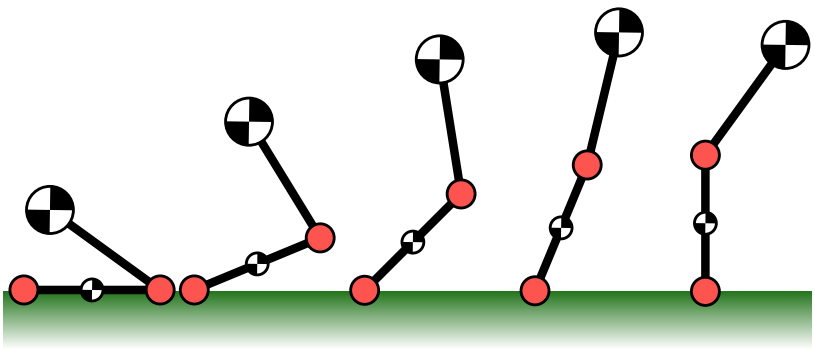


Figure 3.6: Jumping motion of the constrained system starting from $q_1 = 0$. The angle q_2 is constrained to be $q_2 = -q_1 + 0.8\pi$. Both links have length $0.225\,m$. This configuration of the constrained motion has no means of breaking contact with the ground in the range $0 \leq q_1 \leq 0.45$.

Following the same derivations as described above, the regions for this model are shown to change, leaving a region in the range $0 \leq q_1 \leq 0.45$ in which the system cannot produce a negative ground reaction force for any joint velocity, in the limits of $0 \leq \tau \leq 1$. This means the system is entirely unable to jump under the aforementioned constraints.

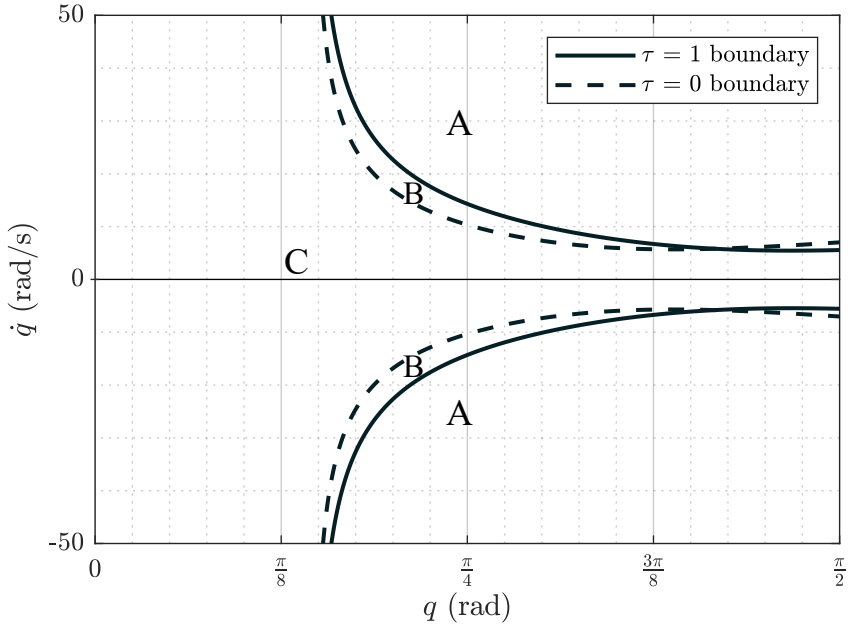


Figure 3.7: The regions of the constrained 2 degree of freedom system which define the system's capability to break contact with the ground. The second joint angle of the system is constrained to move relative to the first as: $q_2 = -q_1 + 0.8\pi$. The regions are: (A) the system will break contact with the ground no matter what, (B) the system may break contact with the ground by applying sufficient torque within the limits of the system, (C) the system has no means of breaking contact with the ground. For this system the boundaries have an asymptote at $q = 0.45$ meaning that the system has no means of breaking contact with the ground for $q \leq 0.45$.

3.2.5 Discussion

This section demonstrated the analysis of model equations for a model with multiple degrees of freedom. A computer algebra system was leveraged to derive and manipulate the complex equations of the 2 degree of freedom model. This novel application demonstrated that analysis of model equations has more flexibility than the lack applications found in the literature might suggest.

Analysis of the 2 degree of freedom system revealed insights into the jumping capabilities of the system. Particularly, how performing certain motions can result in a jumping system being unable to break contact with the ground, regardless of the amount of torque is applied to one of the system's joints.

The derivation of algebraic equations for a system model is a widely useful result for a predictive modelling method as any parameterisation of the system may be plugged into the equations and insights quickly gained from them. However, the complexity of the equations produced by this work limits their utility to other researchers. In consideration of this complexity, constraints were applied to the model to simplify the resulting equations. The use of constraints allowed for plotting of state space boundaries and meaningful insights to be obtained for the system, such as the system's inability to break contact with the ground under specific circumstances. The combined complexity of the model, and of the constraints used to simplify the model, mean that the insights gained through this method require considerable intellectual investment both to present and interpret.

It is to be noted that the intellectual investment cost mentioned here does not imply that the user must be more intelligent; the intellectual investment cost incurred is an increase in the mental load on the user. The user must hold an increased number of concepts in their mind when working with the method and presenting, or interpreting, the results.

Equation 3.1 implies there exists an exponential increase in the computation cost of derivation with increasing degrees of freedom for planar jumping dynamics models. This implies that the use of computer algebra systems to derive algebraic equations of motion for dynamic jumping systems may become infeasible at higher levels of model complexity. For example, equation 3.1 predicts a time of 455 days to derive the equations of motion for a 50 degree of freedom model. Many works in the literature skip the algebraic derivation of equations and use numerical methods to determine the inertia, Coriolis, and gravity matrices to construct the dynamic equations of motion for a particular system [21]. Such methods are limited to the parameterisation of the model used, and are not feasible for use in analytical methods such as the one presented in this work.

Implementation of this method required knowledge in mechanics, dynamics, multivariate calculus, and programming. Anecdotally, the most difficult aspect of using the method was in programming the computer algebra system to perform the derivations of the equations of motion. At the time of writing, MatLab's symbolic toolbox does not facilitate tasks such as performing derivatives with respect to multiple variables. Instead, the symbolic toolbox was used to compute partial derivatives and their products to manually obtain derivatives with respect to multiple variables. This is a limitation of the use of the tool as it requires the user to understand the workings of multivariate calculus at an implementation level.

It is difficult to quantify the intellectual investment of a method as though it were computational cost. Concerning the chronological order of work done in this thesis, this chapter was completed after the other technical works had been completed. Anecdotally, the analysis required little intellectual investment to perform. It is noted that many of the tools used in this chapter were practiced in the other technical chapters. It is also observed that some researchers may be unlikely to openly admit they had difficulty learning some concepts.

The intellectual investment required in learning a concept or method is difficult to assess retrospectively as it can be difficult to empathise with oneself being ignorant to a concept which one now knows. It may be interesting to survey undergraduate students on the time they invest into learning various concepts to determine a more quantifiable cost for the intellectual investment of these methods.

This study could be extended through the derivation of additional equations for low complexity jumping systems. A repository of algebraic equations, implemented in various programming languages, for jumping dynamics models may be useful for future researchers who do not have the intellectual or computational resources available to quickly produce them.

3.3 Dynamic Balance Ellipsoid

Dynamic balance in jumping requires the zero moment point of the ground reaction force to remain inside of the foot-ground contact area. The zero moment point is a common metric which is used in the control of walking robots to ensure that the foot remains flat on the ground and the system does not fall over [37]. This section presents the "Balance Ellipsoid", a novel method for establishing the bounds of the feasible acceleration space for dynamically balanced jumping. The method is inspired by the velocity ellipsoid, a common description of manipulability in robotics [21], and the work done by Parslew et al. to describe the dynamic balance of single degree of freedom jumping systems. An acceleration ellipsoid is constructed using the system's Jacobian matrix and a spherical set of joint angular accelerations, a static variation of this method was used by Khatib and Burdick for optimising the structure of a manipulator [29]. The relationship between the accelerations of the body during balanced motion is linear, meaning that a given zero moment point location on the foot will produce a plane in the body acceleration space. The intersection of such a plane with the acceleration ellipsoid may produce an ellipsoid which represents the bounds of the set of feasible accelerations. No intersection implies that the system is unable to maintain balance given the zero moment point location constraint and the radius of the spherical set of joint angular accelerations.

The following sections present the inspiration for the work, the components required for the derivation and then the derivation of the dynamic balance ellipsoid.

3.3.1 Velocity Ellipsoid

The velocity ellipsoid is a tool used in manipulator design and control. It represents the transformation from joint angular velocities, $\dot{\boldsymbol{q}}$, to body velocity, $\dot{\boldsymbol{x}}$. This transformation is shown geometrically by a spherical input of joint angular velocities being squashed, or stretched, into an ellipsoid of body velocities. The transformation is based on the Jacobian matrix of the system and so singular value decomposition of the Jacobian can be used to determine the directions and magnitudes of the velocity ellipsoid's axes. The velocity ellipse is derived using a spherical set of joint angular velocities with radius r:

$$\dot{\boldsymbol{q}}^T \dot{\boldsymbol{q}} = r^2. \tag{3.18}$$

The inverse of the Jacobian relationship, given by:

$$\dot{\boldsymbol{q}} = \boldsymbol{J}^{-1} \dot{\boldsymbol{x}}, \tag{3.19}$$

is substituted into (3.18) to arrive at:

$$\dot{\boldsymbol{x}}^T (\boldsymbol{J}\boldsymbol{J}^T)^{-1} \dot{\boldsymbol{x}} = r^2. \tag{3.20}$$

While the inverse of the Jacobian is only defined when the Jacobian is a square matrix, i.e. the number of degrees of freedom of the body and the leg are equal, in equation (3.20) it is the product $\boldsymbol{J}\boldsymbol{J}^T$ which is inverted. The product of a matrix and its transpose will always produce a square matrix, which has a well defined inverse, and thus equation (3.20) is valid even for redundant systems.

3.3.2 Acceleration Ellipsoid

The acceleration ellipsoid builds on the work done by Khatib and Burdick [29], in which they optimise a manipulator based on its acceleration capabilities within a working space. Khatib and Burdwick assume the manipulator to have no velocity and thus work with an identical entity to the velocity ellipsoid as $\ddot{\boldsymbol{x}} = \boldsymbol{J}\ddot{\boldsymbol{q}}$ in the static case. The dynamic acceleration ellipsoid includes the accelerations induced in the body by the angular velocities of the joints. The acceleration ellipsoid is derived using the kinematic relationship defined in equation (1.6):

$$\ddot{\boldsymbol{q}} = \boldsymbol{J}^{-1}(\ddot{\boldsymbol{x}} - \dot{\boldsymbol{J}}\dot{\boldsymbol{q}}). \tag{3.21}$$

Note that the acceleration ellipsoid is not the same as the static wrench transmission which is given by

$$\boldsymbol{\tau} = \boldsymbol{J}^T\boldsymbol{f},$$

where $\boldsymbol{\tau}$ is the vector of torques applied to each joint in the leg and $\boldsymbol{f}$ is the vector of forces and torques applied at the body. A sphere of joint accelerations, with radius r, is used as input:

$$\ddot{\boldsymbol{q}}^T\ddot{\boldsymbol{q}} = r^2. \tag{3.22}$$

Substituting equation (3.21) into (3.22) gives the acceleration ellipsoid equation:

$$(\ddot{\boldsymbol{x}} - \dot{\boldsymbol{J}}\dot{\boldsymbol{q}})^T(\boldsymbol{J}\boldsymbol{J}^T)^{-1}(\ddot{\boldsymbol{x}} - \dot{\boldsymbol{J}}\dot{\boldsymbol{q}}) = r^2. \tag{3.23}$$

The acceleration ellipsoid is equivalent to the velocity ellipsoid in size, eccentricity and orientation, however, the acceleration ellipsoid is offset from the origin by the acceleration produced on the body by the centripetal accelerations of each segment in the leg, denoted by the $\dot{\boldsymbol{J}}\dot{\boldsymbol{q}}$ term. When treating the system as static, the product $\dot{\boldsymbol{J}}\dot{\boldsymbol{q}} = \boldsymbol{0}$ and the acceleration ellipsoid will centre about the origin, becoming identical to the velocity ellipsoid.

The relation between joint accelerations and body accelerations may not be useful in control applications as the accelerations do not directly relate to the work being done to the system and thus is not a clear indication of the required or available actuation effort. It would be preferable to show the relationship between the externally applied torques at each joint in the leg, and the resulting forces and torques on the body - in a dynamic setting. This is a potential point of further work for this method.

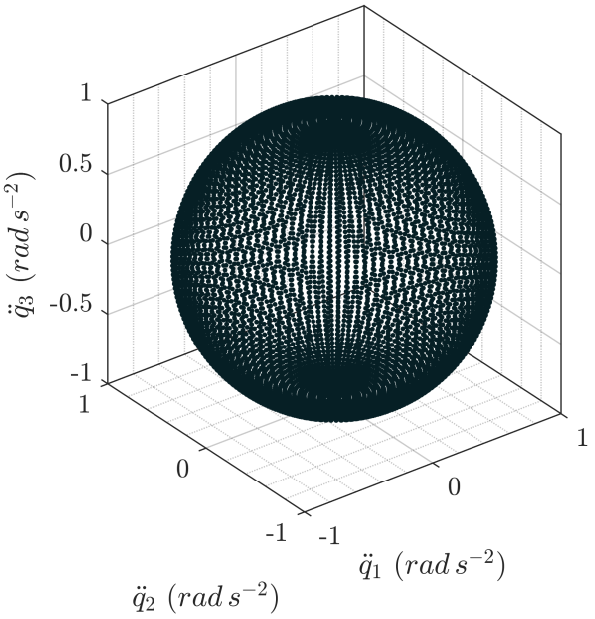


Figure 3.8: Spherical set of joint acceleration vectors with $r = 1$, each vector in the set is plotted as a point in joint space.

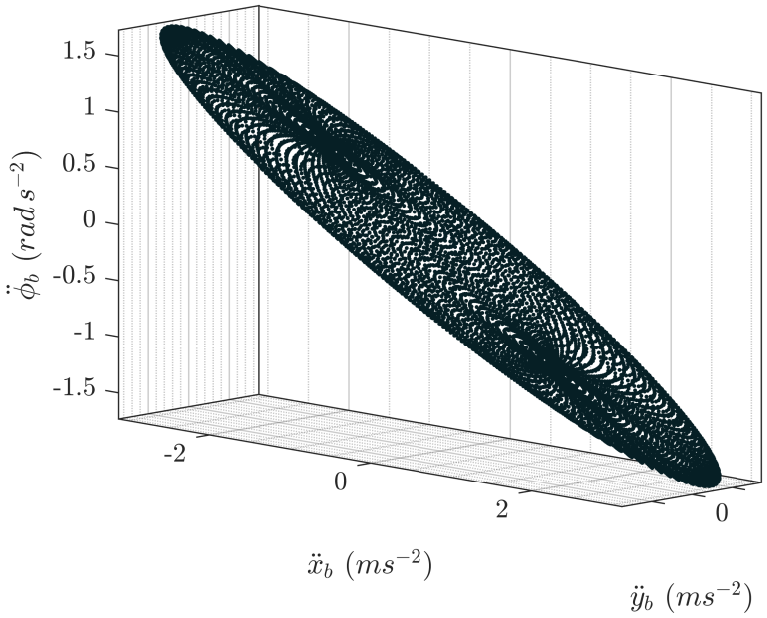


Figure 3.9: Example ellipsoid of body accelerations produced by the spherical set of joint angular accelerations.

3.3.3 Balance Plane

The balance plane is the set of angular and linear accelerations of the body which result in a prescribed zero moment point location. A common metric of balance in legged locomotion is the zero moment point. A review of applications of the zero moment point can be found at [37]. A legged system accelerates its body by pushing against the substrate it stands on. The substrate provides a reaction pressure which acts over the area of contact between the foot and the substrate. This pressure may be resolved into a reaction force and a reaction moment acting at a single position on the ground. There exists a position where the reaction force may be resolved such that the reaction moment is zero, hence "zero moment point". If this point lies within the contacting area of the foot, then the system is considered balanced and the foot will not rotate or tip. If the zero moment point lies outside of the contact area of the foot then the zero moment point is not physically realisable and the substrate will produce both a reaction force and reaction moment on the foot, causing it to rotate.

Parslew et. al derive the location of the zero moment point, x_p, by considering the system as having only an inertial body; the mass and rotational inertia of the leg are assumed to be negligible:

$$x_p = x_b - \frac{y_b}{\tan(\theta)} + \frac{\tau_b}{\|\boldsymbol{r}_p\| \sin(\theta)} \tag{3.24}$$

where θ is the angle made by the horizontal and vertical accelerations of the body. Thus

$$\tan(\theta) = \frac{\ddot{y}_b - g}{\ddot{x}_b} \quad \text{and} \quad \sin(\theta) = \frac{m(\ddot{y}_b - g)}{\|\boldsymbol{r}_p\|}. \tag{3.25}$$

The net reaction force, $\boldsymbol{r}_p$, is defined as:

$$\boldsymbol{r}_p = m\sqrt{\ddot{x}_b^2 + \ddot{y}_b^2} - mg. \tag{3.26}$$

The torque applied to the body is defined as:

$$\tau_b = I\ddot{\phi}_b. \tag{3.27}$$

By substituting equations (3.25), (3.26) and (3.27) into equation (3.24), the zero moment point may be represented in terms of the body accelerations $\ddot{x}_b$, $\ddot{y}_b$ and $\ddot{\phi}_b$.

$$-my_b\ddot{x}_b + m(x_b - x_p)\ddot{y}_b + I\ddot{\phi}_b - mg(x_b - x_p) = 0 \tag{3.28}$$

Equation (3.28) is the equation of a plane in the body acceleration space. Both the offset from the origin and the normal of the plane depend on the zero moment point position, x_p. The balance plane represents the feasible sets of linear and angular accelerations of the body for a given value of x_p. The body acceleration space may be constrained by considering only values of x_p which are inside the contact area of the foot.

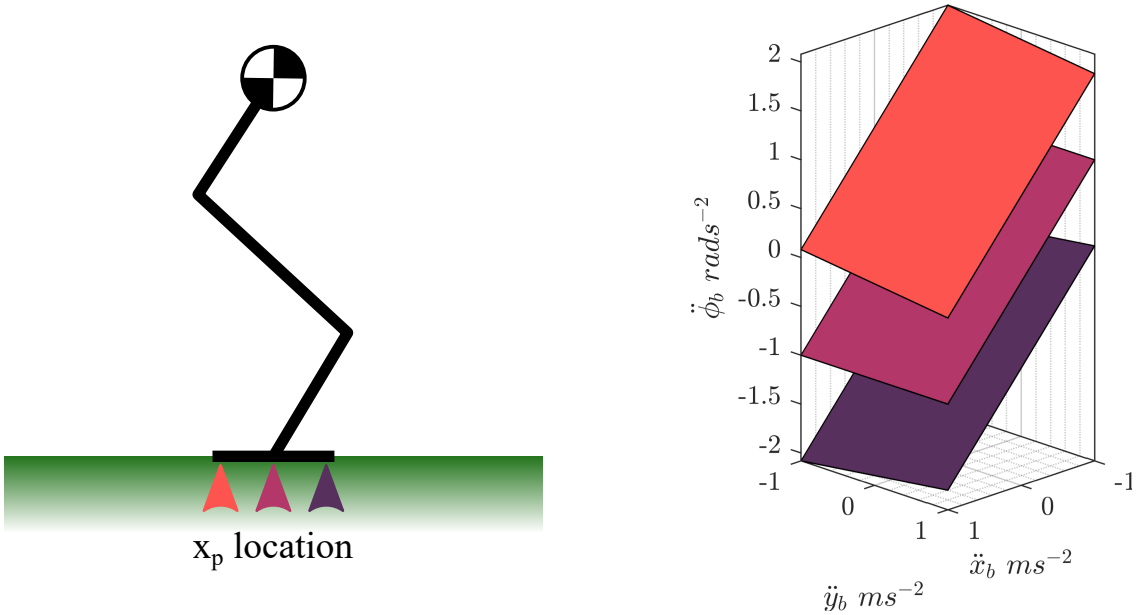


Figure 3.10: Example balance planes for zero moment point locations at the back, centre and front of the foot. The body of the system may move with any acceleration on the plane while maintaining the prescribed zero moment point. In this example $x_b = 0$, $y_b = 1$, $m_b = 1$ and $I_b = 1$

3.3.4 Balance Ellipsoid

The balancing capabilities of a jumping system may be determined by considering the intersection of the acceleration ellipsoid and the balance plane. If the two shapes do not intersect, then the system is unable to achieve dynamically balanced motion given the constraints of the zero moment point, x_p, and the limit of the joint angular acceleration vector norm, r.

The intersection of the acceleration ellipsoid and the balance plane is found by equating equations (3.23) and (3.28).

$$\boldsymbol{P}\ddot{\boldsymbol{x}} - mg(x_b - x_p) = (\ddot{\boldsymbol{x}} - \dot{\boldsymbol{J}}\dot{\boldsymbol{q}})^T(\boldsymbol{J}\boldsymbol{J}^T)^{-1}(\ddot{\boldsymbol{x}} - \dot{\boldsymbol{J}}\dot{\boldsymbol{q}}) - r^2 \tag{3.29}$$

The matrix $\boldsymbol{P}$ is the collection of acceleration coefficients:

$$\boldsymbol{P} = \begin{bmatrix} -my_b & 0 & 0 \\ 0 & m(x_b - x_p) & 0 \\ 0 & 0 & I \end{bmatrix}. \tag{3.30}$$

To simplify the solution to the intersection, equation (3.28) was solved for the angular acceleration and substituted into equation (3.29) to project the ellipsoid onto the linear acceleration, ($\ddot{x}_b$, $\ddot{y}_b$), plane. The ellipsoid may be projected into any plane in the body acceleration space or may be used directly in the 3D or 4D space, the projection was used here to simplify the presentation of results. The projection ignores the angular acceleration of the body in the results and may be suited to the study of jumping systems where the angular velocity at take-off is not a concern. However, the body still undergoes angular acceleration and each vector of body acceleration will have an angular acceleration associated to it.

A nominal, 3 degree of freedom model was used to verify the results. The point foot of the model, i.e. the location of the first joint, was set as the origin of the Cartesian coordinate system. The parameters of which were:

$$\boldsymbol{l} = \begin{bmatrix} 1 \\ 1.5 \\ 1 \end{bmatrix}, \quad \boldsymbol{q} = \begin{bmatrix} \frac{3\pi}{4} \\ \frac{\pi}{4} \\ \frac{3\pi}{4} \end{bmatrix}, \quad x_p = 0.$$

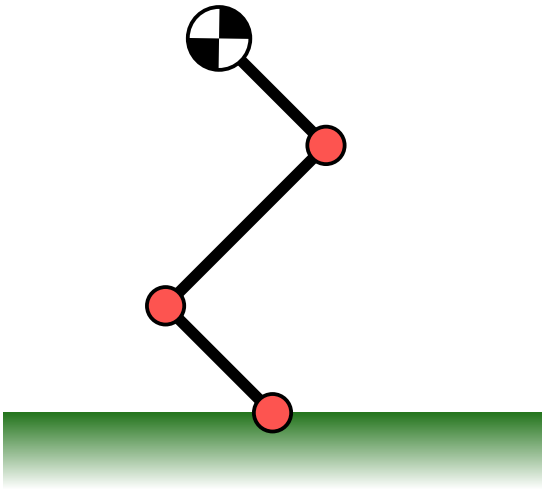


Figure 3.11: Nominal 3 degree of freedom model used to verify the dynamic balance ellipsoid derivation.

The mathematical solution for the balance ellipsoid was verified numerically by calculating the body acceleration and the resulting value of x_p for a discrete set of joint acceleration values which satisfied:

$$\ddot{\boldsymbol{q}}^T \ddot{\boldsymbol{q}} = 1. \tag{3.31}$$

The joint accelerations and body accelerations which produced a zero moment point $\epsilon \leq x_p \leq \epsilon$, where ϵ was a small tolerance value, were recorded and compared to values predicted by the analytical solution. These verification results are shown in figures 3.12 and 3.13.

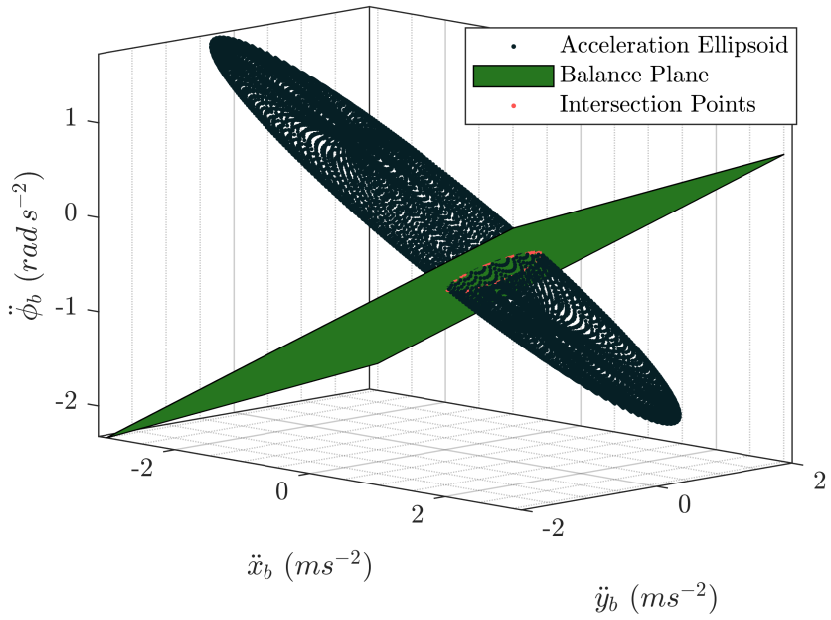


Figure 3.12: Intersection between the acceleration ellipsoid and the balance plane for the given model parameters. The numerically determined points of intersection are shown in orange.

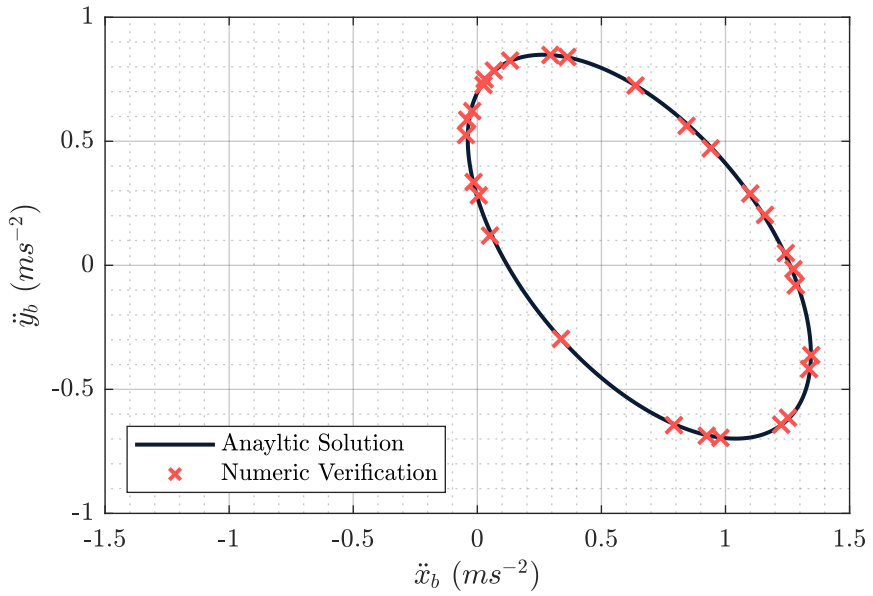


Figure 3.13: The analytically determined balance ellipsoid and the numerically determined balance values.

The numeric values deviate slightly from the analytic solution. This is to be expected as the grid search required a tolerance to find values close to the answer due to the practical impossibility of a grid search finding exact values. The same derivation may be applied to a range of values for the zero moment point. Figure 3.14 shows the balance ellipsoid for

$x_p = -0.25, 0, 0.25$ m.

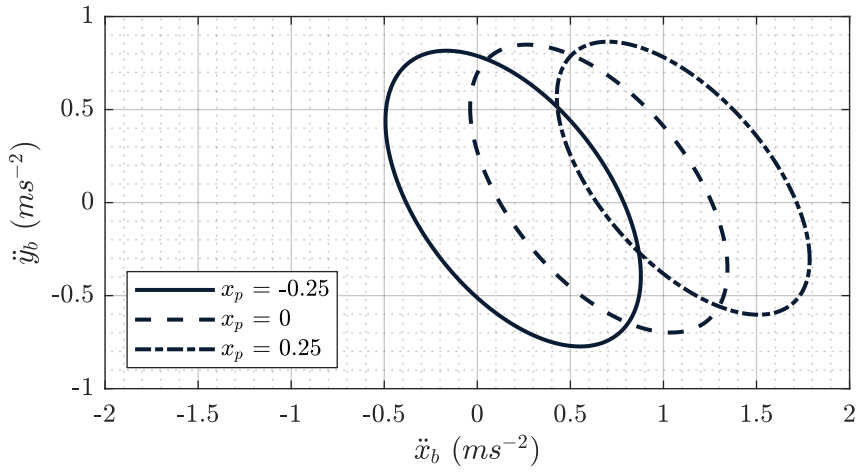


Figure 3.14: The analytically determined balance ellipsoids for $x_p = -0.25, 0, 0.25$ m.

The variation in the ellipsoid is more extreme when the body has greater horizontal distance from the foot. To demonstrate this effect, the model configuration was changed such that:

$$\boldsymbol{q} = \begin{bmatrix} \frac{3\pi}{5} \\ \frac{\pi}{10} \\ \frac{7\pi}{20} \end{bmatrix}.$$

This produced a leaning system with the body far ahead of the foot as shown in figure 3.15.

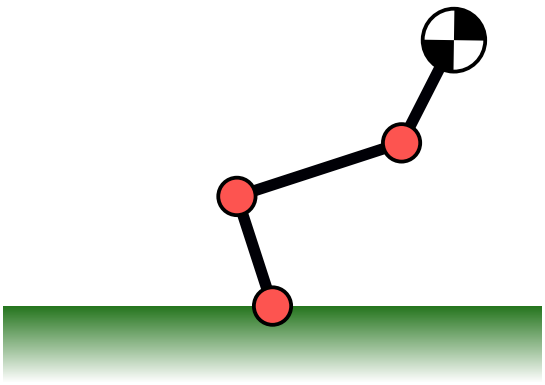


Figure 3.15: Leaning model used to demonstrate variation of the dynamic balance ellipsoid with zero moment point location and leg configuration.

This produced significantly greater variation in the balance ellipsoids for the given range of

zero moment point locations, as shown in figure 3.16.

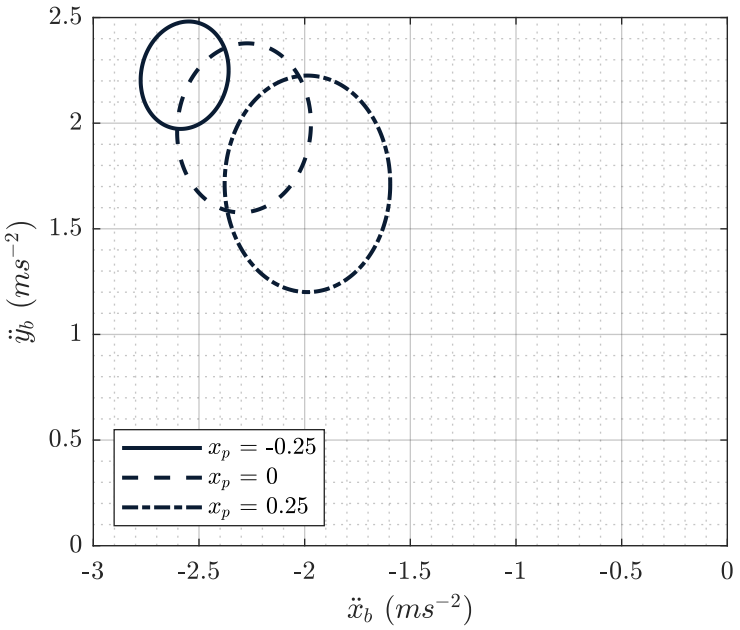


Figure 3.16: The analytically determined balance ellipsoids for $x_p = -0.25, 0, 0.25$ for the leaning model. Notice that the system is able to accelerate the body in a greater range of values when the zero moment point is at the front of the foot, closer to the body.

3.3.5 Discussion

The dynamic balance ellipsoid concept is a novel technical contribution of this work. The ellipsoid provides a tool to aid the investigation of the feasible accelerations for a jumping system. The tool is flexible as it may be used in relation to any subset of the body accelerations.

The derivation of the dynamic balance ellipsoid makes use of multiple conceptual components which gives the tool a considerable intellectual investment cost to use.

Due to the assumption of a massless leg, the tool will only provide an approximate solution to balance problems and may be inappropriate for control applications if margins for error are not allowed or the legs of a system constitute a significant fraction of the total mass of the system. As with work done using similar methods [29], the dynamic balance ellipsoid may be used to provide metrics for the optimisation of jumping system structures. The tool is limited by the requirement of a state for the system, this means that the user must provide a trajectory or set of configurations of the system for analysis.

The ellipse shown in figure 3.13 represents all of the linear accelerations which may be produced in the body of the system, given that the norm of the joint angular acceleration vector satisfies $\|\ddot{q}\| = 1$ and that the zero moment point of the ground reaction force is at $x_p = 0$. The size of the ellipse scales in proportion to $\|\ddot{q}\|$, thus any point inside of the ellipse constitutes

a feasible motion if the joint accelerations are allowed to be decreased. In other words, the ellipse represents the maximum attainable body accelerations for a given joint acceleration magnitude.

For a system with a foot that may be considered as a point, such as SALTO [30], this ellipse represents the set of linear accelerations which may be produced on the body. As the ellipse was projected onto the XY plane, the required angular acceleration is not shown. However, each point on the projected ellipse will have an accompanying angular acceleration which is required to maintain the dynamic balance of the system. For systems with a foot segment, the ellipse would be derived for zero moment point values ranging from the front to the back of the foot.

The dynamic balance ellipsoid is derived using the kinematic equations for a jumping system. As such, the tool does not properly consider the externally applied torques produced by actuators of the system. Limits in accelerations are connected to limits in forces and torques, however, they are not directly related and may prove to be misleading in this regard.

The results presented in this section demonstrate the use of the dynamic balance ellipsoid to investigate the feasible accelerations for a 3 degree of freedom model. The tool was used to explore the effects of varying the pose and desired centre of pressure location on the feasible body accelerations of the system. This work provides additional functionality to the work done by Parslew et al. [34] by showing the variation in acceleration vector magnitudes and allowing for easy comparison of different location for the system's centre of pressure.

The results in Section 3.3.4 show that the example jumping system is unable to accelerate its body in the positive x direction given the constraints to joint accelerations and centre of pressure location.

The range and utility of the results in this section would be improved by the inclusion of jumping systems in motion, i.e. with joint velocities not equal to zero. The dynamic balance ellipsoid could be used to explore the set of feasible body accelerations throughout a nominal jumping motion, providing insight into the dexterity of the system in terms of its ability to accelerate in different directions.

3.4 Conclusions

The work in this chapter demonstrated the application of algebraic analysis to models of low complexity, despite the lack of applications found in the jumping literature. Analysis of low complexity models had higher requirements in intellectual investment and computational cost than analysis of single degree of freedom models seen in the literature.

Analysis of the 2 degree of freedom model revealed insights into the capabilities of the jumping system, including the limitations on the system's ability to break contact with the ground when the segments of the system moved in particular ways relative to each other.

The algebraic equations which resulted from the analysis of low complexity models were found to be too complex to easily plot or concisely derive insight from. While this work demonstrated the feasibility of deriving such systems of equations, additional work is required from the user of the method to distill the results into meaningful insights into jumping systems.

The constraints used in Section 3.2.2 reduced the complexity of the results of the analysis while also limiting their implication to the constrained, and parameterised, variation of the model, instead of the general 2 degree of freedom jumping model.

The trend of computation time described by equation 3.1 implies that the utility provided by computer algebra systems has a limit in model complexity, with the example case of a 50 degree of freedom model requiring an estimated 455 days of computation time to derive the equations of motion.

The intellectual investment required for analysis was difficult to quantify as this chapter was completed after the previous works and the author was well practiced in many of the components required for the method. It is also noted that anecdotal experience may not be an ideal measure for a method's intellectual investment cost.

This chapter presented a novel technical contribution: the concept of the dynamic balance ellipsoid. The dynamic balance ellipsoid builds on existing work by [34] and provides an effective tool for investigation of the feasible body accelerations throughout dynamic jumping motions.

Empirical results using the dynamic balance ellipsoid demonstrate that the tool can be used to gain insights into the abilities of jumping systems given their configurations and joint acceleration limits.

The dynamic balance ellipsoid is limited by its use of joint acceleration limits; it would be better suited to robotics and biomechanics applications if the tool used torque limits instead. The tool is also limited by the assumption of a massless leg and by the requirement of the user to provide the current state of the system as opposed to a trajectory.

Chapter 4

Static Optimisation

Static optimisation is often used in biomechanics to overcome redundancy in models of musculoskeletal systems to determine the likely muscle forces responsible for the motion of the bones in the skeleton. Redundancy in musculoskeletal systems arises when multiple muscles are used to control a single degree of freedom in the system. This means that the muscle forces are under-determined in relation to the skeletal motion and there are infinite combinations of muscle forces which could produce an observed motion. Static optimisation is typically used to select the combinations of muscle forces which meet selected criteria, such as minimum force or minimum power. Despite the name, static optimisation does not require the system to be stationary. Static optimisation considers the state of the system independently of time and does not consider the entire trajectory that the system takes through state space. This makes the optimisation simple and fast to carry out compared to dynamic optimisation. However, due to the time independence of static optimisation, the results may not be smooth throughout the trajectory; the optimal solution at one moment in time may be vastly different to the solution at another time.

The described use of static optimisation is in an inverse dynamics application, i.e. the optimisation is used to determine redundant forces which are responsible for observed motions. In this work, static optimisation is investigated in an inverse kinematics application. This application was not found in the literature and has been identified as a novel use for the existing method. This chapter explores the application of static optimisation to resolve the kinematic redundancy of jumping models, using the resolved second order kinematics to predict jumping motions for *Numida meleagris*.

The second order kinematics of a jumping model can be used to procure motion of the system without the need to determine torques or forces acting in the system. By neglecting the system trajectory, static optimisation is unable to directly account for temporal features of jumping motions, such as countermovements. While this may prove to be a limitation of the method, the benefits and low computational cost of the method may be compensatory in contexts where an optimal trajectory is not the focus. The equations of kinematic models are much faster to compute than dynamics in simulations and optimisations [53]. This means that iterative optimisation methods may also be better suited to kinematic models than dynamic models.

As this is a novel application of a predictive modelling method, this chapter explores the

insights gained into jumping systems using static optimisation for second order kinematic redundancy resolution. The chapter also assesses the method on its predictive power, computational cost, and intellectual investment.

These research outcomes are realised through a case study using a parameterised model of *Numida meleagris* from [6]. To assess the predictive power of the method, the results of the case study are compared with experimentally measured data found in the literature. Computational cost is assessed by timing the method's execution. The intellectual investment is discussed where relevant knowledge and ability requirements become relevant.

4.1 Kinematic Redundancy in Jumping Systems

Articulated models of jumping systems typically comprise more degrees of freedom in the leg than the body has in planar Cartesian space. In a planar model, the body has 3 degrees of freedom: horizontal and vertical position, and orientation (x_b, y_b, ϕ_b) while the leg comprises 4 degrees of freedom. This kinematic redundancy means that the leg is able to produce an arbitrary body pose, velocity, or acceleration with infinitely many different combinations of joint angle positions, velocities or accelerations respectively. In some applications, the orientation of the body is not a concern and so there are only 2 constrained degrees of freedom of the body. This renders a leg with ≥ 3 degrees of freedom kinematically redundant. Note that rotation of the body still occurs in this application, it is just not a consideration when finding the joint accelerations necessary to produce the linear accelerations of the body. In such an application, the third row of the Jacobian, which represents the body's orientation, is ignored and the Jacobian is used as a $2 \times n$ matrix. Kinematic models of jumping systems consider the motion of the body with no regards to the forces or torques acting on the leg. Some studies use kinematic models in conjunction with simple dynamics models in which the body is treated as a point mass and the work done by the leg is assumed to be the work done to the body [34]. This approach may underestimate the energetic cost of a jumping motion. The leg comprises multiple actuators, some of which may do negative work to the system, decelerating segments to control the direction of the body's acceleration, while others

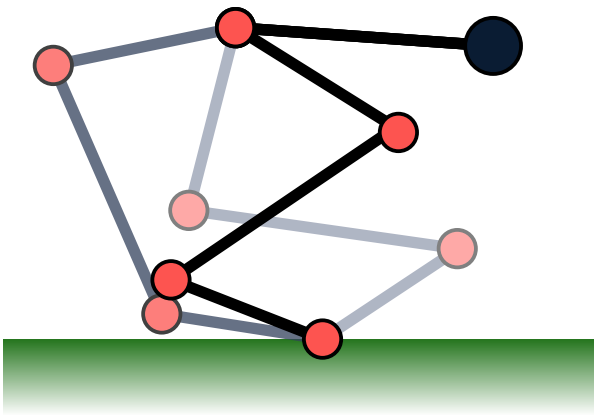


Figure 4.1: Diagram showing various ways in which a redundant, 4 degree of freedom leg can produce the same body pose, (x_b, y_b, ϕ_b). Notice that the final segment of the leg can only be in one position as the orientation of the body is fixed.

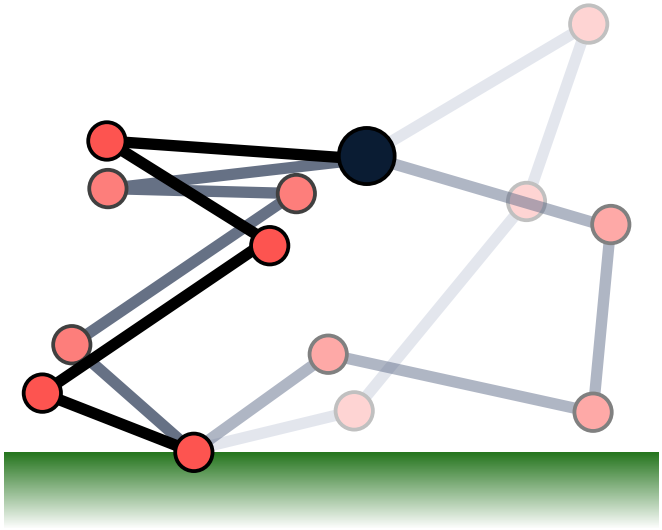


Figure 4.2: Diagram showing various ways in which a redundant, 4 degree of freedom leg can produce the same body position, (x_b, y_b), where orientation is not a constraint. Notice that the final segment of the leg moves freely.

compensate to achieve the desired magnitude of acceleration of the body. This means that there is a difference between the work put into the system by the actuators, W_{in}, and the net work done to the body, W_{out}. This gives an overall efficiency for a given jumping motion:

$$\eta = \frac{W_{out}}{W_{in}} < 1.$$

Note that the work done by the actuators and the work done to the body both depend on the dynamics of the leg and so may vary greatly depending on the nature of the jumping motion produced by the leg. The potential for a difference in the work done by the actuators and the work done to the body implies that the leg dynamics, structure and starting position affect the energetic capability of a jumping system.

4.2 Second Order Kinematic Redundancy Resolution

This chapter considers the use of static optimisation for synthesis of motion data by optimising the joint accelerations based on desired body accelerations. The redundancy of the leg in accelerating the body may be utilised to meet optimisation objectives such as: minimising external torque or ensuring no external torque is applied at certain degrees of freedom. This section presents the derivations and utilities of second order kinematic redundancy resolution using the Moore-Penrose inverse, as found in [21], [51], [54], [71].

The methods described here were proposed in the 1980s for use in manipulator control. The methods were often criticised because they do not posses a cyclic property. For a method to be cyclic, a closed path motion of the body must begin and end with the same overall system configuration. The methods described here do not exhibit this behaviour; the body may return to a starting position with the leg in a different configuration to when the motion started. This is a problem in manipulator control as typical applications involve repetitive motions which may be rendered unpredictable or impractical by these methods. However,

in jumping motions the body does not typically return to the starting position and so cyclic properties of predictive modelling methods are not generally required.

The kinematic acceleration in a jumping trajectory, as defined in equation (1.6) of Section 1.7, is a set of linear equations of the form:

$$\boldsymbol{J}\ddot{\boldsymbol{q}} = (\ddot{\boldsymbol{x}} - \dot{\boldsymbol{J}}\dot{\boldsymbol{q}}) \tag{4.1}$$

where the terms $\boldsymbol{J}$, $\dot{\boldsymbol{J}}$ and $\dot{\boldsymbol{q}}$ depend on the state of the jumping system, $\ddot{\boldsymbol{q}}$ is the vector of joint accelerations to be determined and $\ddot{\boldsymbol{x}}$ is the body acceleration which must be provided prior. In a defined system, where the degrees of freedom of the body and leg are equal, the solution is found by inverting the matrix $\boldsymbol{J}$ as in equation (4.2). However, a redundant system will have a non-square Jacobian matrix and there will be infinitely many valid solutions for $\ddot{\boldsymbol{q}}$.

$$\ddot{\boldsymbol{q}} = \boldsymbol{J}^{-1}(\ddot{\boldsymbol{x}} - \dot{\boldsymbol{J}}\dot{\boldsymbol{q}}) \tag{4.2}$$

The Moore-Penrose inverse may be used to invert the non-square matrix, $\boldsymbol{J}$, and obtain a least squares solution to the redundant acceleration problem. The null space of $\boldsymbol{J}$ is then used to adjust the initial result to meet an additional optimisation objective or to impose constraints on the system, such as ensuring no external torque is applied at an unactuated joint.

4.2.1 Moore-Penrose and Least Squares Optimisation

This section presents the Moore-Penrose inverse and its use in providing a least squares optimisation solution to the inverse kinematics [21]. The Moore-Penrose, or "pseudo-", inverse is used in this work to obtain a least squares solution to the second order, redundant kinematics of a jumping system. In this work the pseudo-inverse is denoted with a superscript $+$ e.g. $\boldsymbol{A}^{+}$, although in some literature a superscript † is used e.g. $\boldsymbol{A}^{\dagger}$. Given that $\boldsymbol{A}$ is an $m \times n$ matrix, then $\boldsymbol{A}^{+}$ is an $n \times m$ matrix. Thus, the nominal solution for $\ddot{\boldsymbol{q}}$ becomes:

$$\ddot{\boldsymbol{q}} = \boldsymbol{J}^{+}(\ddot{\boldsymbol{x}} - \dot{\boldsymbol{J}}\dot{\boldsymbol{q}}). \tag{4.3}$$

The Jacobian for a deficient system, which are not considered in this work, has more rows than columns, $m > n$, and the pseudo-inverse is defined as:

$$\boldsymbol{J}^{+} = (\boldsymbol{J}^{T}\boldsymbol{J})^{-1}\boldsymbol{J}^{T}. \tag{4.4}$$

For a deficient system, the pseudo-inverse is a left inverse, meaning that

$$\boldsymbol{J}^{+}\boldsymbol{J} = \boldsymbol{I}_{n}$$

where $\boldsymbol{I}_{n}$ is the $n \times n$ identity matrix.

The Jacobian of a redundant system, which are considered in this work, has more columns

than rows, $n > m$, and the pseudo-inverse of the Jacobian is defined as:

$$\boldsymbol{J}^{+} = \boldsymbol{J}^{T}(\boldsymbol{J}\boldsymbol{J}^{T})^{-1}. \tag{4.5}$$

For a redundant system, the pseudo-inverse is a right inverse, meaning that

$$\boldsymbol{J}\boldsymbol{J}^{+} = \boldsymbol{I}_{m}$$

where $\boldsymbol{I}_{m}$ is the $m \times m$ identity matrix.

The pseudo-inverse is used to obtain the least squares solution to equation (4.2) as:

$$\ddot{\boldsymbol{q}} = \boldsymbol{J}^{+}(\ddot{\boldsymbol{x}} - \dot{\boldsymbol{J}}\dot{\boldsymbol{q}}). \tag{4.6}$$

The pseudo-inverse has the convenient property that the solution it provides has the minimum Euclidean length. I.e.

$$\begin{aligned} &\min_{\ddot{\boldsymbol{q}}} && \sqrt{\ddot{q}_1^2 + \ddot{q}_2^2 + \ldots + \ddot{q}_n^2} \\ &\text{subject to} && \boldsymbol{J}\ddot{\boldsymbol{q}} + \dot{\boldsymbol{J}}\dot{\boldsymbol{q}} = \ddot{\boldsymbol{x}} \end{aligned}$$

This minimisation is convenient for producing smooth motions with minimal overall accelerations. However, it is not guaranteed to provide the minimum torque solution. To find the minimum torque solution, the null space of the Jacobian is used as demonstrated by Hollerbach and Suh [54] and is described in this thesis in Sections 4.2.3 and 4.2.4.

4.2.2 Weighted Pseudo-Inverse

Whitney [72] used a weighted variation of the Jacobian matrix $\boldsymbol{J}_{w}$ for the optimisation to encourage minimisation of specific joint accelerations, or combinations of joint accelerations:

$$\boldsymbol{J}_{w} = \boldsymbol{W}^{-1}\boldsymbol{J}^{T}(\boldsymbol{J}\boldsymbol{W}^{-1}\boldsymbol{J}^{T})^{-1}. \tag{4.7}$$

Using the weighted Jacobian in place of the standard Jacobian in equation (4.3), the pseudo-inverse will minimise the weighted criteria:

$$\min_{\ddot{\boldsymbol{q}}} \ \ddot{\boldsymbol{q}}^{T}\boldsymbol{W}\ddot{\boldsymbol{q}}$$

In first order kinematics, this method is typically used by setting the weighting matrix $\boldsymbol{W}$ to be the inertia matrix $\boldsymbol{H}$, thus minimising the instantaneous kinetic energy of the system [72]:

$$\min_{\dot{\boldsymbol{q}}} \ \dot{\boldsymbol{q}}^{T}\boldsymbol{H}\dot{\boldsymbol{q}}$$

The inertia weighted Jacobian is applied here, based on the work done by Khatib [51], to the

second order kinematics to minimise the instantaneous value of:

$$\min_{\ddot{\boldsymbol{q}}} \ \ddot{\boldsymbol{q}}^T \boldsymbol{H} \ddot{\boldsymbol{q}}$$

4.2.3 The Null Space of the Jacobian

The null space of the Jacobian is the set of joint acceleration vectors which would have no effect on the body's acceleration. Such a vector can be used to alter another joint acceleration vector which produces a desirable body acceleration to have the leg perform "self motion" to achieve an additional objective. For example, Hollerbach and Suh use the null space of the Jacobian to minimise the torque required to produce the body acceleration [54]. This section presents the null space of the Jacobian matrix and explains how the null space is used to adjust joint acceleration solutions found using the pseudo-inverse.

Consider the $m \times n$ Jacobian matrix, $\boldsymbol{J}$. The null space of $\boldsymbol{J}$, denoted $\mathrm{N}(\boldsymbol{J})$, is the set of vectors whose product with $\boldsymbol{J}$ result in a zero vector:

$$\mathrm{N}(\boldsymbol{J}) = \{\boldsymbol{x} \in \mathbb{R}^n \mid \boldsymbol{J}\boldsymbol{x} = \boldsymbol{0}\}$$

A vector in the null space of $\boldsymbol{J}$ may be added to another vector, which is not in the null space, with no impact on the vector's product with $\boldsymbol{J}$. For example, consider a vector, $\boldsymbol{b}$, which is not a member of the null space of $\boldsymbol{J}$, and a second vector, $\boldsymbol{c}$, which is a member of the null space of $\boldsymbol{J}$, thus $\boldsymbol{J}\boldsymbol{c} = \boldsymbol{0}$. Then the products $\boldsymbol{J}(\boldsymbol{b})$ and $\boldsymbol{J}(\boldsymbol{b} + \boldsymbol{c})$ are equivalent:

$$\boldsymbol{J}(\boldsymbol{b} + \boldsymbol{c}) = \boldsymbol{J}\boldsymbol{b} + \boldsymbol{J}\boldsymbol{c} = \boldsymbol{J}\boldsymbol{b} + \boldsymbol{0} = \boldsymbol{J}\boldsymbol{b}$$

An arbitrary vector, $\ddot{\boldsymbol{\psi}}$, may be projected into the null space of the Jacobian using:

$$\mathrm{proj}_{\mathrm{N}(\boldsymbol{J})} \ddot{\boldsymbol{\psi}} = (\boldsymbol{I} - \boldsymbol{J}^{+}\boldsymbol{J})\ddot{\boldsymbol{\psi}}. \tag{4.8}$$

Where $\boldsymbol{I}$ is the identity matrix. Thus, the set of all joint accelerations which produce a given body acceleration, $\ddot{\boldsymbol{x}}$, are defined by:

$$\ddot{\boldsymbol{q}} = \boldsymbol{J}^{+}(\ddot{\boldsymbol{x}} - \dot{\boldsymbol{J}}\dot{\boldsymbol{q}}) + (\boldsymbol{I} - \boldsymbol{J}^{+}\boldsymbol{J})\ddot{\boldsymbol{\psi}} \tag{4.9}$$

Where the vector $\ddot{\boldsymbol{\psi}} \in \mathbb{R}^n$ may be selected to alter a nominal joint acceleration vector to have the leg meet another criteria, such as minimising torque or setting the acceleration/torque of a chosen joint to zero.

4.2.4 Minimum Torque Optimisation

Hollerbach and Suh [54] demonstrate the use of static optimisation to find minimum torque solutions for redundant manipulators. Their method uses the null space of the Jacobian matrix

to adjust the joint accelerations to find the minimum torque solution while meeting a target acceleration of the end effector of a manipulator, which is equivalent to the body of a jumping system. The following is a walkthrough of the derivation performed by Hollerbach and Suh [54]. The dynamics of the system are defined in Section 1.8 with their general form given in equation (1.8a):

$$\boldsymbol{\tau} = \boldsymbol{H}\ddot{\boldsymbol{q}} + \boldsymbol{C}\dot{\boldsymbol{q}} + \boldsymbol{\tau}_g.$$

As the equations in this section become quite large, the condensed form of the dynamics is used:

$$\boldsymbol{\tau} = \boldsymbol{H}\ddot{\boldsymbol{q}} + \boldsymbol{b} \tag{4.10}$$

Substituting equation (4.9) into (4.10) gives the externally applied torques based on the desired body acceleration, the state of the system and the choice of $\ddot{\boldsymbol{\psi}}$.

$$\boldsymbol{\tau} = \boldsymbol{H}\boldsymbol{J}^{+}(\ddot{\boldsymbol{x}} - \dot{\boldsymbol{J}}\dot{\boldsymbol{q}}) + \boldsymbol{H}(\boldsymbol{I} - \boldsymbol{J}^{+}\boldsymbol{J})\ddot{\boldsymbol{\psi}} + \boldsymbol{b} \tag{4.11}$$

The minimum torque solution would be $\boldsymbol{\tau} = \boldsymbol{0}$ which would require:

$$\boldsymbol{H}(\boldsymbol{I} - \boldsymbol{J}^{+}\boldsymbol{J})\ddot{\boldsymbol{\psi}} = -\boldsymbol{H}\boldsymbol{J}^{+}(\ddot{\boldsymbol{x}} - \dot{\boldsymbol{J}}\dot{\boldsymbol{q}}) - \boldsymbol{b}.$$

The pseudo-inverse is then used to find the minimum norm solution to the zero torque case:

$$\ddot{\boldsymbol{\psi}} = \Big[\boldsymbol{H}(\boldsymbol{I} - \boldsymbol{J}^{+}\boldsymbol{J})\Big]^{+}\Big(-\boldsymbol{H}\boldsymbol{J}^{+}(\ddot{\boldsymbol{x}} - \dot{\boldsymbol{J}}\dot{\boldsymbol{q}}) - \boldsymbol{b}\Big). \tag{4.12}$$

$\ddot{\boldsymbol{\psi}}$ may then be projected into the null space of the Jacobian and added to the nominal solution for $\ddot{\boldsymbol{q}}$ as in equation (4.9):

$$\ddot{\boldsymbol{q}} = \boldsymbol{J}^{+}(\ddot{\boldsymbol{x}} - \dot{\boldsymbol{J}}\dot{\boldsymbol{q}}) + (\boldsymbol{I} - \boldsymbol{J}^{+}\boldsymbol{J})\Big[\boldsymbol{H}(\boldsymbol{I} - \boldsymbol{J}^{+}\boldsymbol{J})\Big]^{+}\Big(-\boldsymbol{H}\boldsymbol{J}^{+}(\ddot{\boldsymbol{x}} - \dot{\boldsymbol{J}}\dot{\boldsymbol{q}}) - \boldsymbol{b}\Big) \tag{4.13}$$

The solution of equation (4.2.4) is the joint angular accelerations having minimum norm torques which also produce the desired body acceleration vector, $\ddot{\boldsymbol{x}}$.

4.2.5 Enforcing Unactuated Degrees of Freedom

This section presents a novel derivation in which the null space of the Jacobian may be used to enforce zero acceleration constraints at a given degree of freedom. During the preparation stage and the initial part of the push-off stage of a jump, the foot will remain flat on the ground.

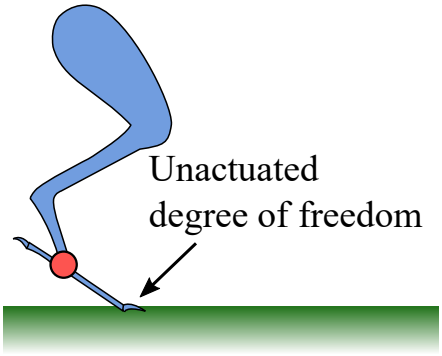


Figure 4.3: Diagram showing the existence of an unactuated degree of freedom at the end of the foot during a jumping motion

However, in the later parts of the push-off stage, the foot may rotate up onto the tip of the toe. Assuming that the foot is rigid and does not bend or flex to accommodate this motion, a model of the system will gain an extra rotational degree of freedom at the tip of the toe when the heel breaks contact with the ground. The extra degree of freedom is generally considered to be unactuated and is modelled using a revolute joint. This event is demonstrated in figure 4.3, in which a 2 degree of freedom system transitions into a 3 degree of freedom system by rotating the foot. There are no connections between the toe's tip and the ground and so the system cannot apply a torque to the degree of freedom as it can to the other degrees of freedom. Instead, the system must induce acceleration indirectly by controlling the other degrees of freedom.

The ability to generate motions in this way may also be beneficial in finding motions for systems which are unable to actuate certain joints. This may be for a failure case in a robotic system or for a person having an injury. The use of kinematics to simulate models having unactuated joints is not appropriate as a kinematic solution may require an externally applied torque at the unactauted degree of freedom. This is an issue with using models which do not consider dynamics. To accommodate the unactuated degree of freedom, $\ddot{\boldsymbol{\psi}}$ may be selected to ensure that the joint angular accelerations require an external torque of zero at the unactuated degree of freedom.

Consider the condensed form of the dynamics:

$$\boldsymbol{\tau} = \boldsymbol{H}\ddot{\boldsymbol{q}} + \boldsymbol{b}$$

where $\boldsymbol{b}$ is the sum of the non-acceleration related terms:

$$\boldsymbol{b} = \boldsymbol{C}\dot{\boldsymbol{q}} + \boldsymbol{\tau}_g.$$

The torque at the i^{th} degree of freedom is given by:

$$\tau_i = \boldsymbol{H}_{i*}\ddot{\boldsymbol{q}} + b_i \tag{4.14}$$

where the subscript i represents the i^{th} element of a vector and the subscript $i*$ represents the

i^{th} row of a matrix. Equation (4.15) is obtained by substituting equation (4.9) into (4.14).

$$\tau_i = \boldsymbol{H}_{i*}\boldsymbol{J}^{+}(\ddot{\boldsymbol{x}} - \dot{\boldsymbol{J}}\dot{\boldsymbol{q}}) + \boldsymbol{H}_{i*}(\boldsymbol{I} - \boldsymbol{J}^{+}\boldsymbol{J})\ddot{\boldsymbol{\psi}} + b_i \tag{4.15}$$

By setting the external torque at joint i, τ_i, to zero and using the pseudo-inverse, a solution for $\ddot{\boldsymbol{\psi}}$ is obtained.

$$\ddot{\boldsymbol{\psi}} = \left[\boldsymbol{H}_{i*}(\boldsymbol{I} - \boldsymbol{J}^{+}\boldsymbol{J})\right]^{+}\left(-\boldsymbol{H}_{i*}\boldsymbol{J}^{+}(\ddot{\boldsymbol{x}} - \dot{\boldsymbol{J}}\dot{\boldsymbol{q}}) - b_i\right) \tag{4.16}$$

The vector $\ddot{\boldsymbol{\psi}}$ found in equation (4.16) may then be used in equation (4.9) to obtain a set joint accelerations which produce the desired body acceleration, $\ddot{\boldsymbol{x}}$, while ensuring the external torque at joint i is zero:

$$\ddot{\boldsymbol{q}} = \boldsymbol{J}^{+}(\ddot{\boldsymbol{x}} - \dot{\boldsymbol{J}}\dot{\boldsymbol{q}}) + (\boldsymbol{I} - \boldsymbol{J}^{+}\boldsymbol{J})\left[\boldsymbol{H}_{i*}(\boldsymbol{I} - \boldsymbol{J}^{+}\boldsymbol{J})\right]^{+}\left(-\boldsymbol{H}_{i*}\boldsymbol{J}^{+}(\ddot{\boldsymbol{x}} - \dot{\boldsymbol{J}}\dot{\boldsymbol{q}}) - b_i\right) \tag{4.17}$$

4.3 Static Optimisation Method

A case study is presented in which static optimisation is used to resolve the kinematic redundancy of a 4 degree of freedom leg and simulate the leg of *Numida meleagris* during a jumping motion. Motion capture data of *Numida meleagris* jumping motions from [6] were used to provide a reference acceleration trajectory for the body of the system. The simulation results include joint angle trajectories, ground reaction forces and the torques applied at the joints. The joint angle trajectories and ground reaction forces are compared with experimentally measured motion data for a representative jump [6]. The simulated joint angle trajectories and ground reaction forces were then compared with the experimentally measured motion of the bird.

A 4 degree of freedom planar linkage system was used to model *Numida meleagris* with the head, arms and torso treated as a single rigid body as shown in figure 4.4. The total system mass was 1.4 kg and distribution of the mass was approximated according to the muscle masses reported in [6], as shown in table 4.1. The leg segment lengths were taken from [73].

The experimentally measured body accelerations, a_h and a_v, from [6], were digitised and

Table 4.1: Distribution of mass in the body and leg segments as fractions of the total system mass.

Segment	Head Arms Torso	Femur	Tibiotarsus	Tarsometatarsus	Toe Digits
Fraction of system mass	0.75	0.1250	0.0750	0.0250	0.0250

modelled using 4^{th} order time polynomials. 4^{th} order was sufficient to capture the general changes in acceleration without over fitting to the experimentally measured data which may be subject to noise and errors. The angular acceleration of the body was neglected as no data was available. The polynomials provided the linear accelerations of the body as continuous

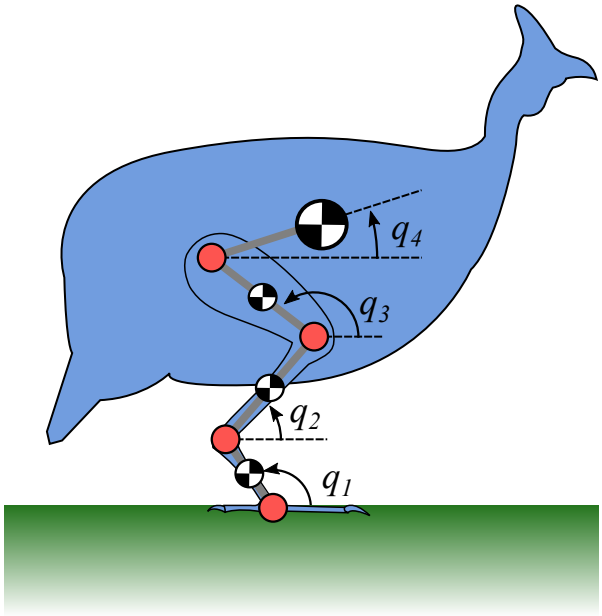


Figure 4.4: The 4 degree of freedom model used to represent *Numida meleagris*. Each segment is treated as a rigid rod with mass and inertia. Segment inertias are approximated by $\frac{1}{12}ml^2$, where m and l are the segment mass and length respectively. The inertia of the body was approximated by treating the body as a solid sphere of radius 0.2 m.

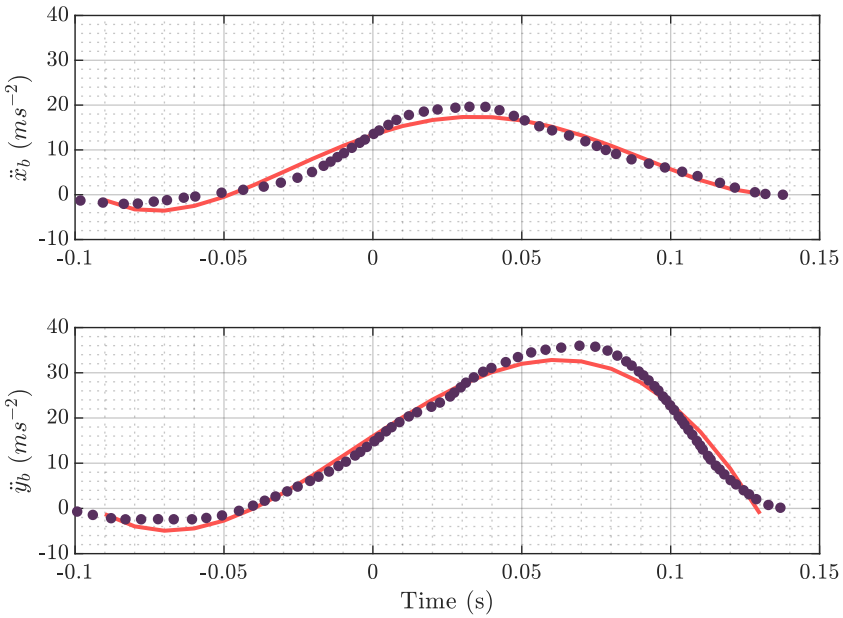


Figure 4.5: 4^{th} order polynomial fit to the body accelerations measured by Henry et al. [6].

functions of time which were used in kinematic simulation of the system:

$$a_h(t) = b_1 + b_2t + b_3t^2 + b_4t^3 + b_5t^4 \tag{4.18}$$

$$a_v(t) = c_1 + c_2t + c_3t^2 + c_4t^3 + c_5t^4 \tag{4.19}$$

The joint angular accelerations at time t were obtained using three variations of static optimisation as described in the previous sections. The objectives of optimisation were as follows.

Minimum Norm Acceleration

The instantaneous minimum norm of the joint accelerations was obtained using the pseudo-inverse according to equation (4.6):

$$\ddot{\boldsymbol{q}} = \boldsymbol{J}^{+}(\boldsymbol{a} - \dot{\boldsymbol{J}}\dot{\boldsymbol{q}}) \tag{4.20}$$

where the target acceleration of the body, $\boldsymbol{a}$ is

$$\boldsymbol{a} = \begin{bmatrix} a_h(t) \\ a_v(t) \end{bmatrix}.$$

The time dependencies of the terms in the above equations are dropped for clarity.

Inertia Weighted Pseudo-Inverse

The inertia weighted Jacobian, $\boldsymbol{J}_h$, is derived by Nakamura et al. [74] by substituting the inertia matrix $\boldsymbol{H}$ into equation (4.7):

$$\boldsymbol{J}_h = \boldsymbol{H}^{-1}\boldsymbol{J}^{T}(\boldsymbol{J}\boldsymbol{H}^{-1}\boldsymbol{J}^{T})^{-1}. \tag{4.21}$$

Replacing the Jacobian, $\boldsymbol{J}$, in equation (4.20) with the inertia weighted Jacobian provides a weighted acceleration minimisation objective:

$$\ddot{\boldsymbol{q}} = \boldsymbol{J}_h^{+}(\boldsymbol{a} - \dot{\boldsymbol{J}}\dot{\boldsymbol{q}}) \tag{4.22}$$

Minimum Norm Torque

The instantaneous minimum norm of the applied torques was obtained using equation (4.2.4).

The kinematic model was simulated using the variable step integration scheme "ode45" in MatLab. As the orientation of the body was not considered, the Jacobian matrix used in the optimisations consisted of the first 2 rows of the system's complete Jacobian, as did the time rate of change of the Jacobian, $\dot{\boldsymbol{J}}$:

$$\boldsymbol{J} = \begin{bmatrix} -l_1\sin(q_1) & -l_2\sin(q_2) & -l_3\sin(q_3) & -l_4\sin(q_4) \\ l_1\cos(q_1) & l_2\cos(q_2) & l_3\cos(q_3) & l_4\cos(q_4) \end{bmatrix}$$

$$\dot{\boldsymbol{J}} = \begin{bmatrix} -l_1\dot{q_1}\cos(q_1) & -l_2\dot{q_2}\cos(q_2) & -l_3\dot{q_3}\cos(q_3) & -l_4\dot{q_4}\cos(q_4) \\ -l_1\dot{q_1}\sin(q_1) & -l_2\dot{q_2}\sin(q_2) & -l_3\dot{q_3}\sin(q_3) & -l_4\dot{q_4}\sin(q_4) \end{bmatrix}.$$

4.3.1 Determination of Heel-off Event

In this work, joint angles were defined relative to the ground, whereas the joint angle trajectories reported by Henry et al. were defined relative to one of the front digits of the foot. This

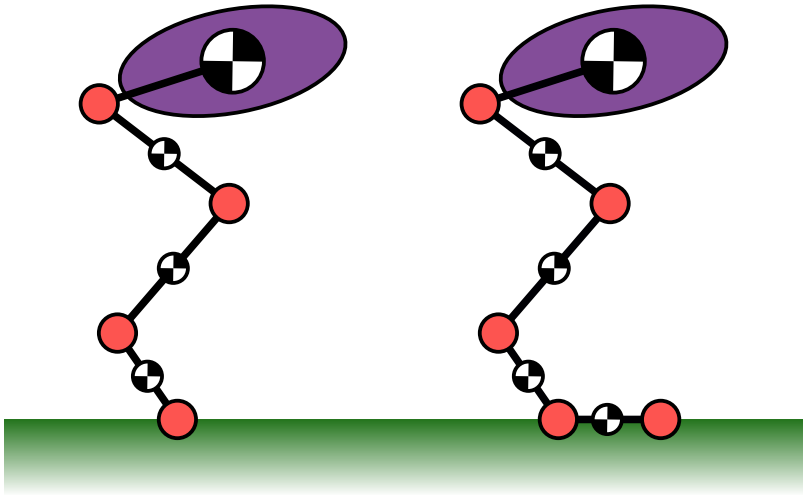


Figure 4.6: Diagrams of the 4 degree of freedom, without foot model (left) and the 5 degree of freedom, with foot model (right). The additional joint at the end of the foot is indexed with 0 (e.g. q_0) to keep the joint indices consistent between the two models.

means that towards the end of the jump, when the rear digit of the foot breaks contact with the ground, the angles are not sufficient to describe the system's position relative to the ground and they are no longer useful for comparison. The point at which the rear digits break contact with the ground is not specified in Henry et al.'s work. An approximate time of the event was determined using a hybrid modelling approach.

To determine the motion of the foot during the jumping motion and the point in time at which the heel of the foot lifts off the ground, two models were used: one to simulate the motion of the leg and another to determine the motion of the foot given the motion of the leg. The two models are shown in figure 4.6.

A 4 degree of freedom, "without foot" model was used in the optimisations to simulate the kinematic motion of the system throughout the jumping motion. This model treated the leg as being joined directly to the ground. The without foot model accounted for the degrees of freedom in the system at which the bird was able to produce external torques using its muscles.

The second model, "with foot" represented by subscript f, included an extra degree of freedom at the end of the foot which the bird was not able to directly actuate as no muscle crosses from the tip of the toe to the ground. The bird is able to influence acceleration at the degree of freedom, but not apply an external torque to the degree of freedom. The with foot model was used to verify that the motion data produced by the without foot model was physically possible. Given the assumption that the bird had no means of gripping and pulling itself towards the ground, the ground reaction force must be $F_v >= 0$ throughout the jumping motion. With the foot stationary and flat on the ground, i.e. $q_0 = \pi$, $\dot{q}_0 = 0$ and $\ddot{q}_0 = 0$, the with foot model was used to verify that the required ground reaction force was positive and the motion was possible without requiring the foot to pull on the ground. This was done using inverse dynamics of the model to determine the accelerations of each segment as in equation (1.8a).

By definition of the model, the only means of an external torque being applied at joint 0 is

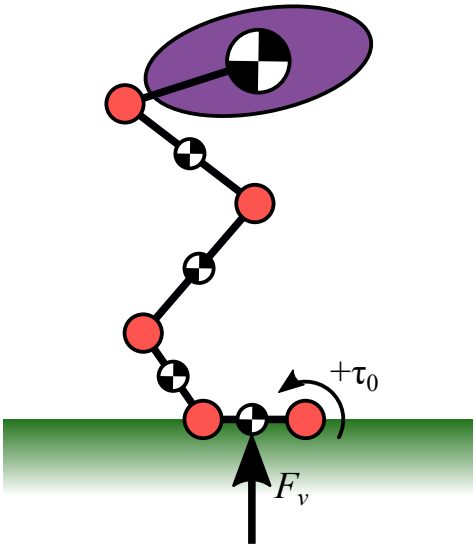


Figure 4.7: Diagram showing the ground reaction force and the external torque, τ_0, acting at the end of the foot. Positive moments are defined as acting anti-clockwise, thus a positive value of F_v which acts to the left hand side of joint 0 will produce a negative torque at joint 0.

through the ground reaction force acting on the foot and producing a moment about the joint. As joint 0 is at the end of the foot, and the ground reaction force can only act in the positive (upwards) direction, the external torque acting at the toe joint must be

$$\tau_0 <= 0,$$

as demonstrated in figure 4.7. When τ_0 becomes greater than zero, the system can no longer be modelled using the without foot model and so simulation must be performed using the with foot model. This requires the use of equation (4.17), or similar, to ensure the motion proceeds with $\tau_0 = 0$. However, in this work the simulation was stopped after detection of the heel off event.

$$\boldsymbol{q}_f = \begin{bmatrix} \pi \\ \boldsymbol{q} \end{bmatrix}, \quad \dot{\boldsymbol{q}}_f = \begin{bmatrix} 0 \\ \dot{\boldsymbol{q}} \end{bmatrix}, \quad \ddot{\boldsymbol{q}}_f = \begin{bmatrix} 0 \\ \ddot{\boldsymbol{q}} \end{bmatrix}.$$

4.4 Results

This section presents the results of static optimisation for resolution of kinematic redundancy with the three objectives described previously. Diagrams showing the pose of the leg at 5 points in time throughout the resulting trajectories are presented. The joint angle trajectory and ground reaction force results are presented alongside the experimentally measured data from [6]. Each set of results is grouped by the objective used to produce them.

The static optimisations for each objective were timed over 5 runs each. The average run time for each objective are presented in the following table. Note that this time does not include the derivation time for the model as it is not necessary to the method.

Optimisation Objective	Average Compute Time (s)
Minimum Norm Acceleration	0.019
Minimum Norm Inertia Weighted Acceleration	0.049
Minimum Norm Torque	0.536

Table 4.2: The average time taken to integrate the jumping motion trajectory using static optimisation to determine the joint acceleration vector from a prescribed body acceleration while meeting the objectives shown.

4.4.1 Minimum Norm Acceleration

The following results are the time independent optimisation solutions for $\ddot{\boldsymbol{q}}$ which minimise $\|\ddot{\boldsymbol{q}}\|$.

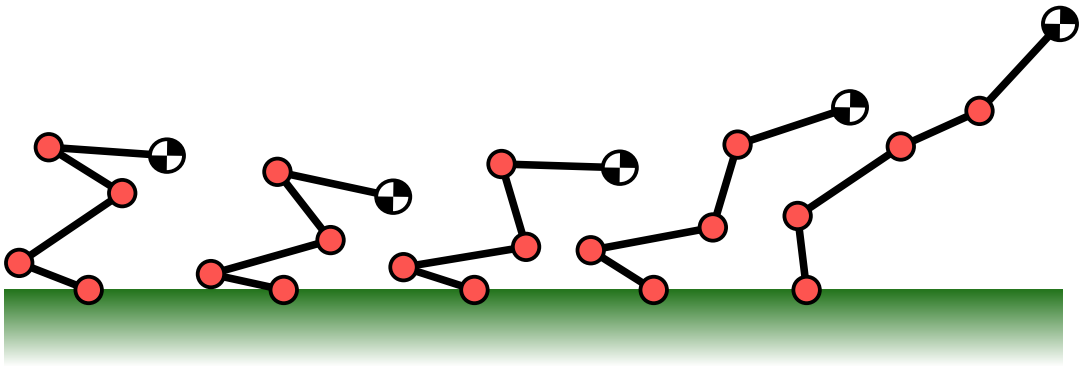


Figure 4.8: Poses of *Numida meleagris* for the minimum norm acceleration motion. Segment inertias are not labelled for clarity.

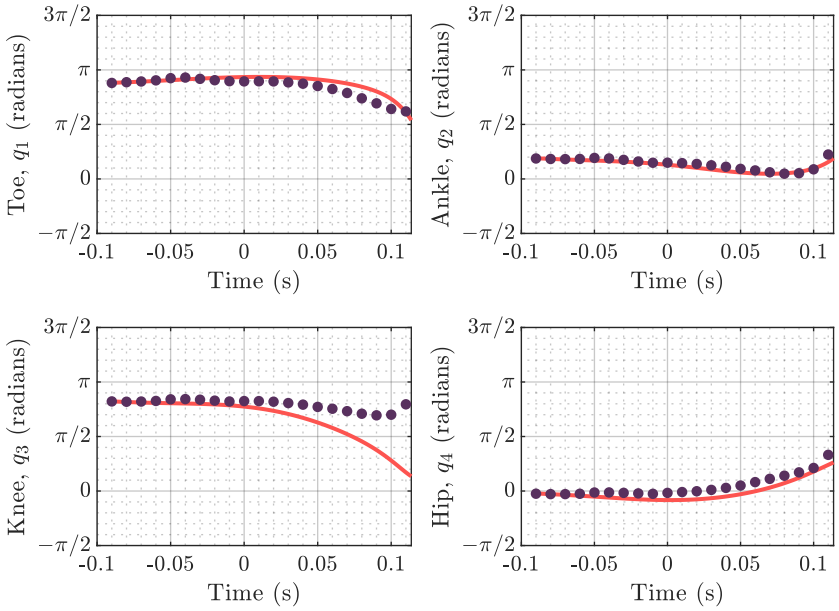


Figure 4.9: Minimum norm acceleration pseudo-inverse joint angle trajectories. With experimentally measured angles of *Numida meleagris* jumps from [6]. The toe, ankle and hip joints follow the general trajectory as the experimentally measured data whereas the knee joint deviates from the experimental results, reaching a difference of 2 radians at the end of the trajectory.

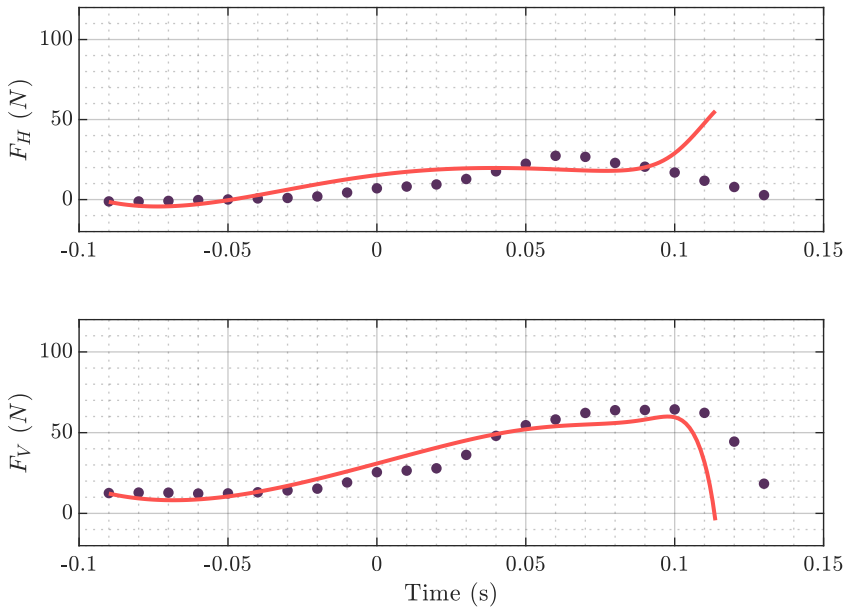


Figure 4.10: Ground reaction forces of the motion produced by static optimisation, compared with the force plate measurements presented by Henry et al. Both reaction forces change rapidly in the optimisation results due to the leg reaching near full extension. This is close to the point at which the heel of the foot would lift off the ground in the actual jumping motion.

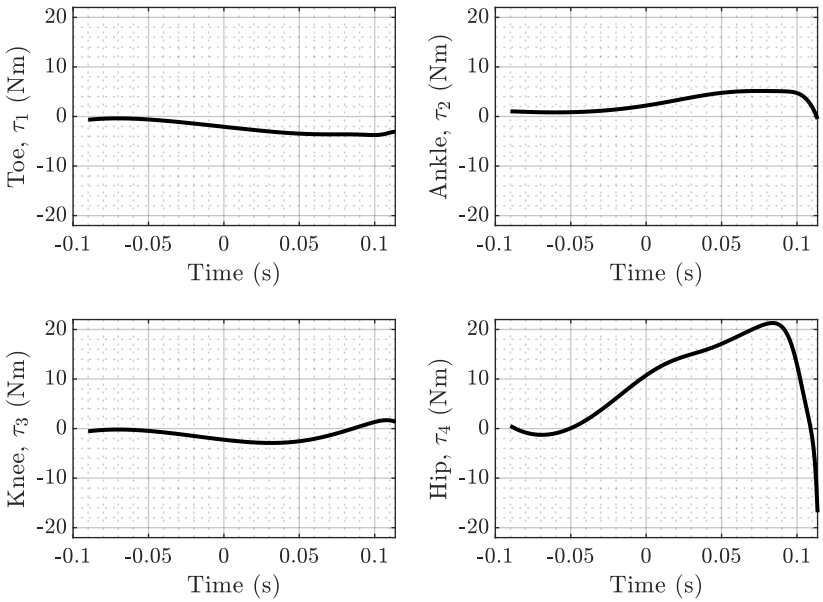


Figure 4.11: Joint torques produced by static optimisation. No comparable data was measured by Henry et al. The torque applied at the hip is significantly greater than the torques at other joints. The sharp changes at the end of the trajectory for all torques are due to the leg reaching near full extension, requiring relatively greater torque magnitudes for a given body acceleration.

4.4.2 Minimum Norm Inertia Weighted Acceleration

The following results are the time independent optimisation solutions for $\ddot{\boldsymbol{q}}$ which minimise the instantaneous value of $\ddot{\boldsymbol{q}}^T \boldsymbol{H} \ddot{\boldsymbol{q}}$.

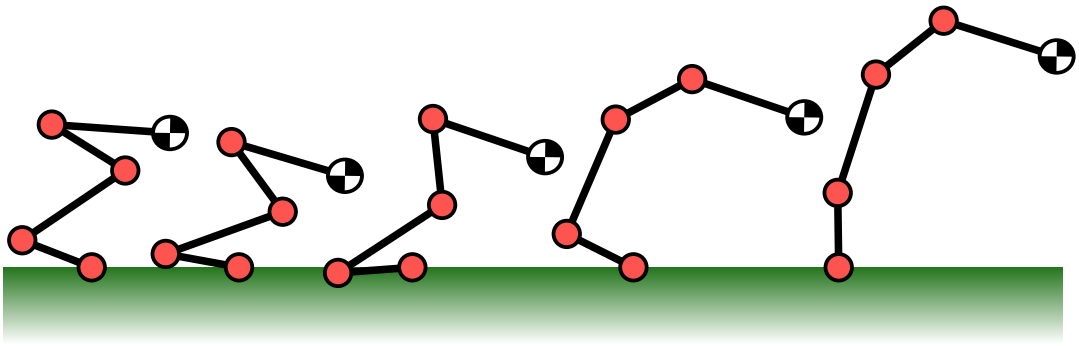


Figure 4.12: Poses of *Numida meleagris* for the minimum norm weighted acceleration motion. The tarsometatarsus and tibiotarsus segments penetrate the ground during the motion.

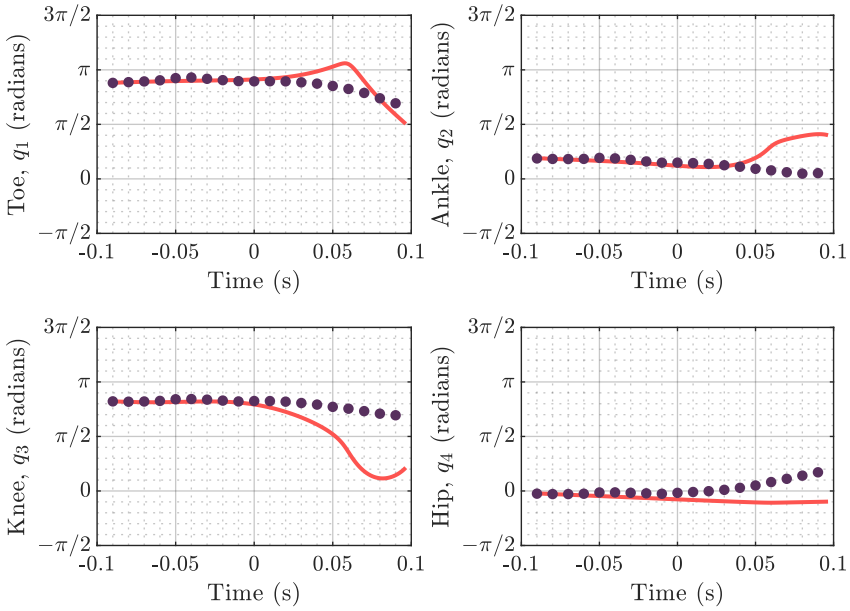


Figure 4.13: Inertia weighted pseudo-inverse joint angle trajectories. With experimentally measured angles of *Numida meleagris* jumps from [6]. The body centre of mass remains below the hip joint throughout the motion.

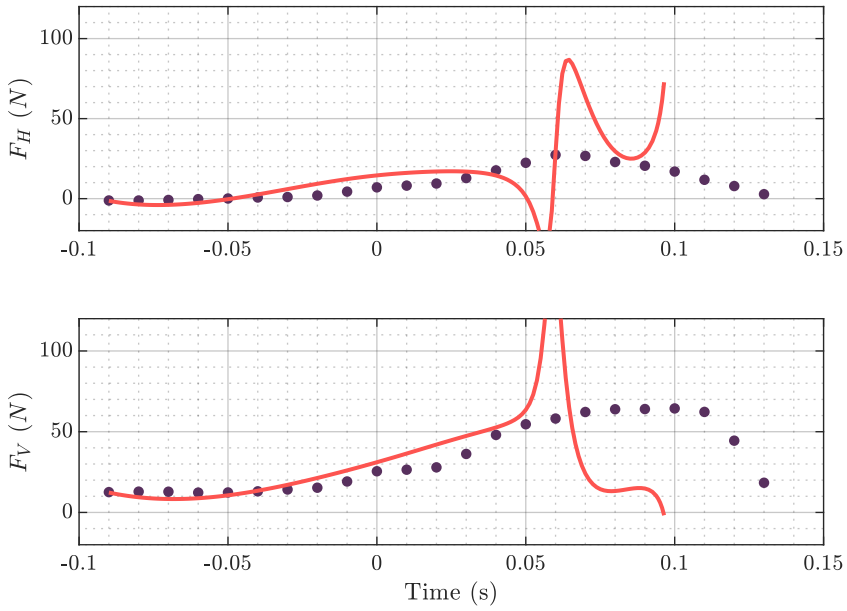


Figure 4.14: Ground reaction forces of the motion produced by static optimisation, compared with the force plate measurements presented by Henry et al. The sharp changes in reaction force correspond to the point in the trajectory when the toe joint angular velocity changes direction.

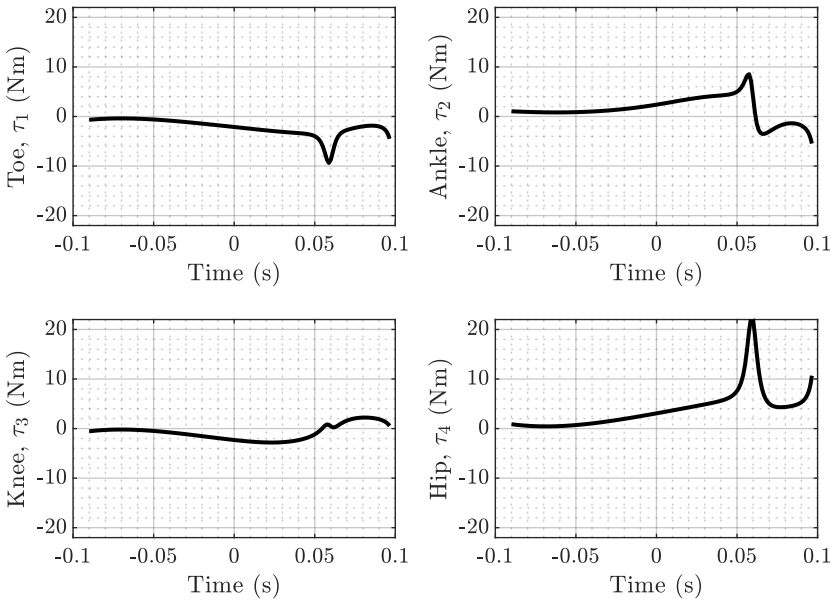


Figure 4.15: Joint torques produced by static optimisation. No comparable data was measured by Henry et al. The magnitudes of the torques are similar for much of the motion. Large changes in torque correspond to the point in the trajectory when the toe joint angular velocity changes direction.

4.4.3 Minimum Norm Torque

The following results are the time independent optimisation solutions for $\ddot{\boldsymbol{q}}$ which minimise $\|\boldsymbol{\tau}\|$ using the null space of $\boldsymbol{J}$. The motion of the system following the minimum torque objective did not produce a feasible motion; the vertical ground reaction force frequently fell below zero. To circumvent this, the ground reaction force was ignored during simulation. A significant change in the vertical ground reaction force is seen at 0.083 s, this time is assumed to coincide with the take-off of the jumping motion.

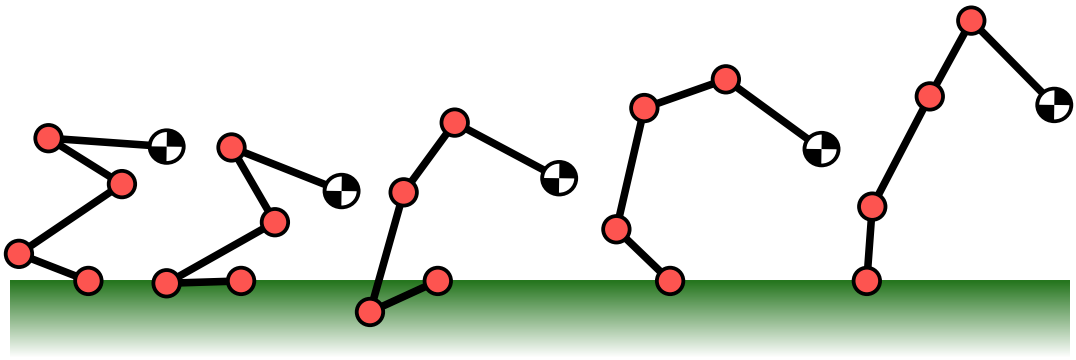


Figure 4.16: Poses of *Numida meleagris* for the minimum torque motion. The tarsometatarsus and tibiotarsus penetrate the ground. The knee is fully extended at the point of take-off. The hip joint remains above the centre of mass of the body throughout the motion.

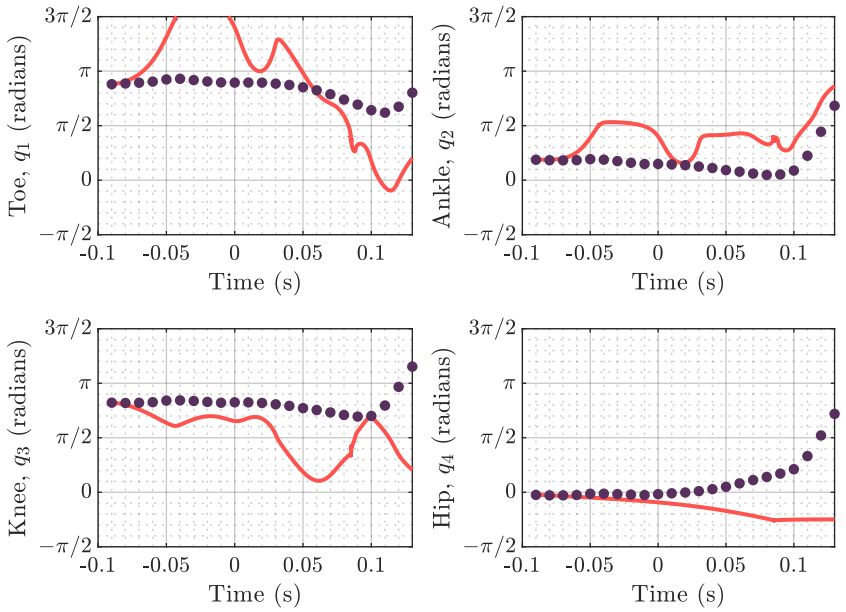


Figure 4.17: Minimum norm torque joint angle trajectories. With experimentally measured angles of *Numida meleagris* jumps from [6]. These results demonstrate significant variation in the joint angles.

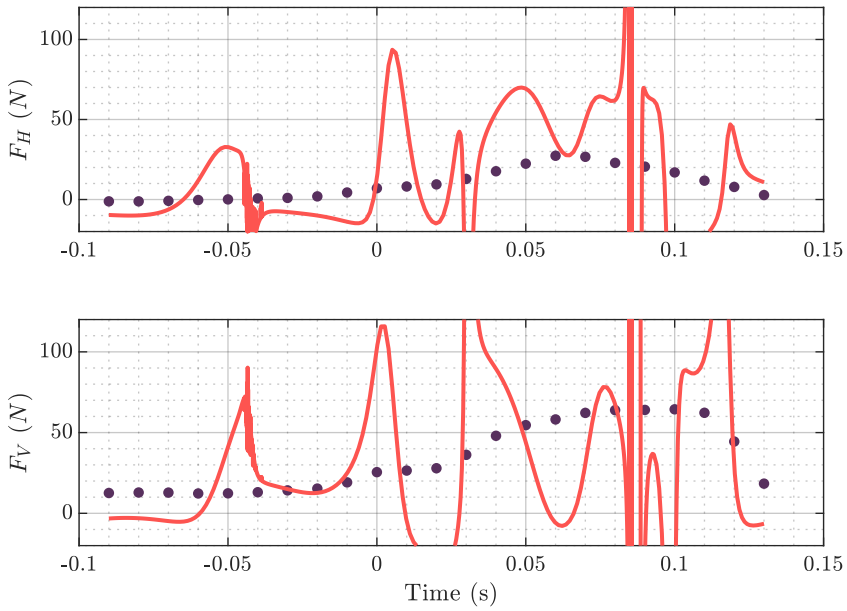


Figure 4.18: Ground reaction forces of the static optimisation results, compared with the force plate measurements presented by Henry et al. Both the vertical and horizontal reaction forces vary considerably throughout the motion. Both forces invert multiple times in quick succession at 0.083 s, this is taken as the point of take-off for the trajectory. Previous points at which F_v becomes negative are neglected as they may be related to stutter-jumping.

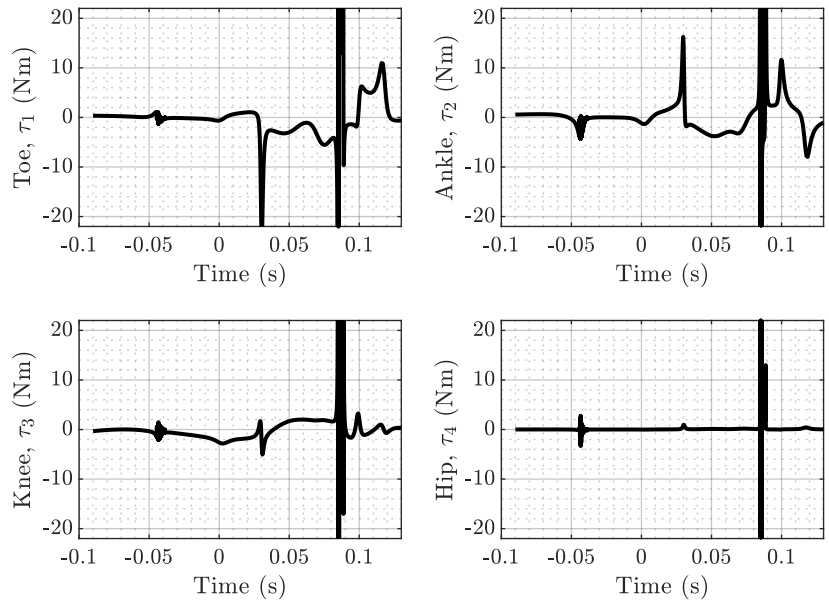


Figure 4.19: Joint torques produced by static optimisation. No comparable data was measured by Henry et al. For periods of the motion the torques are small in magnitude, however, points at which the joint angular velocities change result in spikes in torque magnitudes.

4.5 Discussion

The implementations of static optimisation in this work showed computation times ranging from approximately 20 ms to 500 ms. This is a significantly small computation time for synthesis of jumping motion data. With such a small computational cost, static optimisation would be an ideal candidate method for the early prototyping stages of research or design.

Static optimisation was applied to a high complexity model. The method is applicable to low complexity and single degree of freedom models as well. However, the null space of the Jacobian matrix is only relevant when the system is redundant. Low complexity and single degree of freedom models are likely to be deficient, in which case static optimisation would be used to find motions which match the prescribed body acceleration as closely as possible based on the least squares metric. As such, static optimisation may be considered a moderately rigid method due to the fact that it is not universally applicable to models of jumping systems.

The minimum norm acceleration solution of the static optimisation produced a jumping motion which was close to the experimentally measured motion data found in the literature. The toe, ankle, and hip joints followed the general paths of the experimentally measured data while the knee joint began to deviate in the final quarter of the trajectory. This shows promise for the method's application to jumping problems. However, the minimum norm results also included an inversion of the hip joint near the end of the jumping motion. While this may be attributed to flex in the spine of a real bird, it demonstrates an issue with the method which was seen in all sets of results in this work: physical constraints cannot be applied to the optimisation.

The other variations of static optimisation in this work (minimum norm inertia weighted acceleration and minimum norm torque) produced unrealistic motions. The results deviated from the experimentally measured data significantly. The minimum norm torque method performed especially poorly, producing large oscillations in the joint angle trajectories, ground reaction forces, and externally applied torques. The two variations also produced motions which pushed the tarsometatarsus and tibiotarsus segments through the ground, which is infeasible for jumping motions.

As the results were demonstrated to be generally infeasible for a biological jumping system to produce, static optimisation may be considered as unsuitable for gaining insights into the motions of jumping systems at this level of implementation. However, it may be possible to enforce constraints in a similar approach to the use of degree of freedom constraints in equation (4.16). Another potential work around would be to use a weighting matrix, as in equation (4.7), to penalise the accelerations of joints which are found to violate physical constraints. By penalising the accelerations of problem joints, the displacement of those joints should also be reduced. This may be enough to keep the joint angles within the bounds of the physical constraints.

A similar kinematic method, proposed by Richards et al. [55], which is based on quaternion

interpolation, is unlikely to violate joint angle constraints as the method uses the joint angles as input for both the start and end configurations. Should static optimisation prove to be problematic in regard to physical constraints, the method proposed by Richards et al. may be a suitable alternative.

For all implementations of the static optimisation method in this chapter, the starting configuration of the system and trajectory for the acceleration of the body were required as inputs. These factors are likely to have implications on the results produced by the method and would be worth investigating in future studies.

While this chapter utilised the dynamics of the system, including mass and inertia approximations for the segments in the model, this was not necessary to carry out the static optimisations. The dynamics were included in this study to aid in the validation of the results against data from the literature [6].

4.6 Conclusions

The results in Figures 4.9, 4.10 and 4.11 are comparable to the experimentally measure data for the jumping system. This demonstrates the potential for static optimisation to be used as a predictive modelling method for jumping problems.

The results in Sections 4.4.2 and 4.4.3 imply that the use of additional objectives in static optimisation may be inappropriate for the synthesis of jumping motions.

All of the results in this chapter violated physical constraints of the biological system which was modelled. Further work would be required to explore applications of static optimisation which can accommodate the physical constraints of jumping systems, either through inclusion of constraints in the optimisation or as additional objectives to discourage constraint violations in the results. Without this work, static optimisation is unlikely to be a suitable method for gaining insight into the motions of jumping systems.

The computational cost of static optimisation is in the order of 10s and 100s of milliseconds, which means that the method may be well suited to support experiments which use large parameter sweeps or iterative optimisation methods. The method may also be particularly useful for real-time computer animation applications, such as video games.

This study did not investigate the effects of starting configurations on the performance of static optimisation. It is anticipated that changing the starting configuration of the system will have a significant impact on the jumping motion produced by the optimisation.

This work demonstrated the application of a predictive modelling technique which is novel in the context of jumping problems. While the work failed to derive insights into jumping systems using the method, a good case has been made in favour of using the method in future studies of jumping systems.

Chapter 5

Dynamic Optimisation

Dynamic optimisation (or trajectory optimisation) is generally used in biomechanics to find the muscle excitation signals which produce an optimal jumping motion. This work considers segmented models with no muscles, thus the muscle excitations are replaced by the external torques acting at each joint. With multiple muscles affecting each degree of freedom in an articulated model, there are an infinite combination of muscle forces which may produce a given set of torques at each degree of freedom. The muscle forces may be recovered using static optimisation, however, static optimisation does not consider previous and future forces in the model, leading to potential of discontinuous muscle force solutions. By considering the entire trajectory, dynamic optimisation is able to account for the time dependencies of muscle forces and find continuous results.

The literature review identified many applications of dynamic optimisation to jumping problems. As such, this chapter will assess the predictive power, computational cost, and intellectual investment of dynamic optimisation in the context of jumping dynamics problems.

5.1 Summary of Dynamic Optimisation

Dynamic optimisation requires a scalar objective function to describe the optimality of a given solution. While static optimisation looks at the current state of a system independent of the trajectory being followed, dynamic optimisation includes the dynamics of the model to find an optimal trajectory through the state space. Dynamic optimisation may be used to find the following in the state space of a given model:

- A trajectory between two given states
- An optimal starting point and trajectory which lead to a given end state
- An optimal end point and trajectory from a given start state

In the case of jumping, it is practical to prescribe a single, fixed starting state which is usually based on the natural, static standing pose of the system. Pre-determination of an end state for the jump is a difficult task as an articulated leg may produce the same take-off velocity from infinitely many configurations, and the optimal trajectories which lead to such states may vary considerably. Thus the final state is not constrained in jumping optimisations. It

is typical to optimise for the maximum jump height (or equivalently the maximum take-off velocity) of a system as this is a simple metric to use in a scalar objective function.

In dynamic optimisation, the problem is transcribed into a finite set of parameters which form a non-linear program. These parameters are often referred to as decision variables. The parameters typically represent the controls and states of the system throughout the trajectory. Those parameters are then optimised using a non-linear program solver such as MatLab's "fmincon" (other solvers include SNOPT [75], SciPy's "optimize" [76] and IPOPT [77]). Direct transcription methods are applied in this work as they are straightforward and transparent to implement. Indirect methods are potentially more accurate, but are beyond the scope of this work.

Transcription of the dynamics is the derivation of constraint functions which are applied to the decision variables, enforcing that the variables describe a motion which obeys the dynamics of the problem. These constraints are specifically referred to as continuity constraints or continuity conditions [78]. Constraints may also be used to limit the controls used or to prevent the system from entering undesirable states. Adding more constraints to an optimisation problem leads to an increase in the the number of gradients which must be computed. Similarly, increasing the complexity of a constraint function increases the computational effort for the gradient of the constraint. Both of these will lead to an increase in the computation time required per step of the optimisation. However, constraints may reduce the search space which may potentially lead to the solution being found in fewer steps. From experience, although it may seem counter intuitive, if an undesirable outcome is anticipated, such as two segments intersecting during a motion, it is best to include the constraint proactively. While additional constraints may increase the computation time of the optimisation, the time saved by not producing undesirable results will outweigh this. This relates to a common dilemma seen in many programming implementations where the code's efficiency is optimised without consideration to the time it takes to optimise the code itself. Spending a day refactoring code to reduce the execution time from 1 hour to 55 minutes is unlikely to be worthwhile, unless the code is being executed many times.

A non-linear program, the result of transcription, typically comes in the form:

$$
\begin{aligned}
\min_{\boldsymbol{x}} \quad & J(\boldsymbol{x}) \\
\text{s.t.} \quad & \boldsymbol{A}\boldsymbol{x} \leq \boldsymbol{b} \\
& \boldsymbol{A}_{eq}\boldsymbol{x} = \boldsymbol{b}_{eq} \\
& c(\boldsymbol{x}) \leq \mathbf{0} \\
& c_{eq}(\boldsymbol{x}) = \mathbf{0} \\
& lb \leq \boldsymbol{x} \leq ub
\end{aligned}
\tag{5.1}
$$

The decision variables, $\boldsymbol{x}$, are the parameters which are to be optimised and generally represent the state and control of the system at each node. The decision variables may also include the duration of nodes as well. The objective function, $J(\boldsymbol{x})$, is a function which produces a scalar value and is used to score the performance of the system based on the decision vari-

ables. In optimal control, and in this work, the objective is thought of as a cost and so the objective is minimised, whereas reinforcement learning consider the objective as a reward and seek to maximise the objective function. The matrix $\boldsymbol{A}$ and vector $\boldsymbol{b}$ are used to apply linear inequality constraints to the decision variables, while $\boldsymbol{A}_{eq}$ and $\boldsymbol{b}_{eq}$ provide linear equality constraints. No linear constraints were used in this work; the relationship between the state and Cartesian positions in jumping models are non-linear sine and cosine functions. The function $c_{eq}(\boldsymbol{x})$ is a non-linear equality constraint function used to constrain aspects of the system to zero. Similarly, $c(\boldsymbol{x})$ is a non-linear inequality constraint function used to constrain aspects to zero or below. c_{eq} is typically used to constrain the decision variables of $\boldsymbol{x}$ which represent the states of the system at the nodes to the numerical solution for the dynamic integration. *lb* and *ub* represent the lower and upper bounds on the decision variables respectively.

This work considers direct transcription methods as they are straightforward and transparent to implement. Indirect methods while potentially more accurate, are beyond the scope of this work. The two main methods of transcription are multiple shooting and direct collocation. Multiple shooting derives the constraints by explicit integration of the dynamics between nodes. Using the decision variables at node i, a numerical solution is found for the system's configuration at node $i+1$ according to the dynamics. This solution then forms the constraint for the decision variables at node $i+1$, and so on. Direct collocation uses implicit integration of the dynamics and produces a piece-wise polynomial representing the trajectory of the system based on the decision variables. The gradient of the polynomial represents the dynamics of the system and is constrained to match the rate of change of state defined by the dynamics for a given node. Maintaining the straightforwardness of methods, this work uses multiple shooting for its clear implementation.

Gradient based optimisation generally finds a local minimum. By changing the starting values of the decision variables, the optimiser may obtain different gradients and find a different local minimum. Using a range of starting values can help in finding various minima of the problem, one of which may be the global minimum of the problem. In reality, the decision variables comprise many parameters and performing a search in this manner may require a significant amount of time if the optimisation itself takes a long time to run. Instead, the starting values may be chosen based on an estimate of a solution, it is common practice to either use experimentally measured motion data where available, or to linearly interpolate the trajectory of the system and control signal, to produce a starting set of decision variables. For example, if the system starts in state $\boldsymbol{s}$ and the objective is maximise jump height, a plausible solution may be for the system to increase the height of the body over the duration, thus a starting point of the optimisation would be to set the decision variables such that the state $\boldsymbol{s}$ follows this motion. Note that the solver handles constraints and the objective so it is possible to start the optimisation with an infeasible motion.

5.2 Candidate Constraint and Objective Functions

In this work, dynamic optimisation was used to find the external torque trajectories which produced an optimal jumping motion for the *Numida meleagris* model described in Section 4.3. The increased complexity of dynamic optimisation compared to static optimisation brings about a different set of questions about the optimality of the motion. In static optimisation, the trajectory of the system is defined by the prescribed body acceleration which was obtained from experimental work and may in the future be generated using polynomial functions. Dynamic optimisation requires an objective function for the trajectory followed by the system. This requirement of the user to produce a scalar function to describe how "good" a jumping motion was is notoriously difficult for many dynamic problems [79]. Constraint functions are used both to ensure that the solution obeys the dynamics of the model, and to limit aspects of the solution such as the maximum power requirements. It is possible that any of the following may need to be considered by a jumping system or its designer:

- No slipping
- Limit angular momentum at take-off
- Exceed a minimum take-off velocity or maximise take-off velocity
- Control the direction of take-off velocity
- Minimise the time to take-off
- Maximise energy efficiency or move within the limits of actuators

Components of an objective function generally do not share units and so weighting parameters are also included to ensure the correct trade off between objectives is made. This may be done using trial and error or as a means of scaling the objectives to the same unit size. Here, c is used to denote a weighting parameter.

The following sub-sections describe constraints and objectives which may be used to ensure or encourage such goals in jumping for the benefit of readers wishing to implement dynamic optimisation. Some of these functions are used in the case study which follows this section.

No Slipping

As the system pushes against the ground it receives a reaction force parallel to the ground due to friction. Should the system push against the ground with horizontal force greater than the available friction force the foot would slide along the ground, potentially leading to the system falling to the ground. As such, it may be necessary to keep the horizontal ground reaction force within the limits of the available friction. Such a constraint may be defined as:

$$F_h \leq \mu F_v$$

where F_h is the horizontal reaction force from the ground to the foot and F_v is the vertical reaction force. This constraint may need to be relaxed close to the point of take-off as the vertical reaction force approaches zero.

Angular Momentum Limit

During the push off stage the system generates momentum, some of which may be angular momentum about the centre of mass of the system. Rotation of the system during the flight stage of a jump may lead to the system landing with an unfavourable orientation. Some jumping systems, such as insects, accept such conditions, sacrificing the ability to land in a controlled manner for more explosive, catapult-like jumps. Such jumps are generally favoured by small systems as their terminal velocity is low enough that a landing will not result in significant damage, whereas larger systems tend to limit their angular momentum to ensure they land safely. Thus, the angular momentum of the system should be constrained within limits which ensure the safe landing of the system. For large systems where constraining the angular momentum on take-off is a likely requirement, it has been shown that consideration of angular momentum negatively impacts the maximum jump height attainable in athletic high jumping [80]. Wilson et al. propose that angular momentum constraints are considered in jumping optimisations because of this effect. The angular momentum of the system, $I\omega$, may be constrained according to:

$$I\|\omega\| \leq I\omega_{\text{max}}$$

where $I\omega_{\text{max}}$ represents an absolute maximum angular momentum of the system. Similarly, the objective function may include a penalty to deter solutions with high angular momentum without the need to determine a hard constraint for the problem:

$$J(\boldsymbol{x}) = c\, I\|\omega\|.$$

Minimum Take-off Velocity

A low airspeed in flight will lead to stalling. when jumping is used to take off for flight, the system needs to surpass this minimum airspeed velocity to ensure sufficient lift is generated for flight. Any velocity above this value is not wasted, it is generally the case that generating velocity by pushing from solid ground is more efficient than pushing from air, thus the system saves effort in the flight stage by producing more velocity in the push off stage. This minimum take-off velocity may be included as a lower bound constraint on the take-off velocity, $\boldsymbol{v}_{\text{TO}}$:

$$\|\boldsymbol{v}_{\text{TO}}\| > v_{stall}$$

where v_{stall} is the stall speed of the system.

Maximise Take-off Velocity

The most common objective in jumping optimisation, maximising take-off velocity is a demonstration of the system's maximum capabilities. For a point mass system this is an objective function of:

$$J(\boldsymbol{x}) = -c\,\|\boldsymbol{v}_{\text{TO}}\|$$

For a system with rigid segments, the take-off velocity is not the velocity of the body at take-off. The momentum of the system must be considered as much of the leg's momentum will not be directed the same way as the body. Thus, the objective function becomes:

$$J(\boldsymbol{x}) = -\frac{c}{m}\sum_{i=1}^{n} m_i \boldsymbol{v}_i$$

where m is the total mass of the system, m_i is the mass of the i^{th} segment and $\boldsymbol{v}_i$ is the velocity of the centre of mass of the i^{th} segment. Note that this does not account for the angular velocities of each segment about their centre of mass.

Directed Jumping

In a prey-seeking jump, an element of path planning must be involved to ensure the predator doesn't miss its target. For example, jumping spiders exhibit time minimal, directed jumps to a target location [81]. It is common practice to ignore the effects of control actions during flight and to treat the system as having ballistic motion throughout the flight stage. The path taken by the system under this assumption is well defined and so constraining the jump to pass through a desired point is a case of integrating the ballistic motion of the system and checking that it intersects the desired point. To achieve this, the dot product of a desired direction and the take-off velocity may be used as a hard constraint:

$$\frac{\boldsymbol{v}_{\text{TO}}}{\|\boldsymbol{v}_{\text{TO}}\|} \cdot \hat{\boldsymbol{d}} = 1$$

where the take-off velocity must be normalised to a unit length and $\hat{\boldsymbol{d}}$ is the unit vector which represents the desired take-off direction. Alternatively, the direction of the jump may be set as part of the objective:

$$J(\boldsymbol{x}) = -c\,\frac{\boldsymbol{v}_{\text{TO}}}{\|\boldsymbol{v}_{\text{TO}}\|} \cdot \hat{\boldsymbol{d}}.$$

This objective may be adjusted to also have the system maximise take-off velocity by using the original vector of $\boldsymbol{v}_{\text{TO}}$:

$$J(\boldsymbol{x}) = -c\,\boldsymbol{v}_{\text{TO}} \cdot \hat{\boldsymbol{d}}.$$

Minimum Time to Take-off

Evading a predator may require a system to take-off in as little time as possible. The jump may also require a minimum velocity to ensure sufficient distance from the predator is gained.

Minimum time trajectories can be set up by including the duration of the trajectory in the parameters under optimisation. When doing this, the time between nodes is calculated at each iteration based on the current duration parameter. This method may be taken one step further by using parameters for each node's duration, this is used when some regions of the trajectory require fine controls while others suit coarse controls. The system may be constrained in that it must take-off after a maximum of t_{max} seconds:

$$t_{\text{TO}} \leq t_{\text{max}}$$

where t_{TO} is the time until take-off. Or the system may be encouraged to minimise the time until take-off as part of the objective:

$$J(\boldsymbol{x}) = c\, t_{\text{TO}}.$$

Efficiency and Actuator Limits

Systems which use jumping as their main method of locomotion are likely to aim to minimise the energetic cost of jumping. Such jumps may include an objective function which seeks to minimise the total work, W, or the maximum power, P_{max}, required for the motion:

$$J(\boldsymbol{x}) = c\,\frac{E_T}{W} \qquad \text{or} \qquad J(\boldsymbol{x}) = c\, P_{\text{max}}.$$

Where E_T is the total energy of the system at take-off. The total work may be approximated using the angular displacements and the applied torques at each joint. Similarly, power may be determined using angular velocities and applied torques at each joint. Both of these come from the decision variables, however, a more accurate approximation of the work done may be found using trajectories generated by numerical integration between nodes, though this may incur a large computational load on the optimisation. Energetic objectives are not useful without constraints or further objectives to enforce motion, as the solution to a minimisation of total work is to do absolutely nothing. A constraint such as a minimum take-off velocity should ensure the solution found reaches such a velocity. Perhaps the most common objective function is to penalise the sum of squared torques applied to the system:

$$J(\boldsymbol{x}) = c \sum_{i=1}^{n} \tau_i^2.$$

While the objective has no clear physical meaning, it works to smooth the applied torques and reduce large spikes while also encourage the spreading of torques across all degrees of freedom rather than a single one. The objective also somewhat relates to work but does not consider the motion of the system. Without using integration, a power minimising objective may be more desirable:

$$J(\boldsymbol{x}) = c \sum_{i=1}^{n} \text{abs}(\boldsymbol{\tau}_i^T)\text{abs}(\dot{\boldsymbol{q}}_i).$$

The absolute value is used to ensure torques which extract energy from the system are not encouraged by the objective function.

5.3 Method

This section demonstrates the application of dynamic optimisation to the *Numida meleagris* model described in Chapter 4. Optimisations were performed using a range of node counts between 2 and 50. The addition of nodes increases the number of decision variables and continuity constraints which must be evaluated at every iteration, resulting in increased computation times. In this study, the quality of optimisation results decreased with increasing number of nodes until the nodes become close enough together that the dynamics of the system were effectively linearised at which point the optimisation produced better results.

The optimisation assumes that the starting state of the system is given. Using the direct multiple shooting method, the optimisation problem was transcribed using n nodes. The transcribed parameters included $n-1$ states, $n-1$ control inputs and a single value, t_d which represented the duration of the trajectory. Nodes were equally spaced from the start to end of the trajectory according to t_d. Parameter values aligned to the times of the trajectory as shown in Table 5.1.

Table 5.1: Multiple Shooting parameters and their relevance to the trajectory's duration. t_1 corresponds to the start time of the trajectory and each index thereafter represents the time at that node. "x" denotes the absence of a parameter at a given node in the trajectory.

Node	1	2	...	$n-1$	n
Time	t_1	t_2	...	t_{n-1}	t_n
State	x	s_2	...	s_{n-1}	s_n
Control	u_1	u_2	...	u_{n-1}	x

Controls were treated as step inputs with the torque at a node applied as a constant value until the next node, this provided the simplest representation of the control input while producing plausible results. The parameterised torques were bounded according to the torques produced by the static optimisation model in Chapter 4. The starting state was not parameterised as this was fixed for the problem. The control at the final time of the trajectory was not necessary as no further integration would occur. An additional control parameter at the end of the trajectory would have allowed the system to quickly end the jump by changing the applied torques to pull the foot from the ground, however, the results produced without the additional control parameter were deemed sufficient. The end state of the trajectory was included to provide the solver with information about the objective function as take-off velocity was the main objective for optimisation. The dynamics of the system were integrated using the explicit Runge-Kutta 4,5 method implemented in MatLab's ode45 function. The optimisation was used to find the external torques, and resulting trajectory through state space,

which minimise the following objective function:

$$J(\boldsymbol{x}) = -\boldsymbol{v}_{\mathrm{d}} \cdot \boldsymbol{\rho}_n + c\frac{1}{n}\sum_{i=1}^{n-1} \boldsymbol{u}_i^T \boldsymbol{u}_i. \tag{5.2}$$

$\boldsymbol{\rho}_n$ is the momentum of the system at the end of the trajectory:

$$\boldsymbol{\rho}_n = \sum_{i=1}^{n} m_i \boldsymbol{v}_i.$$

$\boldsymbol{v}_{\mathrm{d}}$ is a unit length vector which represents the desired take-off velocity direction extracted from the experimentally measured data presented by Henry et al. [6]

$$\boldsymbol{v}_t = \begin{bmatrix} 0.3846 \\ 0.9231 \end{bmatrix}. \tag{5.3}$$

. The weighting parameter, c, was set to 0.01. $\boldsymbol{u}_i$ is the vector of external torques applied for the duration of node i. Momentum was used to account for the motion of the leg segments as well as the motion of the body, considering that the leg mass constitutes around 25-35% of the total system mass [6], [39]. If only the velocity of the body were considered, then motions with a large hip angular velocity, which result in a large velocity of the body, may be selected. Such a motion would result in a lower take-off velocity than anticipated as the leg's momentum would reduce the overall momentum of the system.

5.3.1 Undesirable Solutions from Dynamic Optimisations

This section will discuss some of the dynamic optimisation solutions obtained which were not anticipated, nor desirable. The methods used to ensure such results were not obtained are presented and the effects they have on the solution are discussed. In this work constraints were included in response to undesirable solutions. The solutions and the constraints which fixed them are presented here so that they might help researchers experiencing similar issues.

Solution moves below the ground

A common result of the optimisation was for the system to fall under gravity and then use an exceptionally large stroke length to jump from under ground. As the dynamics ignore ground reaction forces this was a feasible solution albeit unrealistic. To prevent such motions, state parameters were constrained such that each segment remained above the ground. The vertical position of the i^{th} segment's end point, y_i is given by:

$$y_i = \begin{bmatrix} J_{2,1} & J_{2,2} & \dots & J_{2,i} \end{bmatrix} \boldsymbol{q}. \tag{5.4}$$

The negative of the vertical positions were then used as inequality constraints, forcing the solution to keep segments above the ground. The tarsometatarsus position was constrained

separately by bounding the toe angle to be between $0 \leq q_1 \leq \pi$. The drawback to this method is that the constraints are only enforced at the nodes, the system may pass through the ground in between nodes and the constraints while satisfying the constraints at the nodes. This might be remedied by increasing the number of nodes in the transcription, at the cost of increased computation time, or by adding the position constraints at times in between nodes as well as at the nodes themselves.

Studies found in the literature use collision detection and ground contact models, typically based on spring-damper systems, within the system model to prevent ground penetration [12], [15], [16], [34]. Such practice removes the need to include optimisation constraints and thus eliminates the issue of constraint violation between nodes. The problem with such methods is that they require careful tuning of contact force models to ensure the system is not able to take advantage of contact forces in producing unrealistic results, akin to jumping from a trampoline instead of solid ground. By rejecting solutions which penetrate the ground via the optimisation constraints, tuning of the model is avoided.

Solution violates the no pulling assumption

In order to use the vertical ground reaction force to determine the take-off of the system, the rest of the jumping motion must not involve any pulling on the ground, i.e. $F_v \geq 0$ must hold true for the entire motion except the moment of take-off. This was enforced by constraining the vertical ground reaction force to be $F_v > 0$ for all nodes excluding the final node, at which point the reaction force was constrained to be $F_v \leq 0$ as the system breaks contact with the ground and successfully jumps. Note this excludes stutter jump motions although they may be beneficial according to Aguilar et al. [47], but it does not exclude countermovements where the body is lowered with a positive vertical ground reaction force which is less than the weight of the system.

Solution uses joint constraint forces to its advantage

The constraint forces of a revolute joint prevent linear motion between joined segments. These forces are especially large when two segments align and the system is moving along the direction of alignment. An optimisation may use a whip-like motion to gain significant acceleration during the motion. In reality this is an undesirable effect as the ligaments and supports in the joint are responsible for producing such forces with large forces potentially leading to damage in the joint. Sellers et al. constrain the stress in bone structures to be below a given limit in determining the running gait of Tyrannosaurus rex [82]. Their method was not implemented in this work. A similar solution may be to constrain the joint angular accelerations at nodes to be within certain limits, this would discourage the large accelerations of a whip-like motion, however, the constraints may require some amount of trial and error to determine the acceleration limits necessary to prevent the use of joint constraint forces without preventing plausible jumping motions.

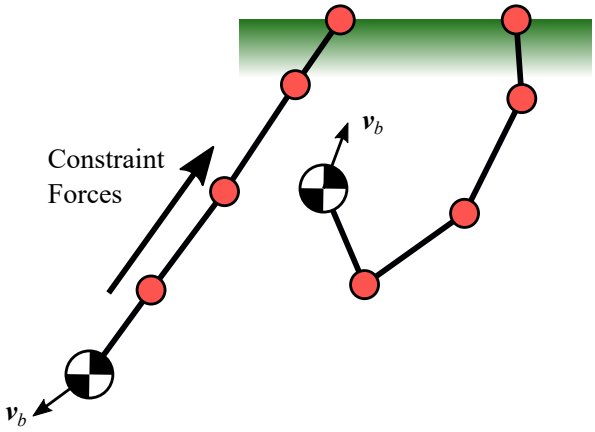


Figure 5.1: Example motion in which the body moves with velocity in the direction of the leg alignment. As the the segments are all aligned, the body reaches the maximum extension and the revolute joint constraint forces prevent further motion. This results in a large acceleration of the body in the opposite direction, similar to the crack of a whip.

Solution includes segment intersections

Some optimisation results involved the intersection of segments as no collision dynamics were included in the model. This problem has a straightforward solution if the intersecting segments are adjacent as the angle of one segment relative to the other segment may be constrained to $-\pi < \hat{q}_n < \pi$. However, if the intersecting segments have an intermediate

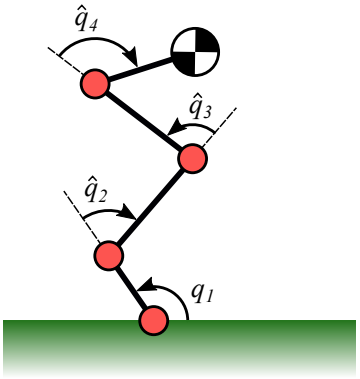


Figure 5.2: Example of joint angles defined relative to the previous adjacent segment. Note that q_1 is still defined relative to the inertial frame of reference.

segment between them, the segments must be treated as line segments in Cartesian space to determine if an intersection occurs. The method proposed by Antonio [83] provides an effective means of determining whether two line segments intersect or not. The difficulty with this solution is that the segments either intersect, or they do not. This means that a constraint function for intersection will be discontinuous and therefore problematic for a gradient descent method, such as fmincon. For this reason, the intersections of non-adjacent segments was not included in the constraints. The end point position of the i^{th} segment, (x_i, y_i) is defined as:

$$\begin{bmatrix} x_i \\ y_i \end{bmatrix} = \begin{bmatrix} J_{1,1} & J_{1,2} & \dots & J_{1,i} \\ J_{2,1} & J_{2,2} & \dots & J_{2,i} \end{bmatrix} \boldsymbol{q}. \tag{5.5}$$

Intersections between non-adjacent segments may occur between the following segments in the *Numida meleagris* model:

- Tarsometatarsus - Femur
- Tarsometatarsus - HAT
- Tibiotarsus - HAT

The position of the tarsometatarsus which is joined to the ground is defined as $(0, 0)$. The intersection point is not a requirement as no intersection of segments should occur.

An alternative solution would be to include constraint forces directly in the model. Anderson and Pandy use spring-like ligament torques to prevent the segments from reaching physically impossible angles [15]. One issue with such models is that the optimisation may find solutions which use the constraint forces to their advantage. Another issue with using spring-like constraint forces is that the appropriate force model for the constraint depends on the motion and inertia of the system to which they are applied. This means that the constraints must be tuned to the model to ensure their effect is appropriate. Interestingly, Anderson and Pandy also include a penalty in the objective function to prevent joint hyper-extension, implying that the constraint forces may have been insufficient for preventing such motions.

5.4 Results

The optimisation results are presented for the following range of node counts:

$$n = \{2, 3, 4, 5, 10, 25, 50\}.$$

This is done to show the variation in results with the coarseness of the trajectory discretisation. The results are presented along side the motion capture data obtained by Henry et al. [6]. The objective score for each result is presented in table 5.2. The resulting joint angle trajectories, joint torques and finally the ground reaction forces are presented in this section.

The following graphs are representative of the results presented in this section. Each of these graphs are labelled to show how the results will be presented.

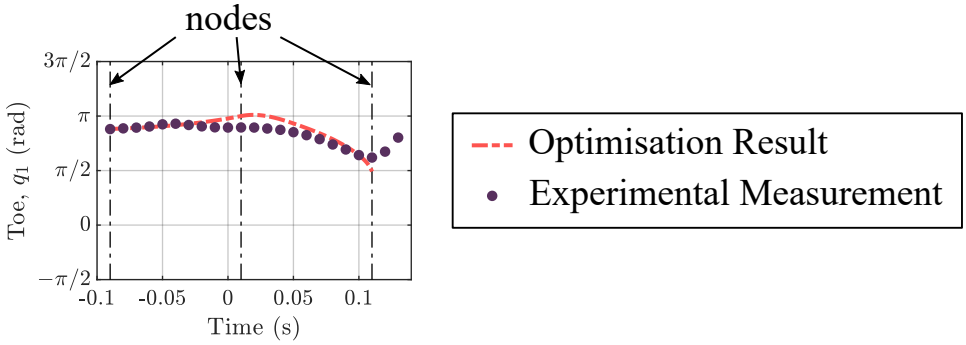


Figure 5.3: Example joint angle trajectory optimisation result using 3 nodes. The resulting duration of the optimised trajectories vary from each other and the experimentally measured data. The final node represents the end of the trajectory and the point of take-off.

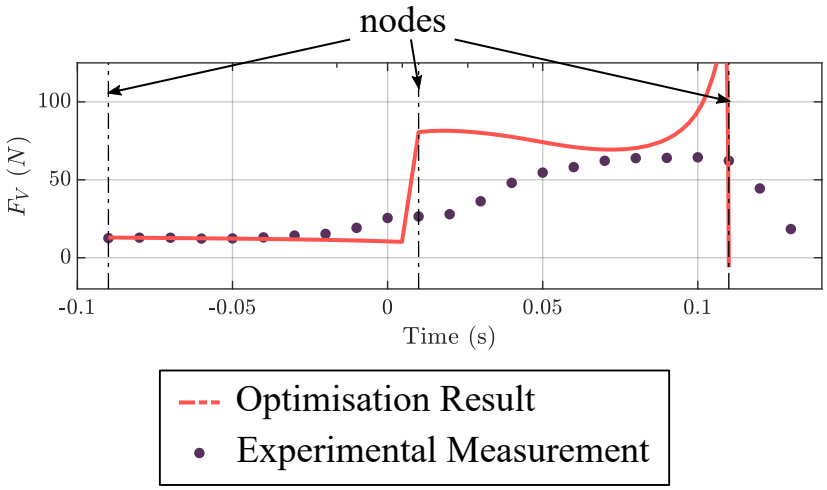


Figure 5.4: Example ground reaction force from optimisation result using 3 nodes. The reaction forces change abruptly due to the step change in torque. However, the step change at the end of the trajectory occurs over a longer duration and is not due to a step change in torque; the change in reaction force at the end of the trajectory is due to the leg of the system approaching full extension.

Table 5.2: The final objective function scores and computation times for each optimisation run. Note that the objective function represents a cost and so lower values are considered to be better.

Number of Nodes	2	3	4	5	10	25	50
Objective Score	0.41	-6.14	-5.43	-3.99	-6.55	-6.74	-7.04
Computation Time (s)	2.00	25.8	41.6	56.7	219	1990	32500

5.4.1 Joint Angle Trajectories and Sample Poses

The resulting joint angle trajectories are presented here with the experimentally measured data from *Numida meleagris* [6]. Poses of the system sampled at equal intervals in time throughout the trajectory are presented to demonstrate the resulting motions. No example poses are presented for the 2 node optimisation as the motion was infeasible.

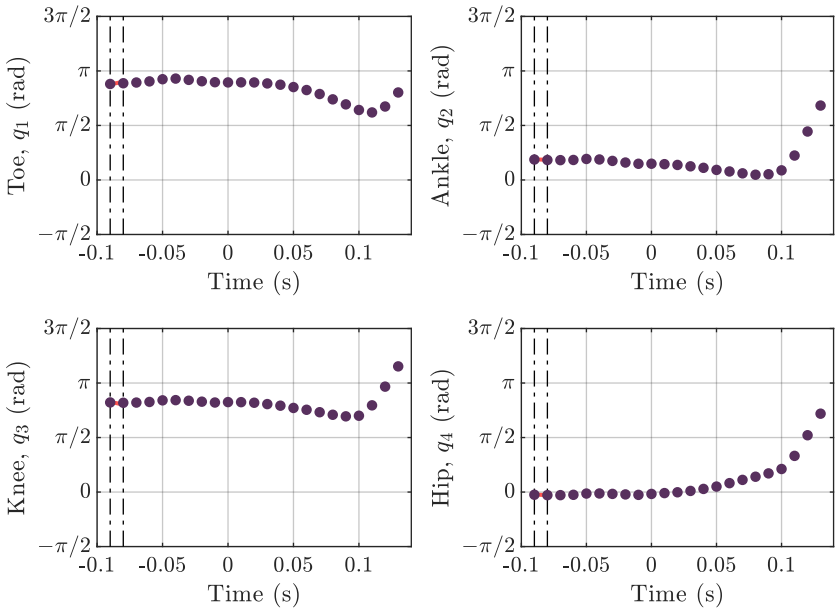


Figure 5.5: Dynamic optimisation joint angle trajectories with 2 nodes. The optimisation failed to converge. The time parameter t_d was removed in testing but the optimisation still failed to find a feasible solution.

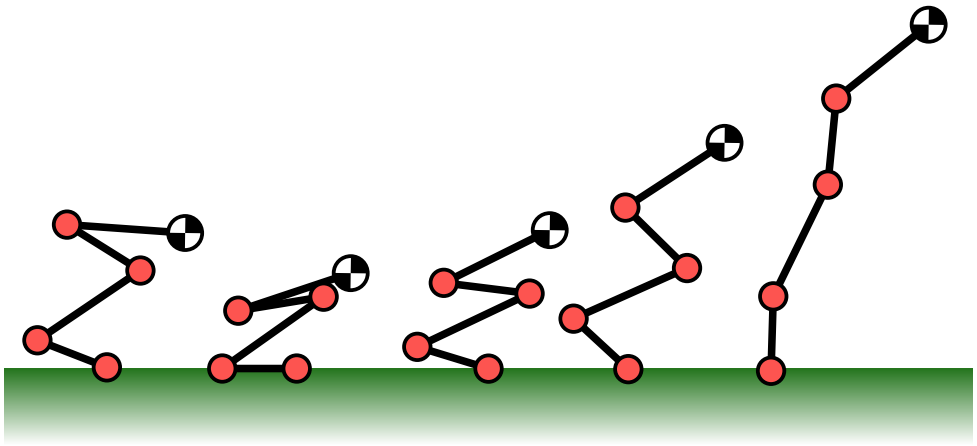


Figure 5.6: Poses of the system at equal intervals throughout the resulting trajectory for the 3 node optimisation.

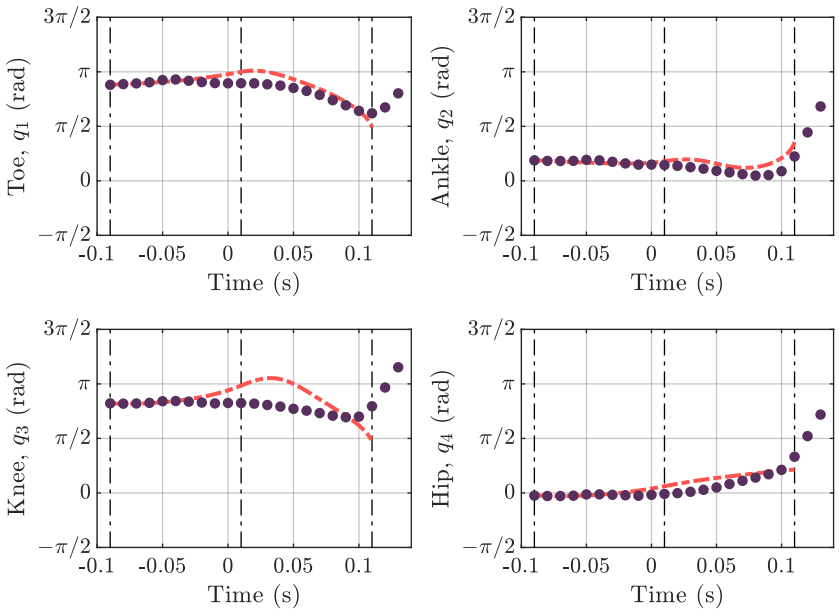


Figure 5.7: Dynamic optimisation joint angle trajectories with 3 nodes.

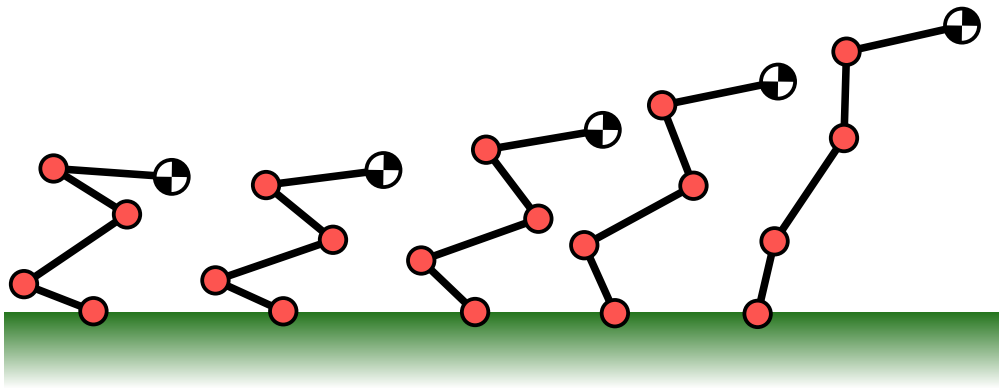


Figure 5.8: Poses of the system at equal intervals throughout the resulting trajectory for the 4 node optimisation.

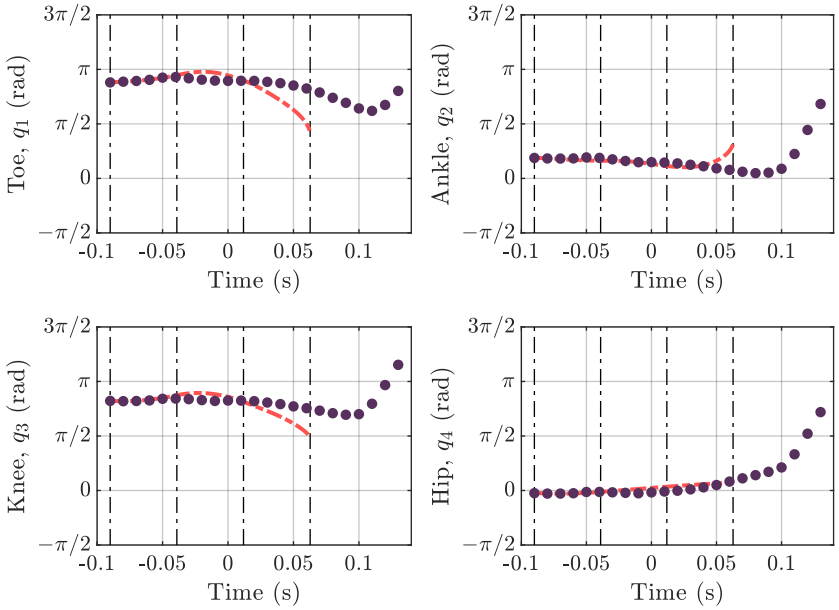


Figure 5.9: Dynamic optimisation joint angle trajectories with 4 nodes.

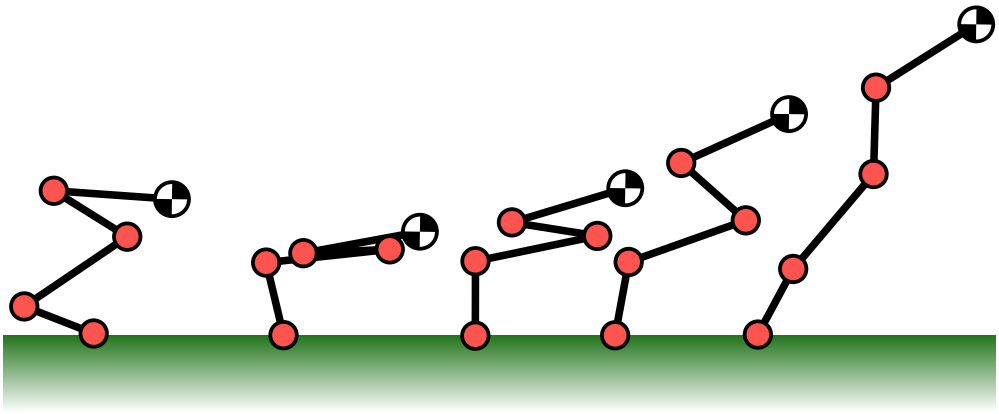


Figure 5.10: Poses of the system at equal intervals throughout the resulting trajectory for the 5 node optimisation. The poses show the segments almost overlapping.

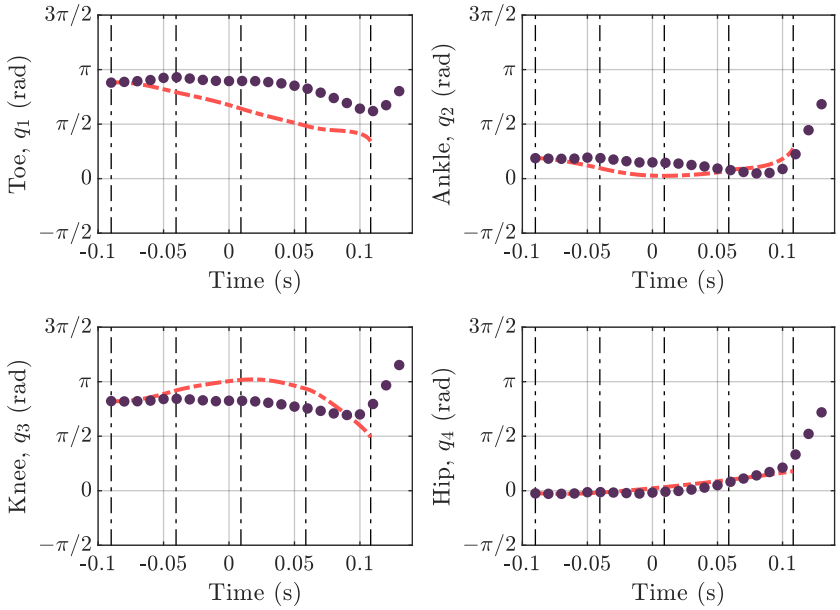


Figure 5.11: Dynamic optimisation joint angle trajectories with 5 nodes.

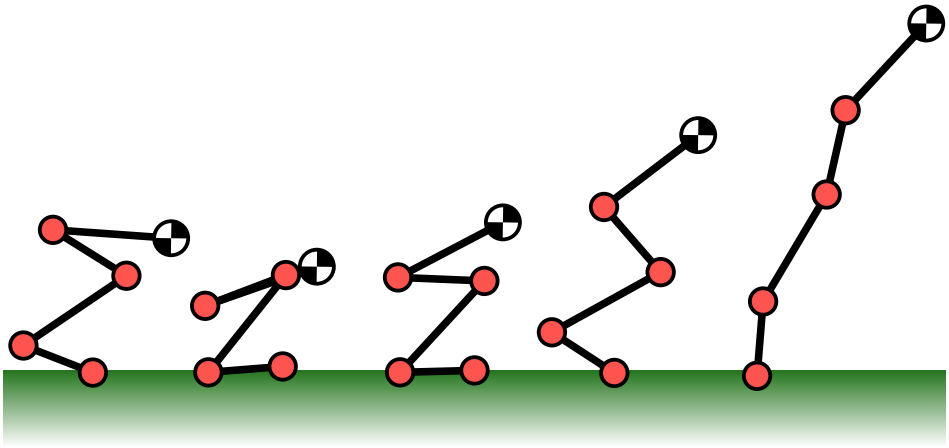


Figure 5.12: Poses of the system at equal intervals throughout the resulting trajectory for the 10 node optimisation. With 10 nodes the segments are seen to overlap, this occurs in the time between nodes.

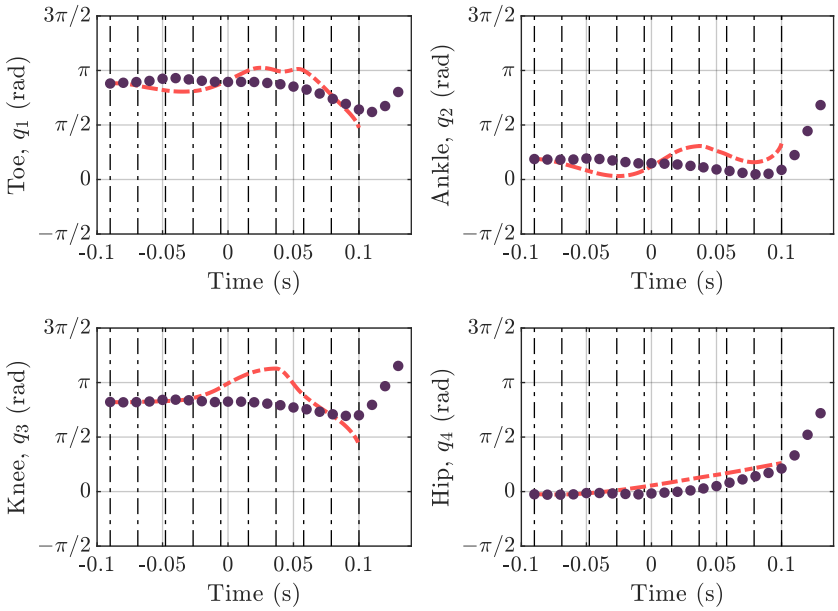


Figure 5.13: Dynamic optimisation joint angle trajectories with 10 nodes.

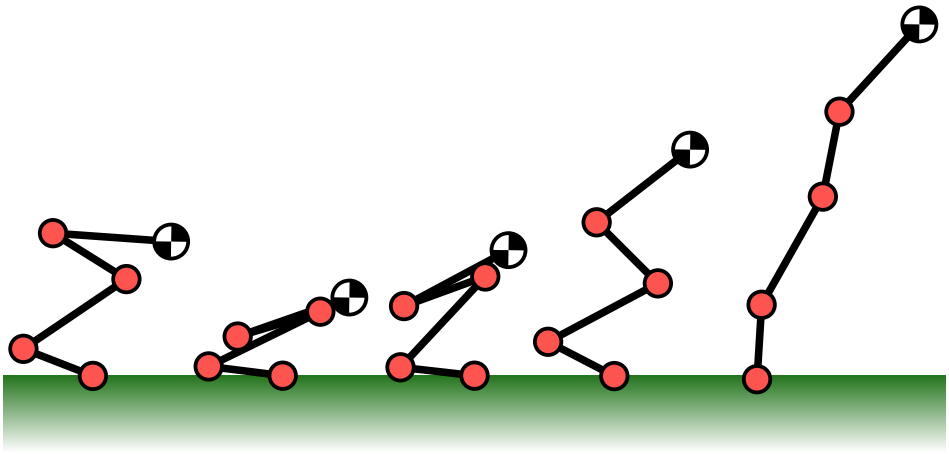


Figure 5.14: Poses of the system at equal intervals throughout the resulting trajectory for the 25 node optimisation. The system compresses much more in the preparation stage than in results from fewer node optimisations.

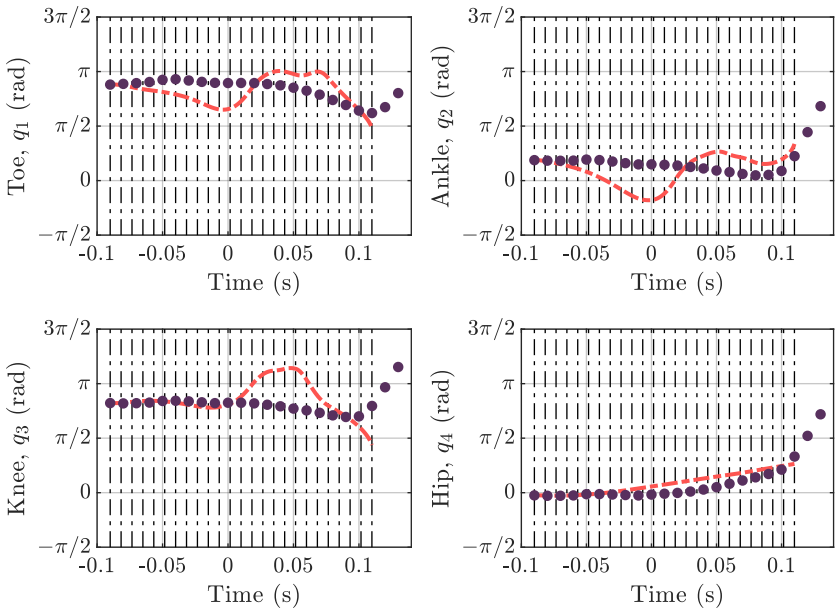


Figure 5.15: Dynamic optimisation joint angle trajectories with 25 nodes.

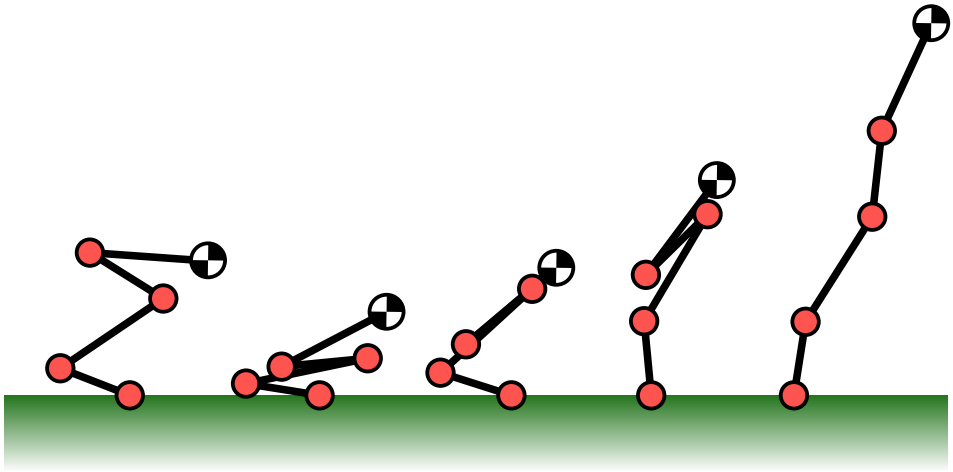


Figure 5.16: Poses of the system at equal intervals throughout the resulting trajectory for the 50 node optimisation. The resulting motion involves significant compression of the system throughout the motion.

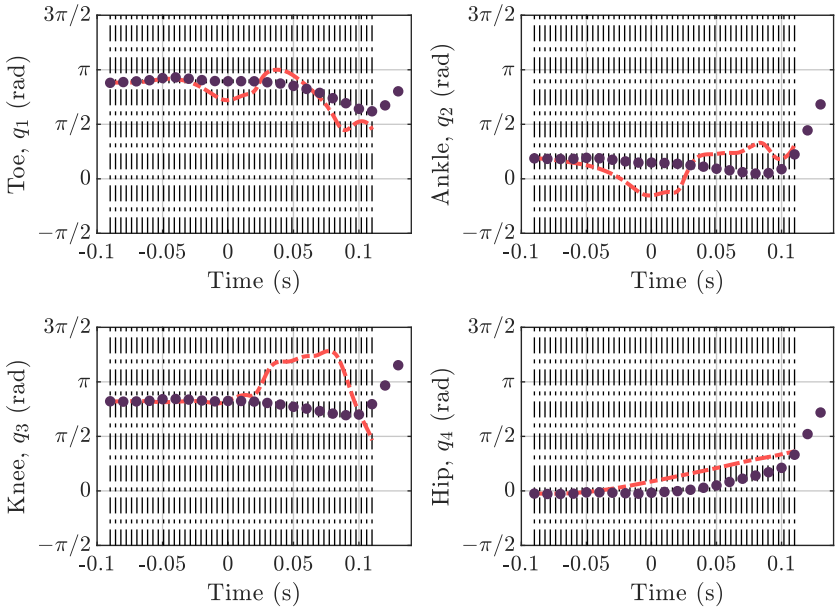


Figure 5.17: Dynamic optimisation joint angle trajectories with 50 nodes.

5.4.2 Torques

This section presents the resulting torque trajectories of the optimisations.

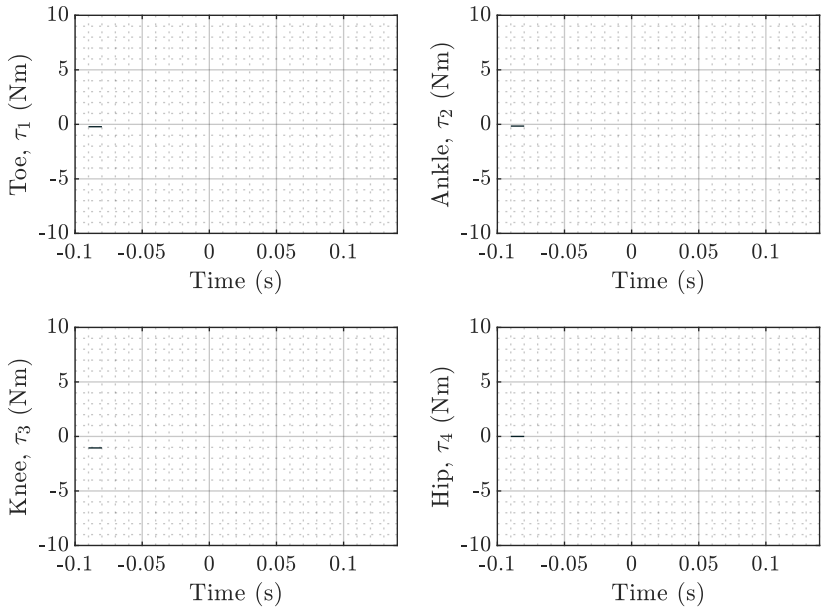


Figure 5.18: Dynamic optimisation control signal with 2 nodes. As the optimisation failed, no meaningful torques were produced.

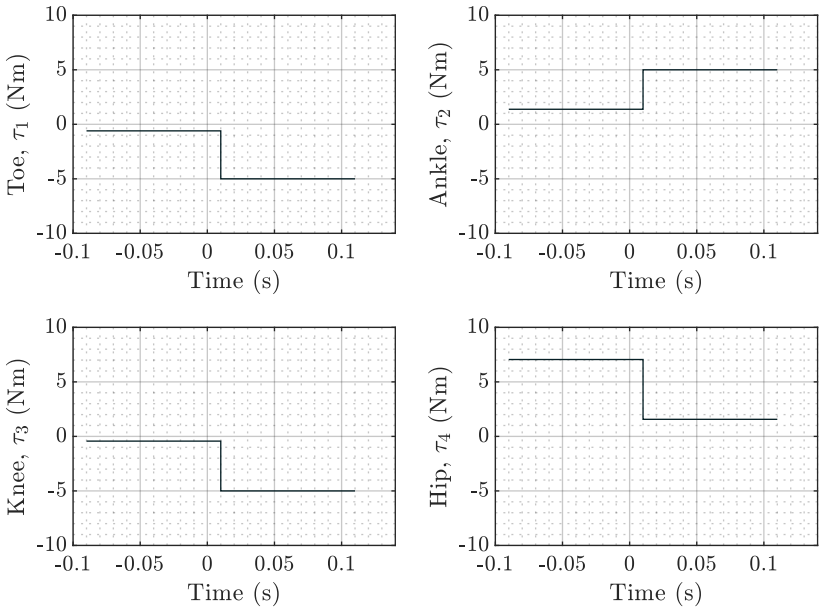


Figure 5.19: Dynamic optimisation control signal with 3 nodes. The hip joint is used most in the initial half of the motion with the remaining joints activating in the second half of the motion.

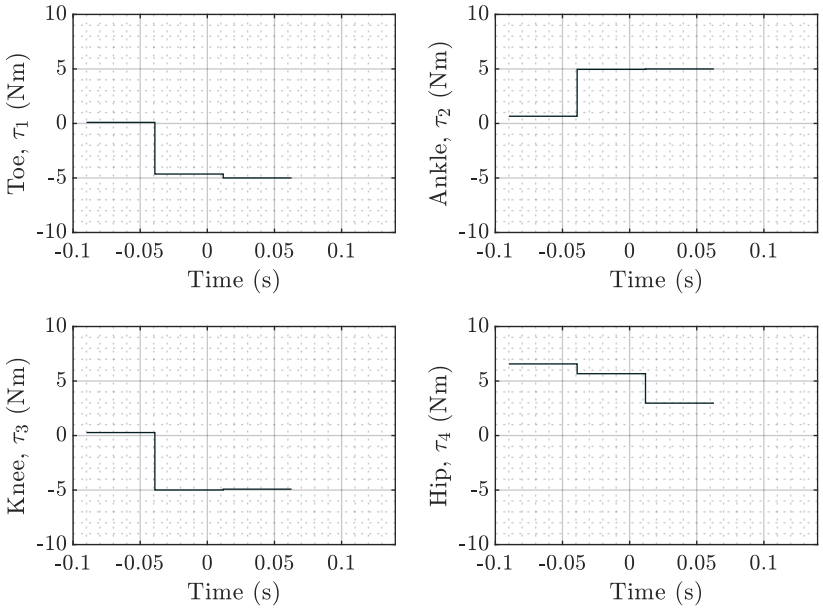


Figure 5.20: Dynamic optimisation control signal with 4 nodes. The toe, ankle and knee joints are not significantly used in the initial stage of the motion.

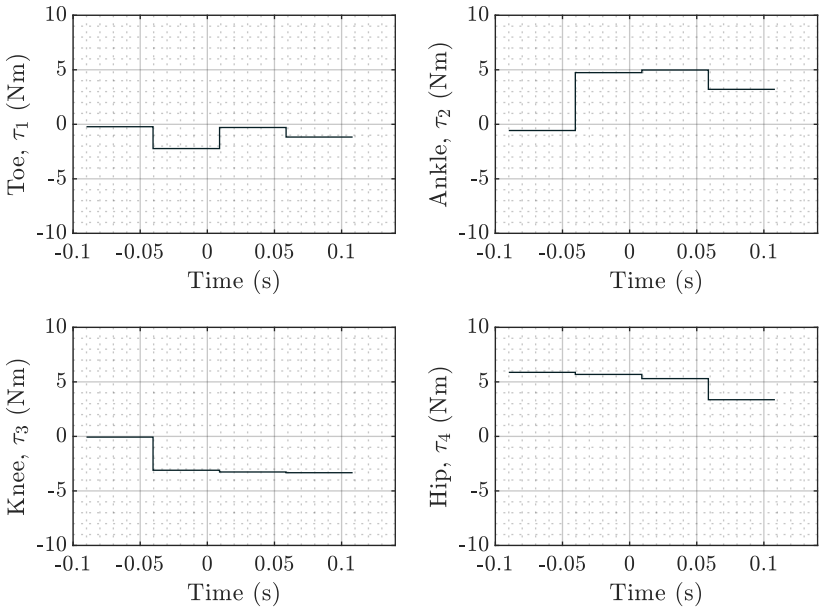


Figure 5.21: Dynamic optimisation control signal with 5 nodes. Little torque is applied at the toe joint throughout the motion. The knee and ankle are not used in the initial stage of the motion.

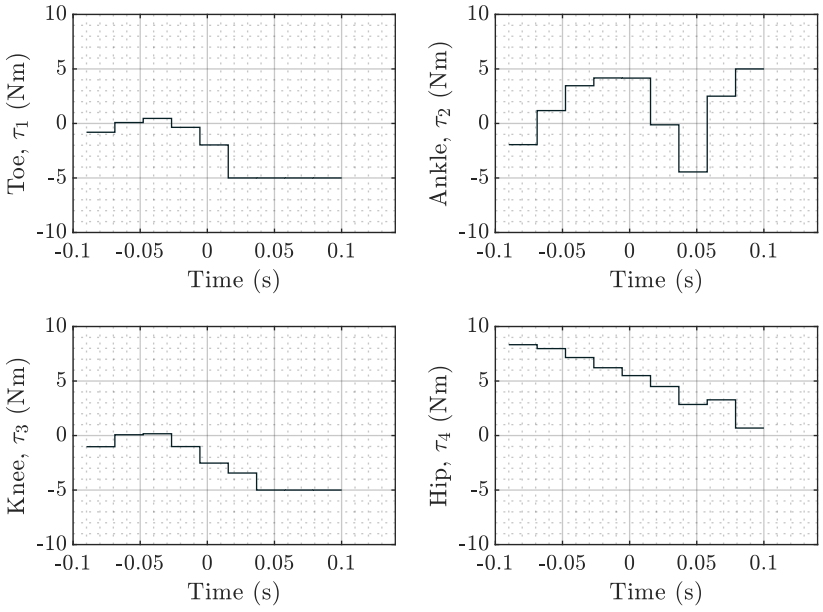


Figure 5.22: Dynamic optimisation control signal with 10 nodes. The toe and knee joint torques are not used a lot in the initial stage of the motion.

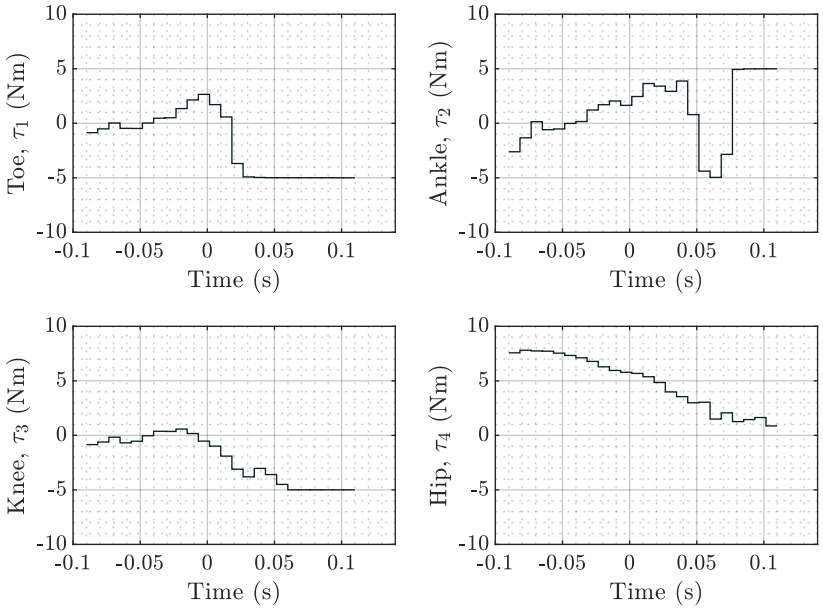


Figure 5.23: Dynamic optimisation control signal with 25 nodes. With more nodes the finer adjustments in the applied torques become more clear.

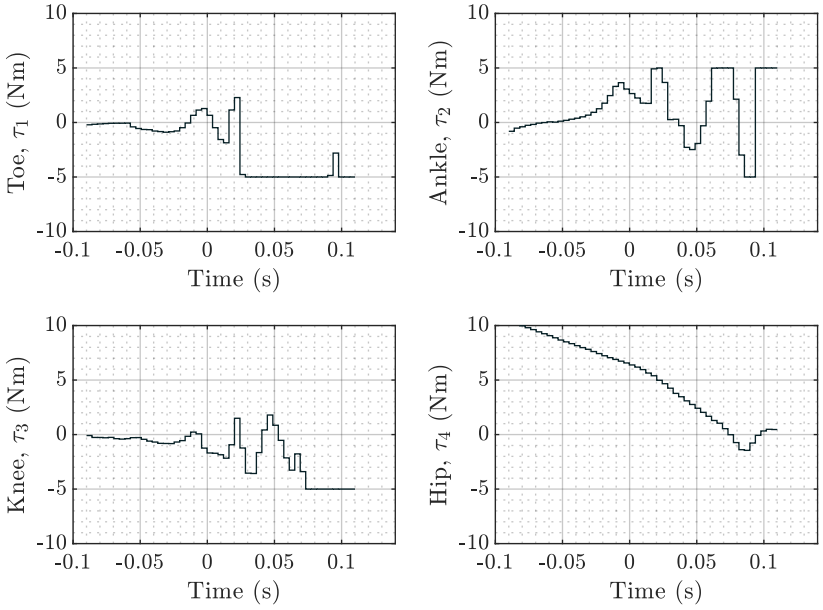


Figure 5.24: Dynamic optimisation control signal with 50 nodes. Large oscillations in the torque applied occur in these results. Especially in the knee and ankle joints.

5.4.3 Ground reaction forces

The horizontal and vertical ground reaction forces, F_h and F_v are presented here. Due to the ability of the system to incur step changes in the applied torques, the reaction forces exhibit significant discontinuities which is not realistic to a biological jumping system. However, this may be similar to the ground reaction forces of manipulators or legged jumping robots.

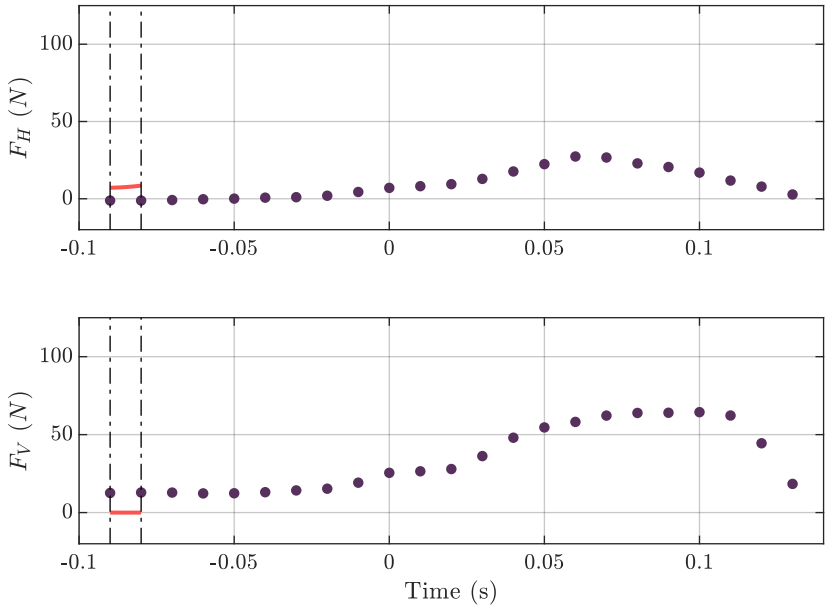


Figure 5.25: Dynamic optimisation ground reaction forces with 2 nodes.

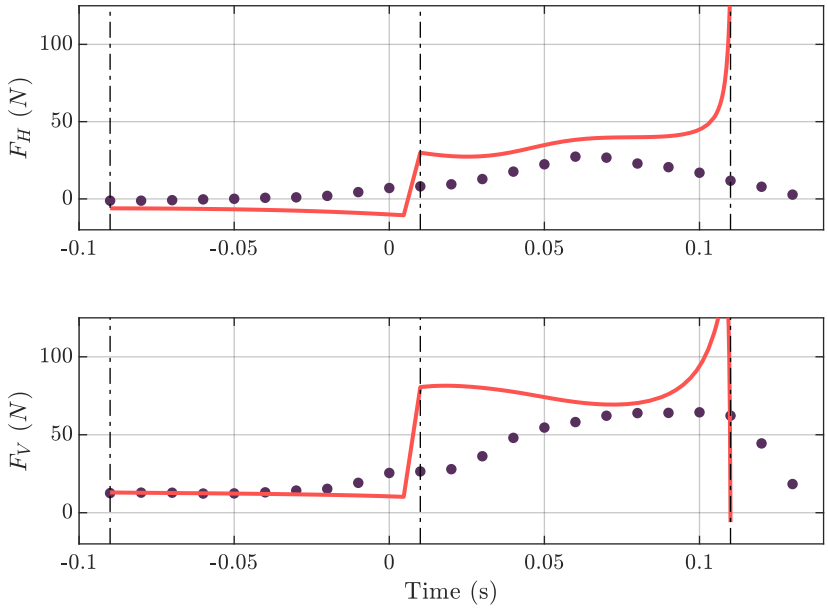


Figure 5.26: Dynamic optimisation ground reaction forces with 3 nodes.

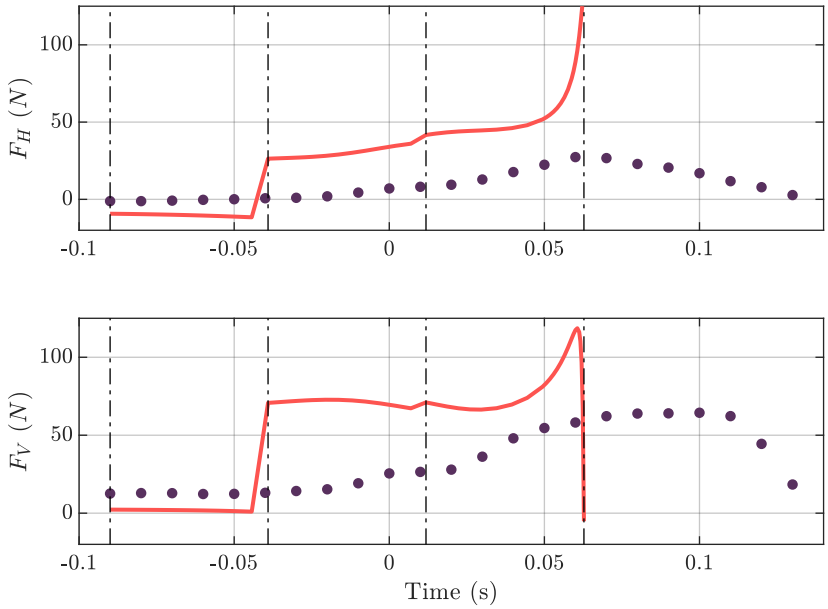


Figure 5.27: Dynamic optimisation ground reaction forces with 4 nodes.

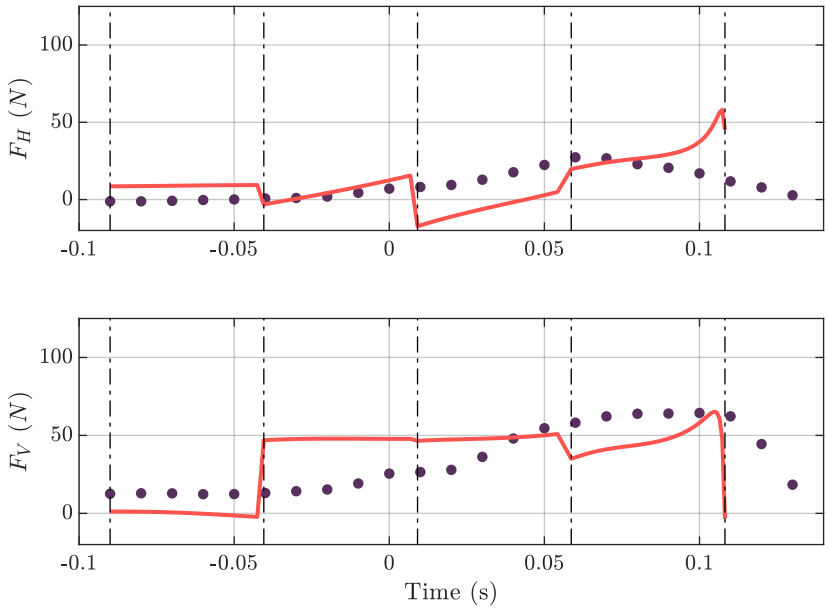


Figure 5.28: Dynamic optimisation ground reaction forces with 5 nodes.

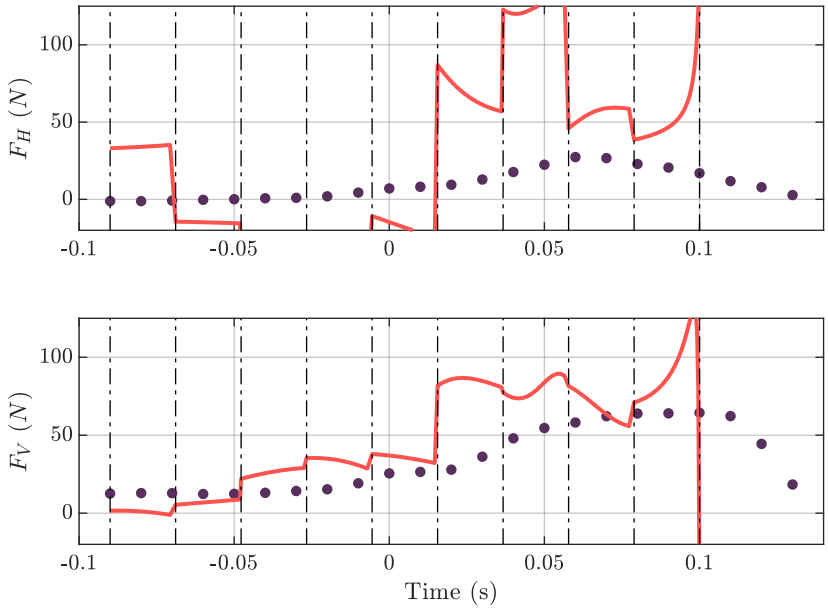


Figure 5.29: Dynamic optimisation ground reaction forces with 10 nodes.

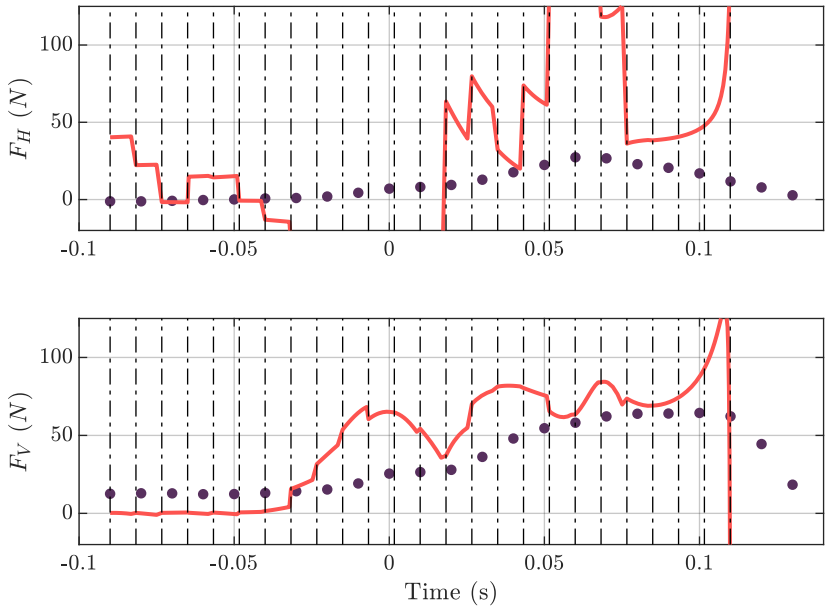


Figure 5.30: Dynamic optimisation ground reaction forces with 25 nodes.

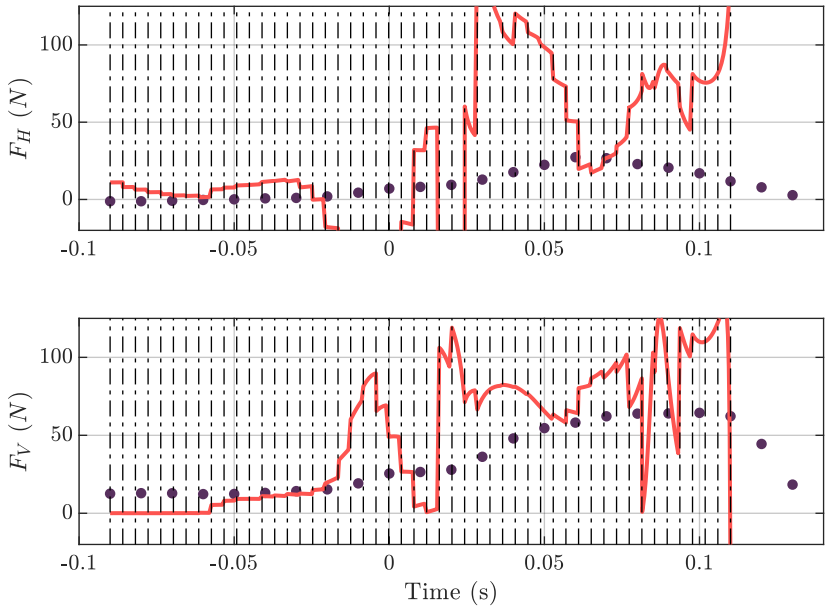


Figure 5.31: Dynamic optimisation ground reaction forces with 50 nodes.

5.5 Discussion

The number of nodes used in the optimisation appears to follow the principals of function fitting. The more nodes that were used in the optimisation, the more the solution became over fit to the problem and the information between nodes became noisy. The single shooting implementation, with 2 nodes, did not find a meaningful solution. Constraining the duration of the trajectory for the single shooting implementation to be 0.2 s resulted in an infeasible solution as well. With 3 and 4 nodes, the control signal had enough freedom to produce a meaningful jumping motion. As the number of nodes was increased further, solutions were found which violated the constraints between nodes and the trajectories became less smooth. Between 25 and 50 nodes, the system dynamics were split into small enough sections that changes in state were mostly linear. At this point, the solutions more reliably adhered to the constraints and the solutions became meaningful.

Under the assumption that a jumping system cannot hold onto the ground and pull itself with a negative vertical force, the take-off event is defined by the vertical ground reaction force becoming less than zero. This occurs in the 5, 10 and 25 node solution between nodes, meaning that the trajectory would be invalid in practice even though no constraints were violated at the nodes during optimisation.

The torque bounded dynamic optimisation is not representative of a biological system as there were no constraints on actuator powers or modelling of muscle force response times. Because of this, the results differ from the experimentally measured data. This is especially noticeable in the ground reaction forces, with the vertical reaction force being generally larger in the optimisation results than in the experimentally measured data.

The control signals produced by the optimisation vary considerably and their variation increases with the number of nodes as more opportunities for change are introduced. These instantaneous changes in control input lead to discontinuities in the ground reaction force. This implies that a step function for the control input is not suitable for biomechanics applications. However, in a robotics setting, where motor response is virtually instantaneous, this may be a representative result.

There is no step change in the control input at the final node of the trajectory, yet all of the solutions show a sharp change in the vertical ground reaction force at the end. This is due to the configuration reached by the system at this stage in the jump. The joint angles are all close to $\frac{\pi}{2}$, meaning that the segments in the leg are becoming aligned in the vertical direction. In this configuration, the ground reaction force supporting the applied torques is horizontal. This is seen in the sharp increase in the horizontal reaction force in the solutions.

The intellectual investment required to implement a dynamic optimisation is significant. Papers and books describing the methods [84], [85] often use a lot of mathematical notation. Dynamic optimisation methods make use of algebra, calculus, dynamic programming, nonlinear programs, and computer programming. A good quality dynamic optimisation implementation would alleviate many of these intellectual requirements. However, such tools are

not readily available at the time of writing. Programs such as fmincon, [38], SNOPT [75], SciPy's "optimize" [76], and IPOPT [77] provide non-linear program solvers. It is required of the user of such methods to transcribe their dynamics problem into a non-linear program to then be solved. Thus, the user must still be versed in algebra, calculus, the formulation of non-linear programs, and computer programming.

The computation time for the dynamic optimisation method increases non-linearly with the number of nodes. The trend is not exponential. However, 9 hours to complete the optimisation with 50 nodes is a significant amount of compuational cost. This is also similar to the times reported by Anderson and Pandy [15] after compensatory adjustment for CPU differences. The computation time was much larger than the time reported by Bishop et al. [58], for a single optimisation run in their work. This discrepancy in computation time is likely due to the difference between the direct collocation method used by Bishop et al. and the multiple shooting method used in this work. Multiple shooting integrates the dynamics of the model between nodes whereas direct collocation approximates the dynamics using a given order spline between nodes. Direct collocation is much faster to compute while multiple shooting generally provides better accuracy in its results.

Learning to use a method which takes a long time to produce results, such as dynamic optimisation, requires the student to have a strong understanding of what will happen when they make changes to any part of the transcription method, objective function, constraints functions etc which is not always possible, especially when the problems are non-linear and potentially chaotic.

The phenomenon described in Section 5.3.1 was seen in the results of the dynamic optimisations, particularly in the 50 node optimisation. This lead to unrealistic motions being found by the method. The unrealistic motions also corresponded to torque trajectories featuring large, sudden changes in the applied torques. It could be the case that the ability of the optimisation to vary torque so drastically enabled the realisation of these unrealistic results. A future study may consider constraining the applied torques either by inclusion of a muscle model or by enforcing limits to the gradients of the torque signals.

5.6 Conclusions

Dynamic optimisation not only enables its user to consider multiple aspects of a jumping problem, it forces them to. Without proper constraints, the methods can converge to results which are undesirable and unpredictable.

The intellectual investment required to use dynamic optimisation is made significant by the large computational cost of the method. A user of the method cannot easily employ a trial and error approach with the methods as they can take a long time to produce results.

Unnatural motions produced by the dynamic optimisation may be due to poor selection of constraints for the optimisation, or due to poor selection of the model being optimised. Future

work may consider optimising bounded gradients for the applied torques or constraining the gradients of the optimised torque signals.

The results of this study imply the existence of a "sweet spot" in the complexity of a dynamic optimisation's transcription. With sufficiently few nodes in the trajectory the results obtained may be sensible and close to realistic. The inclusion of additional nodes seems to lead to a decline in the quality of results, with constraint violations occurring more frequently between nodes. With a sufficiently large number of nodes, the optimisation is forced to satisfy constraints between nodes as the dynamics are essentially linearised, producing more realistic results.

Chapter 6

Reinforcement Learning

Reinforcement learning methods pose an interesting prospect for biomechanics research as they are model-free optimisation methods. This is particularly useful in high complexity applications, as researchers often look to dynamic simulation software packages [31], [32] which simulate dynamic systems without exposing the model equations. Reinforcement learning has been popularised by demonstrations from research companies such as DeepMind and OpenAI [86], [87]. However, these headline-making successes are generally backed by significant computational resources which may not be available to some academic researchers.

Other model-free optimisation methods include evolutionary algorithms, however, these are not included in the scope of this work. Evolutionary algorithms may be used to optimise a parameterised policy or a trajectory and their implementation is generally similar to reinforcement learning with both methods having the potential to find global optima to complex problems. It is claimed that "[Reinforcement learning] methods able to take advantage of the details of individual behavioral interactions can be much more efficient than evolutionary methods in many cases" [22]. Thus, evolutionary algorithms are left out of the scope of this work.

The work in this chapter assesses the application of Deep Deterministic Policy Gradients (DDPG) [23] for predictive modelling of planar jumping systems. As the method is novel in its application to jumping dynamics problems, this work seeks to evaluate what insights may be gained through its application.

DDPG is a state of the art reinforcement learning method which was anticipated to be suitable for jumping dynamics problems. The DDPG algorithm was applied to the *Numida meleagris* jumping model used in previous chapters of this thesis. The method and results were assessed on predictive power, computational cost, and intellectual investment.

6.1 Why DDPG?

One of the factors which influenced the selection of DDPG was that the method produces a deterministic, continuous policy. In the context of jumping, the result of the DDPG algorithm may be the deterministic function:

$$\boldsymbol{\tau} = \pi(\boldsymbol{q}). \tag{6.1}$$

Note that the output of the policy could be used as any aspect of the model e.g. $\ddot{q}$ or target joint angles for a PID controller. A deterministic policy is an important feature for studying the motions produced by the policy. Many reinforcement learning algorithms optimise a stochastic policy, which outputs a probability distribution for actions to take in a given state. This output of probability distributions from the policy is essential to the derivation of the policy gradient method [88] which is used in state of the art methods such as Trust Region Policy Optimisation [24] and Proximal Policy Optimisation [25]. This is also a requirement in some Actor Critic methods, such as the Soft Actor Critic algorithm [26].

The policy used in DDPG also outputs continuous actions, this is especially useful for fine motor control using torques. For different jumping models such as those controlled by activation signals, a discrete policy may be suitable. An example of a reinforcement learning algorithm which provides a discrete policy is Deep Q-Network [89].

An important distinction between reinforcement learning methods and the other optimisation methods considered in this work is that reinforcement learning methods seek to produce a globally optimal policy whereas static and dynamic optimisation methods produce optimal trajectories, taking the system from an initial state to a desired state. Producing a policy, or controller, instead of a trajectory may allow the researcher to evaluate optimised system behaviours from various starting configurations. The parameterisation of the optimised policy may also provide insight into the decision making behaviours, i.e. what state information most influences the actions output by the policy. For the case of producing a single trajectory, the reinforcement learning method may be used by starting each episode form the same state. This practice would not work with dynamic programming, or off-policy, methods of reinforcement learning such as value iteration.

6.2 Pitfalls

This section identifies some of the difficulties, and their potential workarounds, which arose when applying reinforcement learning methods to jumping dynamics problems.

6.2.1 Catastrophic Failure in Jumping Problems

Sampling data from a high dimensional problem is a considerable hindrance to the use of reinforcement learning methods in dynamics problems. Jumping problems in particular exhibit a "catastrophic failure case" (not to be confused with catastrophic forgetting). Catastrophic failure occurs when the system enters a state from which the system cannot feasibly recover. In jumping, this typically involves the system falling over and is an especially easy case to produce.

The random exploration strategies used in most reinforcement learning methods are likely to result in the system falling over during exploration and being unable to progress. Reinforcement learning methods are capable of overcoming failure cases as the optimisation penalises

the system falling to the ground. Any actions which lead to negative rewards, i.e. the system falling over, are filtered out of the policy and what remains are actions which do not lead to the system falling over. However, in jumping, falling over is a large possibility and ruling out all the actions which lead to falling over is no small task. Learning all the ways not to perform a task before learning the correct way is not an efficient process.

The difficulty with jumping is that it is considerably easier for an attempt to fail than it is to succeed. Sampling may be insufficient for finding solutions to jumping problems and so further measures may need to be taken when applying reinforcement learning methods to problems which exhibit such catastrophic failure cases. The following subsections will describe two potential means of overcoming this problem.

6.2.2 Point Mass Example

The following sections will provide examples based on the following simplified system. Consider a point mass jumping system which is at rest on the ground. The mass is actuated by a vertical thruster which produces a discrete acceleration on the mass of either a = -1, 0 or 1 ms^{-2}. The mass is attached to a frictionless, massless rod of length 0.5 m with a stopper at the top, and the only way the system can jump is by reaching the top of the rod and pulling the rod off the ground. The state of the system, $\boldsymbol{x}$, represents the vertical height above ground and the vertical velocity of the mass. If the thruster accelerates the mass upwards continuously it would take 1 s for the system to take-off with a velocity of 1 ms^{-1}. This is considered as the only feasible jumping motion for this example model. While it is possible for this system to take-off with a lower velocity than 1 ms^{-1}, a system with increased complexity would have a significantly smaller set of feasible motions compared to non-feasible ones than this simple system. Thus, for illustrative purposes, the margin of success in this problem is decreased.

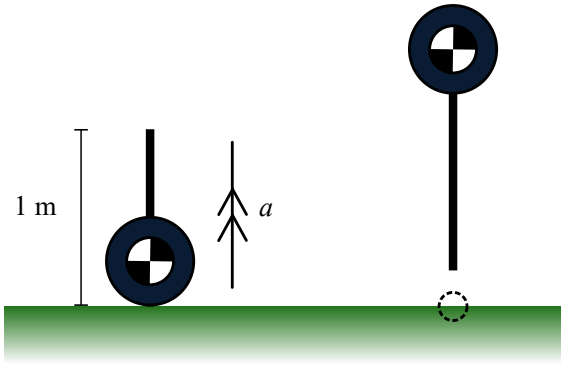


Figure 6.1: The point mass system in its starting state (left) and after a successful take-off (right). a represents the acceleration due to the applied control action.

6.2.3 Frame Skipping

Episodes are typically limited in their duration to ensure the method interacts with each stage of a task often. During an episode, the method will have a number of opportunities to select a control action. The period of these decisions is defined by the "frame skipping" parameter of the algorithm [23], [24], [89], [90]. Frame skipping defines the number of frames, or time steps, that a chosen action is applied for. Problems with high frequency responses will require a low frame skipping value while slowly changing systems lend themselves to higher frame skipping values.

Frame skipping is often used in dynamics problems as simulations tend to use small time steps; if reinforcement learning methods selected a pseudo-random action at every time step of the simulation the chances of any meaningful trajectories being generated would be slim. However, a high frame skipping value may lead to the actions being selected too infrequently leading to oscillations, overshooting or generally insufficient control resolution to complete the task.

To demonstrate the significance of frame skipping, consider the point mass system described in section 6.2.2. For example's sake, consider that the system is simulated at a time step of 0.02 s, thus a 1 s attempt includes 50 steps of simulation. A controller operating with no frame skipping would be required to provide a set of 50 consecutive decisions. Under the assumption that the only feasible jumping motion requires continuous positive acceleration, the controller has a probability of $\frac{1}{3}$ of choosing the correct action at any given step in the simulation. Without frame skipping, the controller would have to correctly choose an action 50 times in a row which is a probability of $(\frac{1}{3})^{50} = 1.4e-24$, significantly lower than the probability of winning the lottery.

Frame skipping holds a chosen action for a set number of frames, reducing the number of decisions required from the controller and thus increasing the probability of a successful attempt. This effect is shown in figure 6.2. Note that this is an extremely simplified illustrative example, in reality the probability of success is not so clearly defined and the number of frames skipped may have a very different relationship with this probability. Furthermore, in jumping it is common to measure success using the velocity of the system at take-off and so any attempt which achieves take-off may contribute to the optimisation and increase the policy's probability of selecting the correct action at each step. While this is the case, this example system does not include a catastrophic failure case; one wrong action will not cause the system to fall over and the attempt can easily recover from a mistake. Non-linear dynamics problems with high dimensional state spaces and multiple control actions are difficult to reduce to a small number of decisions due to the dimensionality of the action space. As such, frame skipping may only increase the probability of success slightly, meaning that further methods are required to ensure sampling finds success.

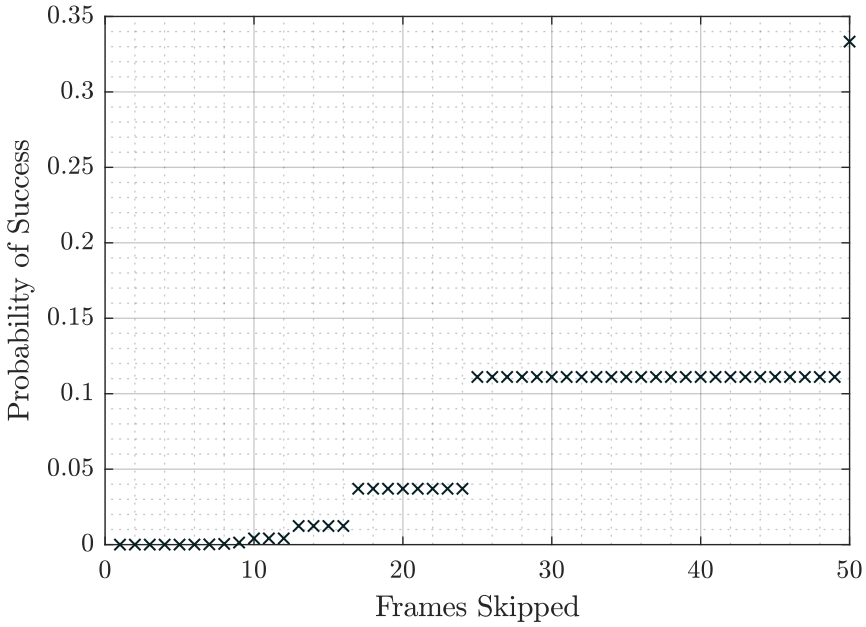


Figure 6.2: Plot of the effect of frame skipping on the probability of a successful attempt at the point mass jumping problem. Steps on the plot are due to the duration of the episode not dividing perfectly into each action, e.g. 30 frames skipped and 40 frames skipped will both require 2 actions to be chosen.

6.2.4 Reward Shaping

Another practice used in reinforcement learning to improve the chance of success is reward shaping. Reward shaping is the act of adding incentives to a reward function which encourage desirable algorithm behaviours, such as guiding the exploration of the reinforcement learning method. Reward shaping can be used to guide the algorithm's exploration of the problem by continuously providing information on how good each action was.

Using the point mass jumping problem mentioned previously, the objective is to maximise take-off velocity, which in turn requires the system to take-off. Assuming that take-off only occurs when the system has accelerated upwards through the entire attempt the reward would be:

$$r = \begin{cases} 1, & \text{if } \boldsymbol{x} = \boldsymbol{x}_{goal} \\ 0, & \text{otherwise} \end{cases} \tag{6.2}$$

Where r is the reward received and $\boldsymbol{x}_{goal}$ is the goal state of the system which in this simplified example would be $(0.5\,m, 1\,ms^{-1})$.

This reward will not be discovered by a sampling method unless a successful take-off attempt is performed. Until the reward is sampled, the optimisation algorithm will have no gradient information to follow and will effectively continue behaving randomly. One method of reward shaping in this scenario would be to providing a reward which is inversely proportional to the distance between the current state and the goal state, thus encouraging any actions which

move the system closer to the goal.

$$r = \begin{cases} 100, & \text{if } x = x_{goal} \\ (x_{goal} - x)^{-1}, & \text{otherwise} \end{cases} \tag{6.3}$$

As the reinforcement learning method collects samples it is highly likely to experience the shaped rewards leading to the policy being optimised to encourage actions which move the system towards the goal state.

This introduction of additional reward signals can become problematic as the task description has been made more complicated. In this simple example it may not be a significant issue, however, if the jump required a countermovement of the mass (i.e. the mass needed to be moved down first and then up) then this shaping would discourage the mass from moving down and cause the reinforcement learning method to fail. Another potential failure case with this example of reward shaping would be for the policy to move the mass as close to the goal state as possible without completing the attempt, collecting the shaped reward signal and ignoring the overall objective of the task.

Shaping a reward function may be non-intuitive for complex tasks such as jumping, during which the intermediate objectives of the motion may be unclear. Shaped reward functions add complexity to the optimisation and make it difficult to anticipate the optimal behaviour for the shaped reward function. This can lead to undesirable results in which the optimised policy focuses on parts of the reward function which are not directly related to the overall task objective. For example, a running model may frequently fall over during training and so it may seem reasonable to shape the reward function by including an incentive to keep the body at a fixed height above the ground. A good policy for the shaped reward function may be to stand still and collect rewards for the body being at the correct height. It may be argued that such qualities exist for any optimisation algorithm and that it is the responsibility of the user to anticipate or correct such discrepancies. The problem is that describing a task with a scalar function is already considered difficult [79] without the requirement to add exploratory guidance to the description as well. Optimal control allows communication of task objectives and constraints via both the objective function and constraint functions. In reinforcement learning, the only means of communication is through the reward function.

By including incentives for exploration in the reward function, the function's use in describing an optimal motion is compromised. Often in optimisation, the objective forms part of the hypothesis by mathematically stating how the task is best carried out. With terms included in the reward function specifically to guide exploration, the optimisation is forced to balance desirable optimal behaviours with those that improve the exploration of the problem.

One possible workaround to the problems related to reward shaping is to gradually remove the shaping from reward functions based on the performance of the current policy. This is known as a "curriculum". A curriculum is difficult to implement as performance is based on a scalar value comprising potentially many shaping components. Knowing when such a value is sufficient to change the curriculum may become a complex task in itself, potentially requiring

examination of policies at multiple stages and reward scores throughout the optimisation.

6.3 Method

An implementation of DDPG provided in the MatLab Reinforcement Learning Toolbox [38] was used to optimise a policy which output torque values to control the *Numedia* jumping model. Training episodes were initialised from a single starting state:

$$\boldsymbol{s}_0 = [2.769, 0.596, 2.583, -0.0705, 1.5474, -1.9000, -1.8751, -1.8947]^T. \tag{6.4}$$

Episodes were run until a termination criteria was met, or the episode reached 500 steps in simulation (0.5 s in real time for the jumping motion). Training was run for a maximum of 20,000 episodes, or between potentially 20,000 and 10,000,000 simulation steps depending on the policy performance. The reward function to be maximised was:

$$r_t = \max(0, 10\,\boldsymbol{v} \cdot \boldsymbol{\rho}_t). \tag{6.5}$$

Where the vector $\boldsymbol{v}$ represents the target take-off velocity described in equation (5.3). The vector $\boldsymbol{\rho}_t$ represents the total momentum of the jumping system at time t. This reward uses shaping at every step to guide the optimisation into maximising the system's momentum in the desired take-off direction, as opposed to including a single reward based on the take-off velocity of the system.

Many optimisations include a penalty on the control action to encourage energy efficient motions or to discourage excessive actuator forces. In this study, no such penalties were included as they produced a local minimum in the optimisation wherein the jumping system moved to terminate the episode immediately so as to not accumulate the negative rewards. A reward of -1 was given for a step which was terminated for either of the following criteria:

- intersection of adjacent segments
- segments touching the ground (other than the foot).

An episode was also terminated if the ground reaction force reached zero or less as this would imply the system has broken contact with the ground, however, this termination was not discouraged with a penalty as it is a requirement of jumping.

The following components of DDPG were considered when collecting results with the method:

- actor,
- critic,
- hyper-parameters.

The actor is used to select actions based on the state of the system and the critic is used to predict the expected return of such actions in their context. Both the actor and critic may be parameterised by any differentiable function, however, they are generally represented using deep neural networks. DDPG uses a target actor and target critic to update the actor and critic. The target actor and critic are equivalent parameterisations to the actor and critic, which are updated using the deterministic policy gradient. The actor and critic are then updated using their respective targets in order to smooth the updates and improve the stability of the method. Factors to consider about the actor and critic include:

- optimisation method used to update the policy and critic, e.g. Adam,
- step size of the optimisation updates for both the policy and critic,
- structure of the neural network, including number of layers and hidden units

The hyper-parameters are features of the DDPG algorithm which include:

- frequency and size of target updates,
- discount factor, which weights the influence of future rewards on current decisions,
- mini-batch size for stochastic gradient descent updates
- size of the experience buffer
- parameters of the noise model used to generate exploratory actions

Often in reinforcement learning works, the algorithms use different noise models for their exploration strategies. Given more time, this would be a point of further study as the method of sampling may have a significant impact on the success and stability of reinforcement learning methods. In the MatLab implementation of DDPG the exploration strategy is limited to the Ornstein-Uhlenbeck noise model, it would be interesting to compare the performance of DDPG when using different exploration strategies.

The parameters of the actor and critic are initialised to random values. Exploratory actions are created by adding stochastic values from the noise model. The experience batches used to update the target actor and critic neural networks are selected at random. These stochastic components of DDPG, and many reinforcement learning methods, mean that the results produced by the method may vary considerably depending on the seed used to create the system's random number generator.

While the hyper-parameters and parameters of the actor and critic have clear meanings, the selection of ideal values is not a well defined process. Many studies will perform a grid search of hyper-parameters [65] to obtain the best performance on the particular problem. Due to the variation in results, multiple seeds were also tested for each set of hyper-parameters investigated in this study. Reinforcement learning methods generally require a large number of samples to produce meaningful results. All of these factors contribute to a considerably large computational cost for reinforcement learning methods.

6.4 Results

125 optimisation runs were recorded during the hyper-parameter tuning process. The following chart shows the distribution of performances for those 125 optimisation runs. 44.8% of the runs resulted in a policy which was unable to jump.

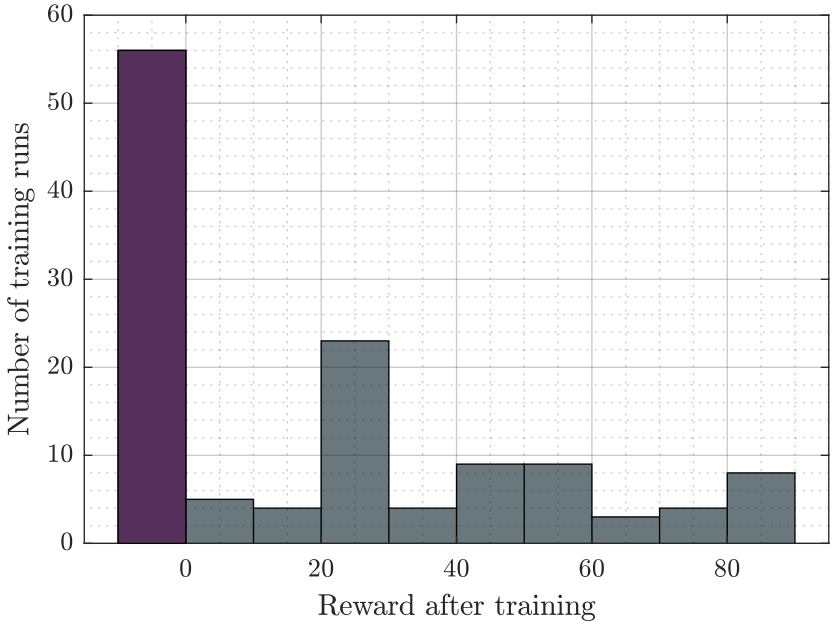


Figure 6.3: The rewards scored by the optimised policies during hyper-parameter selection. Sets of hyper-parameters were tested 25 times each with a total of 125 test runs. The purple bin on the left of the plot represents failed jumping attempts.

Hyper-parameters were selected based on the set of results containing the best performing run. 25 runs were performed using the hyper-parameters provided in appendix C. Four sample runs were taken from the collection and are presented in detail here.

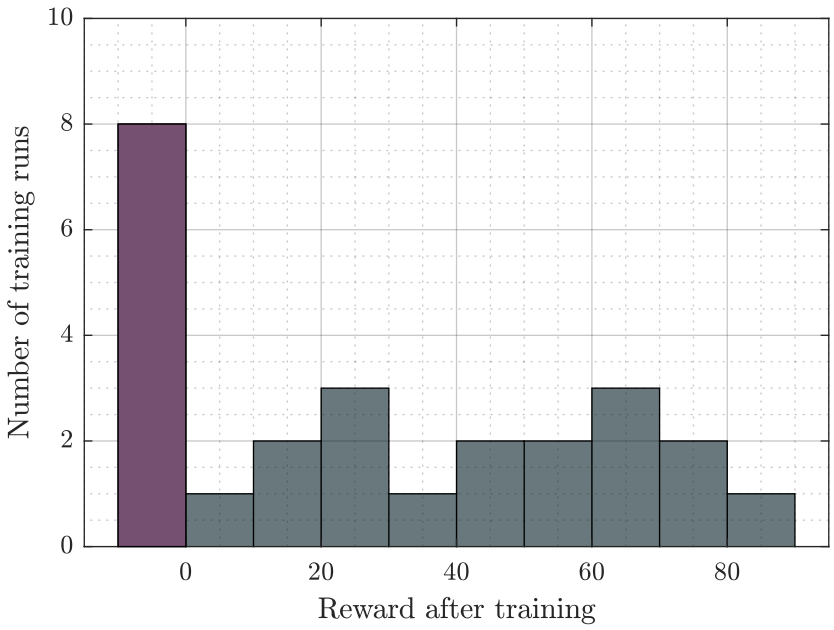


Figure 6.4: The rewards scored by the optimised policy for each random seed run. The purple bin on the left of the plot represents failed jumping attempts.

The highest scoring run achieved a total reward of 82.0265. The results presented in this section are the runs which obtained rewards of:

$$r_1 = 82.0265,$$
$$r_2 = 78.8027,$$
$$r_3 = 48.5575,$$
$$r_4 = 32.2711.$$

6.4.1 Computational Cost

The computation times for the final 25 runs were collected. The average time for the 25 runs was 1005 seconds and the standard deviation was 292 seconds. The computation times ranged from 357 s to 1605 s. The variation in these computation times is due to the performance of the policies for each run. As each run was limited to a maximum number of episodes, a policy which fails every episode immediately does not require as many simulation steps to be computed as a policy which does not fail on the first step.

6.4.2 Joint Angle Trajectories and Poses

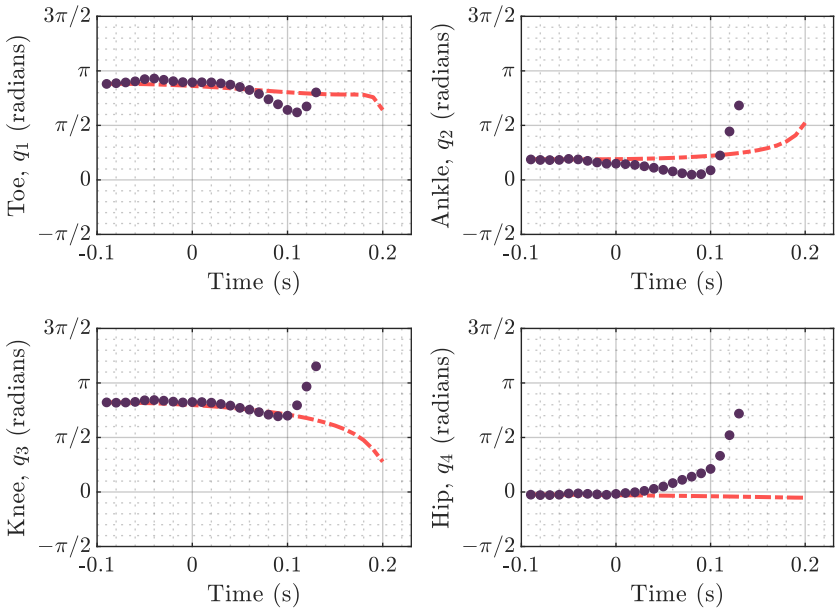


Figure 6.5: Joint angle trajectory for the DDPG optimisation run which obtained a reward of 82.0265, the highest reward.

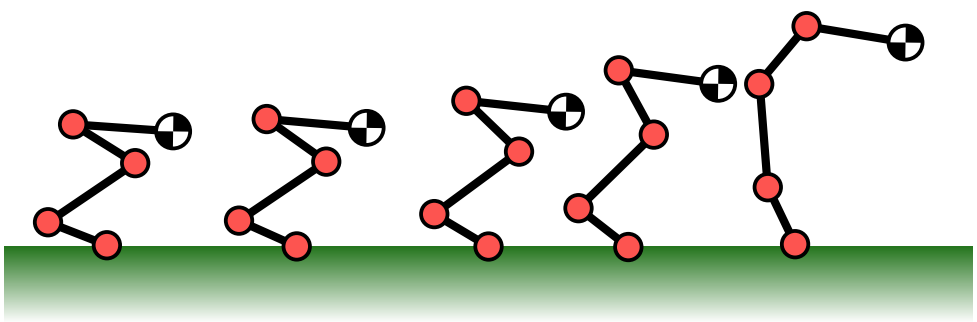


Figure 6.6: Poses of the system at equal intervals throughout the resulting trajectory for the DDPG optimisation run which obtained a reward of 82.0265, the highest reward.

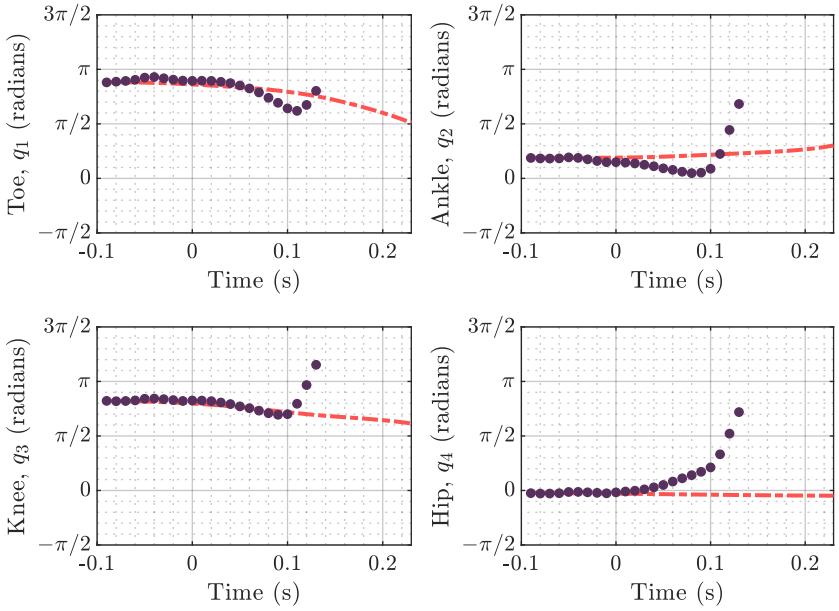


Figure 6.7: Joint angle trajectory for the DDPG optimisation run which obtained a reward of 78.8027, the second highest reward.

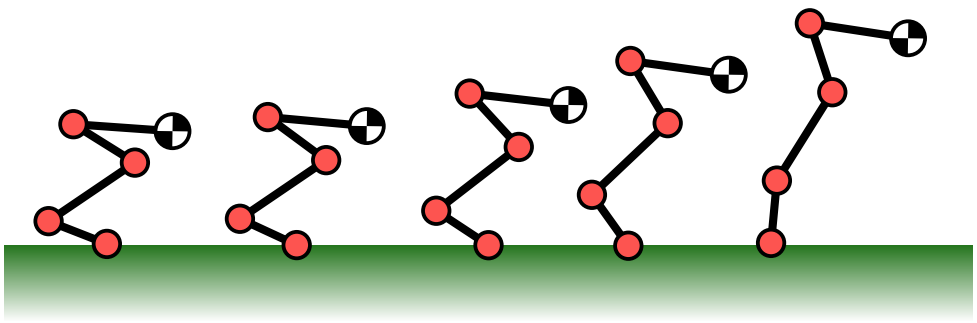


Figure 6.8: Poses of the system at equal intervals throughout the resulting trajectory for the DDPG optimisation run which obtained a reward of 78.8027, the second highest reward.

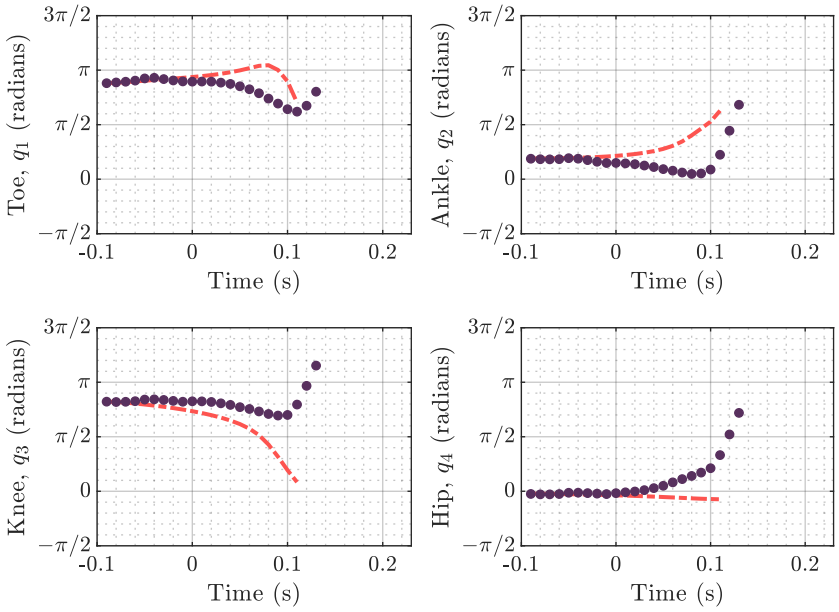


Figure 6.9: Joint angle trajectory for a randomly selected DDPG optimisation run which obtained a reward of 48.5575.

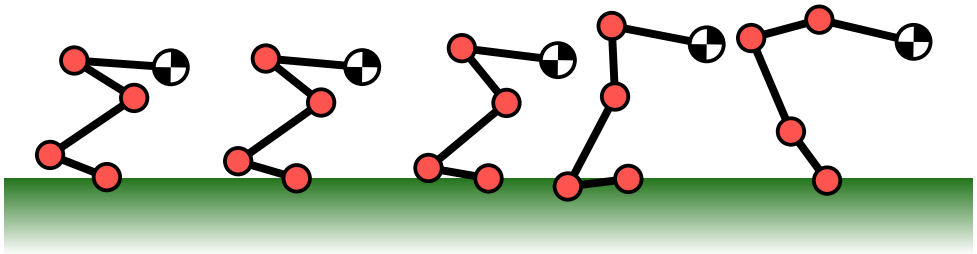


Figure 6.10: Poses of the system at equal intervals throughout the resulting trajectory for a randomly selected DDPG optimisation run which obtained a reward of 48.5575.

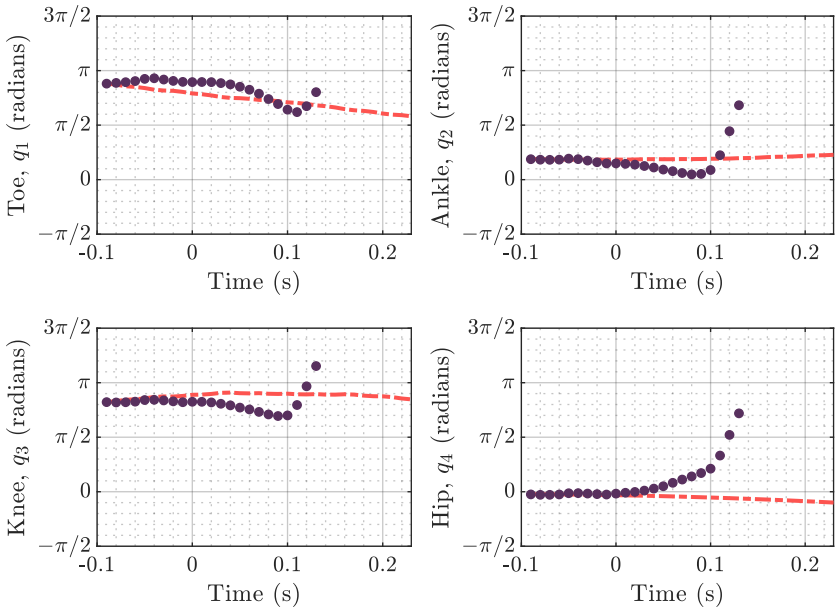


Figure 6.11: Joint angle trajectory for a randomly selected DDPG optimisation run which obtained a reward of 32.2711.

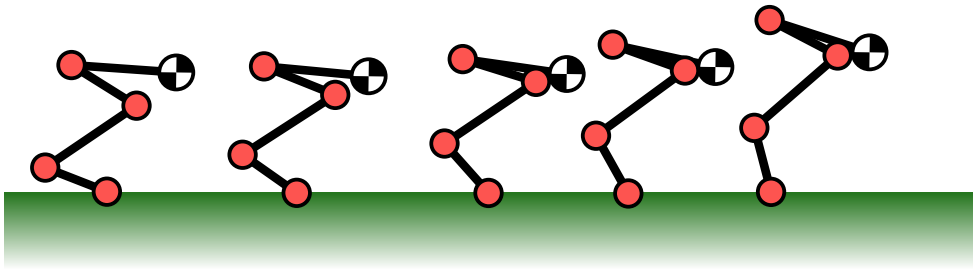


Figure 6.12: Poses of the system at equal intervals throughout the resulting trajectory for a randomly selected DDPG optimisation run which obtained a reward of 32.2711.

6.4.3 Joint Torques

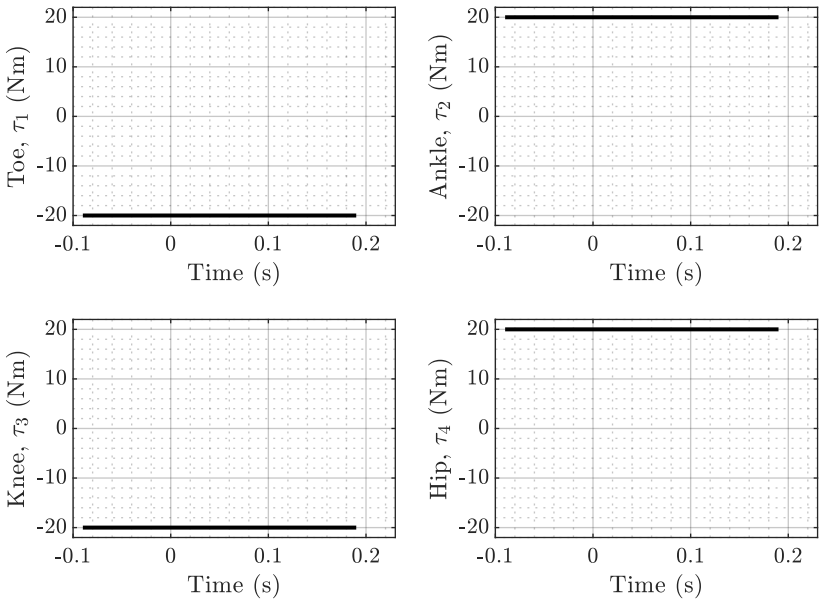


Figure 6.13: Torque trajectory for the DDPG optimisation run which obtained a reward of 82.0265, the highest reward.

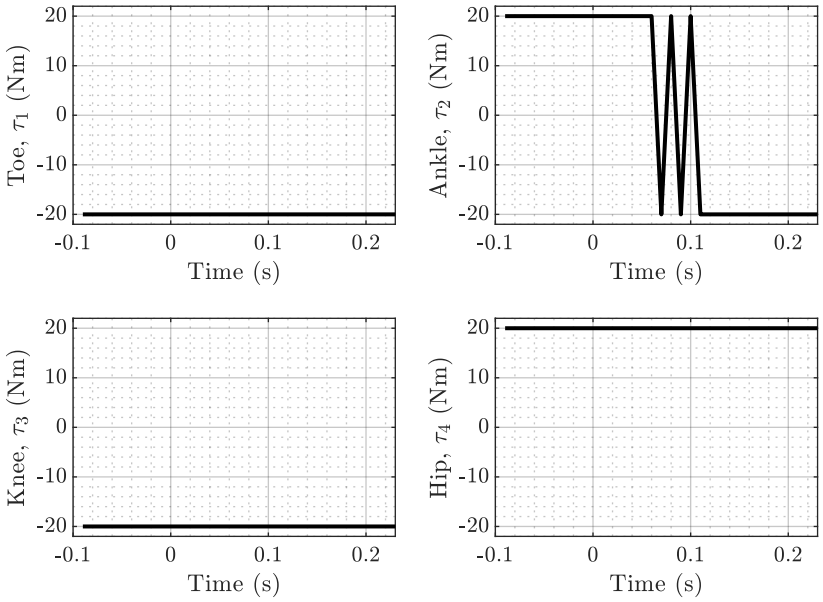


Figure 6.14: Torque trajectory for the DDPG optimisation run which obtained a reward of 78.8027, the second highest reward.

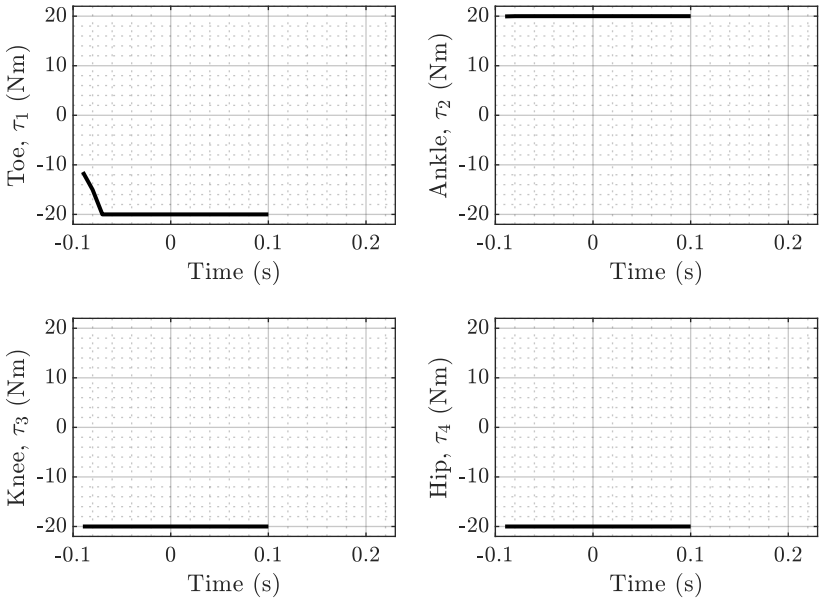


Figure 6.15: Torque trajectory trajectory for a randomly selected DDPG optimisation run which obtained a reward of 48.5575.

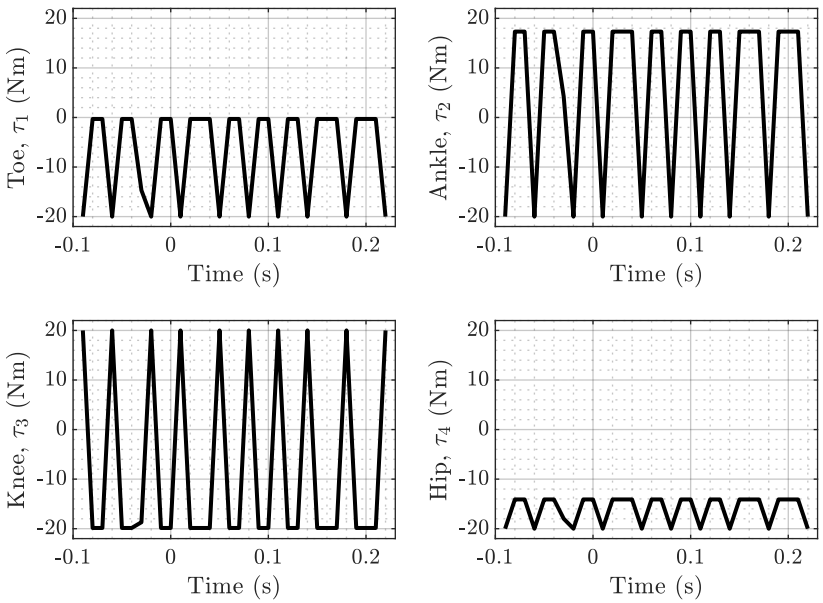


Figure 6.16: Torque trajectory trajectory for a randomly selected DDPG optimisation run which obtained a reward of 32.2711.

6.5 Discussion

The highest scoring result depicts a jumping motion in which the knee joint of the system is inverted towards the end of the motion. Further constraints could have been included as termination criteria for the training episodes although it is anticipated that these additional termination criteria may result in a greater portion of the optimisation runs failing.

The second highest scoring result depicts a realistic jumping motion. The torque trajectory for the second result is very similar to the first. The torque signal for the ankle joint includes a few rapid changes in the middle of the second jumping trajectory. Note that the policy optimised by the DDPG algorithm outputs continuous actions, the bang-bang behaviours seen in the results are an outcome of the optimisation, not the constraints or model.

Both of the top scoring results have a longer duration than the experimentally measured jumping motion. This is likely due to the use of reward shaping, which may have encouraged a long duration which would collect more shaped rewards than a shorter duration jump would have.

The quality of the results produced by DDPG were mixed. The second highest scoring result is reasonably realistic. However, it lasts considerably longer than the jump which was measured by Henry et al. [6] for the same system.

While optimal control and reinforcement learning share many features, such as their derivations from Bellman's principle of optimality, the two groups of methods are often kept separate from each other. The different notation and terminology of optimal control and reinforcement learning may hinder the transfer of intellectual knowledge in one subject area to the other despite the two method groups sharing transferable skills, such as experience with dynamic systems, objective functions, gradient descent, and optimisation of trajectories.

Reinforcement learning methods are becoming more widely adopted by commercial research packages, such as MatLab [38]. However, the methods are not quite at the stage to be used as tools in the way other methods might be. Using a reinforcement learning method requires a considerable level of knowledge regarding the workings of the method. This is because current methods use hyper-parameters which do not yet have universal rules for their tuning process. For example, the step size of an optimisation method may be universally tuned: increase the step size to speed up the optimisation process and decrease the step size if the optimisation diverges. This applies to the learning rate of reinforcement learning methods as well. Hyper-parameters which are specific to reinforcement learning methods are not so general. For example, the discount factor determines how the return of an action is calculated and affects the weighting of immediate to long term rewards in the optimisation; the selection of the discount factor affects the period of time used to determine how valuable a given action was. For a user to select an appropriate discount factor they must understand this concept and apply it to their specific problem, considering factors such as how far into the future consequences will occur for a given action. The discount factor encapsulates a complex interaction between the algorithm and problem and does not have a general rule for

its selection. Increasing the discount factor can lead to high variance in the return if actions do not have lasting effects on the system, or it may lead to the algorithm correctly accounting for the long term effects of actions.

Reinforcement learning methods use random selection in multiple areas of their implementation. While the random sampling used in many reinforcement learning methods enables their global optimisation abilities, it also leads to variety in their results and can render them unreliable. This effect was demonstrated in the results of the hyper-parameter search 6.3 during which 44.8% of the runs failed to obtain any positive rewards and in the final runs during which 32% of the runs failed to obtain any positive rewards. This variability in performance means that the method can be unreliable and may become time consuming to use as multiple runs are generally required to filter out failed attempts.

The issue of unreliability is exacerbated by the computational cost of the method. All 25 of the final runs took around 7 hours to complete. The runs used to select the hyper-parameters used in this study were not timed. However, it may be estimated that the 125 runs took $5\times$ as long, approximately 35 hours in total.

This chapter set out to assess the application of DDPG to jumping problems. The method did not provide insights into jumping systems as the results were unrealistic and unreliably produced. Although the work was unable to produce realistic results using the method, the work has shown some potential for DDPG to synthesise jumping motions. However, the potential of the method is arguably outweighed by the costs involved in using the method.

6.6 Conclusions

DDPG may not be an appropriate method for jumping research in its current state due to its unreliability and significant computational cost. Researchers looking to make use of reinforcement learning methods may have to consider methods which produce a probabilistic policy as deterministic policy methods are uncommon in the literature.

The sampling strategies used by reinforcement learning methods to explore and collect data were shown to run into problems when applied to jumping problems. This is because it is very easy for a catastrophic failure to occur during an attempt at jumping. After the failure occurs, the attempt must be abandoned as the system is unable to recover. Jumping motions exhibit extremely high potential for failure throughout their duration, making it difficult for sample-based methods to find success.

Practices such as frame skipping and reward shaping can be used to improve the performance and likelihood of success for reinforcement learning methods applied to jumping problems. However, reward shaping can lead to undesirable results, as the reward function may become too complex to anticipate what optimal behaviours according to the function would be. Furthermore, the inclusion of exploration incentives in a reward function reduces the utility of the reward function in describing the task objective.

Chapter 7

Conclusions

This chapter provides the concluding remarks of the thesis as well as highlighting the novel contributions of the work. The aim of this work was to provide a critical assessment of the use of predictive modelling methods in jumping research. To achieve this aim, work was carried out to explore the use of predictive modelling in jumping literature before investigation of predictive modelling methods was completed using various case studies. The work also set out to assess the predictive power, computational cost, and intellectual investment of predictive modelling methods. Finally, the potential of future work facilitated by this work is outlined.

7.1 Research Aims

The literature review of this thesis found two predictive modelling methods which are not applied to jumping dynamics problems: static optimisation for the resolution of second order kinematic redundancy, and reinforcement learning. The review also identified that analysis is generally applied to models with a single degree of freedom. To be able to provide a critical assessment of the use of predictive modelling methods, this work contributed the novel application of static optimisation and reinforcement learning to jumping problems using a case study with experimentally measured data found in the literature. The application of algebraic analysis to low complexity jumping models is also a contribution of the work as it demonstrates a formalisation of an analysis method which was found in the literature applied to a single degree of freedom model. The work then assessed the application of these methods. Dynamic optimisation was also included as a case study due to its prolific use throughout the jumping literature.

7.2 Research questions

This section provides details of how the research questions have been answered by the work. The research questions are provided here for convenience:

1. What insights can be gained through the application of predictive modelling methods to novel contexts in jumping dynamics problems?

2. How should the fitness of different predictive modelling techniques be assessed?

7.2.1 Research Question 1.

It was demonstrated in Chapter 3 that analysis of model equations can be used to provide insights into the capabilities of low complexity jumping systems through the results in Section 3.2.4.

Analysis of a high complexity model produced insights into the accelerations of the system's body which will maintain dynamic balance of the system during a jumping motion. This analysis also produced the novel dynamic balance ellipsoid, a tool which may be used to investigate the acceleration capabilities of a jumping system while maintaining its balance.

The results of Chapter 4 imply that static optimisation may not be suitable for obtaining insights into biological jumping systems as the results violated the physical constraints of such systems. However, the considerably low computational cost of the method as well as its simplicity are good arguments in favour of further investigation into the method's application in jumping research. It is also noted that while the results in Section 4.4.1 did violate the physical constraints of the system, the results were considerably close to the experimentally measure motion data for the system.

This work failed to produce realistic motions using the multiple shooting method of dynamic optimisation in Chapter 5. However, as many studies in the literature have previously found success with the method, it is considered suitable for obtaining insights into the dynamic motions of jumping systems.

The results of Chapter 6 demonstrate that DDPG is unsuitable for use in jumping research. The method was unreliable, with 32% of optimisations failing to obtain any positive rewards. The high computational cost of the method means that this unreliability is difficult to overcome as performing multiple optimisations is a time consuming endeavour.

7.2.2 Research Question 2.

This work investigated the use of predictive power, computational cost, and intellectual investment as metrics for assessing predictive modelling methods.

The intellectual investment of a method was defined as "The domain knowledge required from the user of the method, in both the problem and the method". This metric was not found discussed in the literature and yet it is a significant consideration for researchers selecting methods for research. This work found intellectual investment to be difficult to quantify, especially when concepts from the different methods overlapped and the learning requirements of new methods diminished. Despite this, it was deemed that analytic methods carry a low intellectual investment compared to the other methods assessed in this work. The intellectual investment of analytic methods is not fixed; it increases as the model complexity is increased.

Static optimisation was determined to require a moderately low intellectual investment as the method may be used as a tool providing the user is given derivations of the system's kinematics. With the introduction of temporal effects and gradient descent optimisation, dynamic optimisation requires a significant amount of intellectual investment. Reinforcement learning methods require the greatest intellectual investment of the methods assessed in this work as they are not usable as tools without strong knowledge of their workings. The hyper-parameter tuning process is not suited to a trial and error approach and requires the user to understand the effects of each parameter in order to properly tune.

The predictive power of a method was defined as "How flexible the method is, the range and utility of the results it provides, and the limits in model complexity for the method". This metric was suitable for this study as it was quantifiable. The results of each study were compared with experimentally measured data to determine the utility of the results. Analysis was found to be limited in its application to systems with many degrees of freedom as the computational cost of derivations using a computer algebra system increase exponentially; derivation of the equations of motion for a system with 50 degrees of freedom was estimated to take 455 days of computation time. Static optimisation was found to be rigid in the sense that it cannot be used to investigate deficient systems. Dynamic optimisation and reinforcement learning are both flexibly applied to any jumping system.

The computational cost of a method was defined as "The time it takes for a commercial, high spec desktop computer to carry out the method". This work recorded the times taken to carry out each method. Analysis of model equations operated in the order of 1 - 10 seconds. Static optimisation operated in the order of 0.01 - 1 seconds. Dynamic optimisation operated in the order of 1 - 10,000 seconds. Reinforcement learning operated in the order of 100 - 1000 seconds. This metric is easy to interpret and provides a good basis for comparison between predictive modelling methods.

This work did not succeed in assessing the intellectual investment of predictive modelling methods. It is proposed that this metric be investigated by means of a survey or similar method. The computational cost and predictive power metrics were useful and appropriate for the study and should provide useful information for researchers looking to select a predictive modelling method for their own use.

7.3 Future Work

Future work inspired by this thesis may seek to investigate the intellectual investment of predictive modelling methods by surveying researchers who use such methods, asking questions related to the educational backgrounds of the researchers and the time they spent studying the methods they use. Undergraduate students would also be an interesting source of data for the intellectual investment of predictive modelling methods.

Both the gear constrained models and the static optimisation method provide a method of producing jumping trajectories using a set of parameters which describe the desired motion of

the body of the system. The parameters could be optimised by evaluating properties of the resulting trajectories, such as total work done by joint actuators. As the parameters describe the body motion directly, constraint criteria, such as take-off velocity, may be efficiently obtained from the parameters themselves. The difficulty of this may be similar to the single shooting problem in that the path taken by the system may not be smooth in relation to the parameters describing it, although it is not anticipated that this would be an issue as the parameters describe a significant portion of the trajectory by constraining the motion of the body, thus the leg has limited reasonable response especially when static optimisation is used to minimise the joint accelerations.

It would be useful for a body of work to produce the algebraic equations of motion for a variety of jumping systems and provide them in a repository, preferably in multiple programming languages, such as MatLab and Python.

The dynamic balance ellipsoid provides multiple metrics for a jumping system which may be considered in objective functions during optimisation of either a trajectory, or the structure of a jumping system. This work has been done for manipulators already [29]. An example of this would involve optimising the structure of a jumping system over a set of configurations which describe typical jumping motions of the system, with the objective that the direction of the major axis of the dynamic balance ellipsoid aligns with the required acceleration of the body during the jumping motion. The objective function might also include the maximisation of the major axis to ensure the system has a mechanical advantage when using the leg joints to accelerate the body.

Further study using the dynamic balance ellipsoid could investigate the variation of feasible body accelerations throughout nominal jumping trajectories. This may provide insights into the acceleration dexterity of jumping systems and answer questions pertaining to the capabilities of jumping systems.

Optimisation using kinematic models is considered to be much faster than optimisations using dynamic models [53]. The methods described by Geoffroy et al. in [53] and [91] would be interesting to investigate in their application to the jumping models used in this work.

The initial results of static optimisation in Chapter 4 show promise for the method's success. Further work may investigate the limits of the method to a greater extent. The use of static optimisation in this work was limited by the requirement of a starting state for the system as well as the need for an acceleration trajectory for the body of the system. Experiments with static optimisation could be used to explore sets of feasible starting states and body acceleration trajectories and potentially characterise these features for jumping systems.

The violation of physical constraints is a significant limitation of static optimisation. Future work may investigate the use of a weighting matrix or a reformulation of the objective equation to enforce physical constraints in the system. This work would improve the predictive power of static optimisation and set it up as a valid predictive modelling method for jumping research.

Reinforcement learning methods often include a noise model for their exploration strategy. Different algorithms often use different models or at least present benchmark results with different parameters for their exploration noise. A study which normalises the exploration strategies across multiple reinforcement learning methods would be very useful as it would separate the learning and exploration and provide a useful comparison of the algorithms. In the current literature, it is unclear what contributes to the success of new reinforcement learning algorithms due to the convolution of hyper-parameters, neural network structures, and noise models used for benchmarking.

References

[1] S. M. Reilly, S. J. Montuelle, A. Schmidt, E. Naylor, M. E. Jorgensen, L. G. Halsey, and R. L. Essner, "Conquering the world in leaps and bounds: Hopping locomotion in toads is actually bounding," *Functional Ecology*, vol. 29, no. 10, pp. 1308–1316, Oct. 2015, ISSN: 13652435. DOI: 10.1111/1365-2435.12414.

[2] R. Blickhan, "The spring-mass model for running and hopping," *Journal of Biomechanics*, vol. 22, no. 11-12, pp. 1217–1227, Jan. 1989, ISSN: 00219290. DOI: 10.1016/0021-9290(89)90224-8. [Online]. Available: https://www.sciencedirect.com/science/article/pii/0021929089902248/pdf?md5=fca3689e0d2c827db20db1cc789bda3b&pid=1-s2.0-0021929089902248-main.pdf.

[3] C. F. Ong, J. L. Hicks, and S. L. Delp, "Simulation-Based Design for Wearable Robotic Systems: An Optimization Framework for Enhancing a Standing Long Jump," *IEEE Transactions on Biomedical Engineering*, vol. 63, no. 5, pp. 894–903, May 2016, ISSN: 0018-9294. DOI: 10.1109/TBME.2015.2463077. [Online]. Available: http://ieeexplore.ieee.org/document/7173005/.

[4] M. Lindner, A. Kotschwar, R. R. Zsoldos, M. Groesel, and C. Peham, "The jump shot - A biomechanical analysis focused on lateral ankle ligaments," *Journal of Biomechanics*, vol. 45, no. 1, pp. 202–206, Jan. 2012, ISSN: 00219290. DOI: 10.1016/j.jbiomech.2011.09.012.

[5] R. Bakker, S. Tomescu, E. Brenneman, G. Hangalur, A. Laing, and N. Chandrashekar, "Effect of sagittal plane mechanics on ACL strain during jump landing," *J Orthop Res*, vol. 34, pp. 1636–1644, 2016. DOI: 10.1002/jor.23164.

[6] H. T. Henry, D. J. Ellerby, and R. L. Marsh, "Performance of guinea fowl Numida meleagris during jumping requires storage and release of elastic energy," *Journal of Experimental Biology*, vol. 208, no. 17, pp. 3293–3302, 2005, ISSN: 00220949. DOI: 10.1242/jeb.01764. [Online]. Available: https://jeb.biologists.org/content/jexbio/208/17/3293.full.pdf.

[7] C. T. Richards, L. B. Porro, and A. J. Collings, "Kinematic control of extreme jump angles in the red-legged running frog, Kassina maculata," 2017. DOI: 10.1242/jeb.144279.

[8] L. B. Porro, A. J. Collings, E. A. Eberhard, K. P. Chadwick, and C. T. Richards, "Inverse dynamic modelling of jumping in the red-legged running frog, Kassina maculata," 2017. DOI: 10.1242/jeb.155416.

[9] Y. C. Lin and M. G. Pandy, "Three-dimensional data-tracking dynamic optimization simulations of human locomotion generated by direct collocation," *Journal of Biomechanics*, vol. 59, pp. 1–8, 2017, ISSN: 18732380. DOI: 10.1016/j.jbiomech.2017.04.038. [Online]. Available: http://dx.doi.org/10.1016/j.jbiomech.2017.04.038.

[10] X. B. Peng, P. Abbeel, S. Levine, and M. van de Panne, "DeepMimic: Example-Guided Deep Reinforcement Learning of Physics-Based Character Skills," *ACM Trans. Graph*, vol. 37, no. 143, p. 18, 2018, ISSN: 0730-0301. DOI: 10.1145/3197517.3201311. [Online]. Available: http://arxiv.org/abs/1804.02717%0Ahttp://dx.doi.org/10.1145/3197517.3201311.

[11] D. G. E. Robertson, G. E. Caldwell, J. Hamill, G. Kamen, and S. N. Whittlesey, *Research Methods in Biomechanics*. 2014. DOI: 10.5040/9781492595809.

[12] M. G. Pandy, F. E. Zajac, E. Sim, and W. S. Levine, "An optimal control model for maximum-height human jumping," *Journal of Biomechanics*, vol. 23, no. 12, pp. 1185–1198, 1990, ISSN: 00219290. DOI: 10.1016/0021-9290(90)90376-E.

[13] R. M. N. Alexander, "Simple models of walking and jumping," *Human Movement Science*, vol. 11, no. 1-2, pp. 3–9, 1992, ISSN: 01679457. DOI: 10.1016/0167-9457(92)90045-D.

[14] F. C. Anderson and M. G. Pandy, "Storage and utilization of elastic strain energy during jumping," *Journal of Biomechanics*, vol. 26, no. 12, pp. 1413–1427, 1993, ISSN: 00219290. DOI: 10.1016/0021-9290(93)90092-S.

[15] ——, "A dynamic optimization solution for vertical jumping in three dimensions," *Computer Methods in Biomechanics and Biomedical Engineering*, vol. 2, no. 3, pp. 201–231, 1999, ISSN: 10255842. DOI: 10.1080/10255849908907988. [Online]. Available: https://www.tandfonline.com/action/journalInformation?journalCode=gcmb20.

[16] W. S. Selbie and G. E. Caldwell, "A simulation study of vertical jumping from different starting postures," *Journal of Biomechanics*, vol. 29, no. 9, pp. 1137–1146, 1996, ISSN: 00219290. DOI: 10.1016/0021-9290(96)00030-9.

[17] W. Roberts, W. Levine, and F. Zajac, "Propelling a torque controlled baton to a maximum height," *IEEE Transactions on Automatic Control*, vol. 24, no. 5, pp. 779–782, Oct. 1979, ISSN: 0018-9286. DOI: 10.1109/TAC.1979.1102148. [Online]. Available:

https://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=1102148%20http://ieeexplore.ieee.org/document/1102148/.

[18] W. Li, E. Todorov, and D. Liu, "Inverse optimality design for biological movement systems," *IFAC Proceedings Volumes (IFAC-PapersOnline)*, vol. 44, no. 1 PART 1, pp. 9662–9667, 2011, ISSN: 14746670. DOI: 10.3182/20110828-6-IT-1002.00877.

[19] C. M. Hubicki, "From running birds to walking robots: optimization as a unifying framework for dynamic bipedal locomotion," Ph.D. dissertation, Oregon State University, 2015. [Online]. Available: http://ir.library.oregonstate.edu/xmlui/handle/1957/54820.

[20] W. S. Levine, M. Christodoulou, and F. E. Zajac, "On propelling a rod to a maximum vertical or horizontal distance," *Automatica*, vol. 19, no. 3, pp. 321–324, 1983, ISSN: 00051098. DOI: 10.1016/0005-1098(83)90111-5.

[21] B. Siciliano and O. Khatib, *Springer Handbook of Robotics*, B. Siciliano and O. Khatib, Eds. 2008, ISBN: 9783540239574. [Online]. Available: http://library1.nida.ac.th/termpaper6/sd/2554/19755.pdf.

[22] R. S. Sutton and A. G. Barto, *Reinforcement Learning: An Introduction*. 2017, vol. 2. DOI: 10.1016/s1364-6613(99)01331-5.

[23] T. P. Lillicrap, J. J. Hunt, A. Pritzel, N. Heess, T. Erez, Y. Tassa, D. Silver, and D. Wierstra, "Continuous Control With Deep Reinforcement Learning," Tech. Rep., 2016. [Online]. Available: https://goo.gl/J4PIAz.

[24] J. Schulman, S. Levine, P. Moritz, M. Jordan, and P. Abbeel, "Trust region policy optimization," in *32nd International Conference on Machine Learning, ICML 2015*, vol. 3, 2017, pp. 1889–1897, ISBN: 9781510810587.

[25] J. Schulman, F. Wolski, P. Dhariwal, A. Radford, and O. Klimov, "Proximal Policy Optimization Algorithms," pp. 1–12, 2017. [Online]. Available: http://arxiv.org/abs/1707.06347.

[26] T. Haarnoja, A. Zhou, K. Hartikainen, G. Tucker, S. Ha, J. Tan, V. Kumar, H. Zhu, A. Gupta, P. Abbeel, and S. Levine, "Soft Actor-Critic Algorithms and Applications," *arXiv*, 2018, ISSN: 23318422. [Online]. Available: http://arxiv.org/abs/1812.05905.

[27] D. Silver, N. Heess, T. Degris, D. Wierstra, and M. Riedmiller, "Deterministic Policy Gradient Algorithms," Tech. Rep., 2014.

[28] Y.-C. Lin, J. P. Walter, and M. G. Pandy, "Predictive Simulations of Neuromuscular Coordination and Joint-Contact Loading in Human Gait," *Annals of Biomedical Engineering*, vol. 46, 2018. DOI: 10.1007/s10439-018-2026-6. [Online]. Available: https://doi.org/10.1007/s10439-018-2026-6.

[29] O. Khatib and J. Burdick, *Dynamic Optimization in Manipulator Design: the Operational Space Formulation*, 1987.

[30] D. W. Haldane, M. M. Plecnik, J. K. Yim, and R. S. Fearing, "Robotic vertical jumping agility via Series-Elastic power modulation," *Science Robotics*, vol. 1, no. 1, 2016, ISSN: 24709476. DOI: 10.1126/scirobotics.aag2048.

[31] W. I. Sellers, *GaitSym*, Manchester, 2019. [Online]. Available: https://github.com/wol101/GaitSym2019.

[32] S. L. Delp, F. C. Anderson, A. S. Arnold, P. Loan, A. Habib, C. T. John, E. Guendelman, and D. G. Thelen, "OpenSim: Open-Source Software to Create and Analyze Dynamic Simulations of Movement," *IEEE Transactions on Biomedical Engineering*, vol. 54, no. 11, pp. 1940–1950, Nov. 2007, ISSN: 0018-9294. DOI: 10.1109/TBME.2007.901024. [Online]. Available: http://ieeexplore.ieee.org/document/4352056/.

[33] R. M. Alexander, *Principles of Animal Locomotion*. Princeton University Press, 2002, ISBN: 9781400849512. DOI: 10.1515/9781400849512.

[34] B. Parslew, G. Sivalingam, and W. Crowther, "A dynamics and stability framework for avian jumping take-off," *Royal Society Open Science*, vol. 5, no. 10, p. 17, 2018, ISSN: 20545703. DOI: 10.1098/rsos.181544. [Online]. Available: http://rsos.royalsocietypublishing.org/content/royopensci/5/10/181544.full.pdf.

[35] W. I. Sellers, "A biomechanical investigation into the absence of leaping in the locomotor repertoire of the slender loris (loris tardigradus)," *Folia Primatologica*, vol. 67, no. 1, pp. 1–14, 1996, ISSN: 00155713. DOI: 10.1159/000157202.

[36] A. M. Lyapunov, "The general problem of the stability of motion," *International Journal of Control*, vol. 55, no. 3, pp. 531–534, 1892, ISSN: 1366-5820. DOI: 10.1080/00207179208934253. [Online]. Available: https://www.tandfonline.com/action/journalInformation?journalCode=tcon20.

[37] M. VUKOBRATOVIĆ and B. BOROVAC, "Zero-Moment Point — Thirty Five Years of Its Life," *International Journal of Humanoid Robotics*, vol. 01, no. 01, pp. 157–173, 2004, ISSN: 0219-8436. DOI: 10.1142/s0219843604000083.

[38] Mathworks, *Pretrained Convolutional Neural Networks*, 2018. [Online]. Available: https://uk.mathworks.com/help/deeplearning/ug/pretrained-convolutional-neural-networks.html.

[39] W. T. Dempster and G. R. Gaughran, "Properties of body segments based on size and weight," *American Journal of Anatomy*, vol. 120, no. 1, pp. 33–54, 1967, ISSN: 15530795. DOI: 10.1002/aja.1001200104.

[40] A. V. Hill, "The Dimensions of Animals and their Muscular Dynamics," Tech. Rep. 150, 1933, pp. 209–230.

[41] ——, "The Heat of Shortening and the Dynamic Constants of Muscle," *Proceedings of the Royal Society of London. Series B - Biological Sciences*, vol. 126, no. 843, pp. 136–195, Oct. 1938, ISSN: 2053-9193. DOI: 10.1098/rspb.1938.0050. [Online]. Available: https://royalsocietypublishing.org/doi/10.1098/rspb.1938.0050.

[42] R. M. Alexander, "Optimum take-off techniques for high and long jumps," Tech. Rep. [Online]. Available: https://royalsocietypublishing.org/.

[43] T. J. Roberts and R. L. Marsh, "Probing the limits to muscle-powered accelerations: Lessons from jumping bullfrogs," *Journal of Experimental Biology*, vol. 206, no. 15, pp. 2567–2580, 2003, ISSN: 00220949. DOI: 10.1242/jeb.00452. [Online]. Available: https://jeb.biologists.org/content/jexbio/206/15/2567.full.pdf.

[44] D. W. Haldane, J. K. Yim, and R. S. Fearing, "Repetitive extreme-acceleration (14-g) spatial jumping with Salto-1P," in *International Conference on Intelligent Robots and Systems (IROS)*, Vancouver, 2017. [Online]. Available: https://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=8206172.

[45] D. W. Haldane, M. Plecnik, J. K. Yim, and R. S. Fearing, "A power modulating leg mechanism for monopedal hopping," in *IEEE International Conference on Intelligent Robots and Systems*, vol. 2016-Novem, Daejeon, 2016, pp. 4757–4764, ISBN: 9781509037629. DOI: 10.1109/IROS.2016.7759699. [Online]. Available: https://ieeexplore.ieee.org/stamp/stamp.jsp?arnumber=7759699.

[46] J. K. Yim, B. R. P. Singh, E. K. Wang, R. Featherstone, and R. S. Fearing, "Precision Robotic Leaping and Landing Using Stance-Phase Balance," *IEEE Robotics and Automation Letters*, vol. 5, no. 2, pp. 3422–3429, Apr. 2020, ISSN: 2377-3766. DOI: 10.1109/LRA.2020.2976597. [Online]. Available: https://ieeexplore.ieee.org/document/9016133/.

[47] J. Aguilar, A. Lesov, K. Wiesenfeld, and D. I. Goldman, "Lift-off dynamics in a simple jumping robot," *Physical Review Letters*, vol. 109, no. 17, 2012, ISSN: 10797114. DOI: 10.1103/PhysRevLett.109.174301.

[48] S. Onyshko and D. A. Winter, "A mathematical model for the dynamics of human locomotion," *Journal of Biomechanics*, vol. 13, no. 4, pp. 361–368, Jan. 1980, ISSN: 00219290. DOI: 10.1016/0021-9290(80)90016-0.

[49] W. J. Kargo, F. Nelson, and L. C. Rome, "Jumping in frogs: Assessing the design of the skeletal system by anatomically realistic modeling and forward dynamic simulation," *Journal of Experimental Biology*, vol. 205, no. 12, pp. 1683–1702, 2002, ISSN: 00220949.

[50] R. E. Bellman, *Dynamic Programming*. Princeton University Press, 1957, p. 342.

[51] O. Khatib, "Commande Dynamique dans l'Espace Operationnel des Robots Manipulateurs en Presence d'Obstacles," Ph.D. dissertation, l'ecole nationale superieure de l'aeronautique et de l'espace, 1980.

[52] C. A. Klein and C.-H. Huang, "Review of pseudoinverse control for use with kinematically redundant manipulators," *IEEE Transactions on Systems, Man, and Cybernetics*, vol. SMC-13, no. 2, pp. 245–250, Mar. 1983, ISSN: 0018-9472. DOI: 10.1109/TSMC.1983.6313123. [Online]. Available: http://ieeexplore.ieee.org/document/6313123/.

[53] P. Geoffroy, N. Mansard, M. Raison, S. Achiche, and E. Todorov, "From Inverse Kinematics to Optimal Control," *Advances in Robot Kinematics*, pp. 409–418, 2014. DOI: 10.1007/978-3-319-06698-1{_}42.

[54] J. Hollerbach and Ki Suh, "Redundancy resolution of manipulators through torque optimization," *IEEE Journal on Robotics and Automation*, vol. 3, no. 4, pp. 308–316, Aug. 1987, ISSN: 0882-4967. DOI: 10.1109/JRA.1987.1087111. [Online]. Available: https://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=1087111%20http://ieeexplore.ieee.org/document/1087111/.

[55] C. T. Richards and L. B. Porro, "A novel kinematics analysis method using quaternion interpolation–a case study in frog jumping," *Journal of Theoretical Biology*, vol. 454, pp. 410–424, Oct. 2018, ISSN: 10958541. DOI: 10.1016/j.jtbi.2018.06.010.

[56] F. C. Anderson and M. G. Pandy, "Static and dynamic optimization solutions for gait are practically equivalent," *Journal of Biomechanics*, vol. 34, no. 2, pp. 153–161, Feb. 2001, ISSN: 00219290. DOI: 10.1016/S0021-9290(00)00155-X.

[57] M. Kelly, *OptimTraj*, 2016. [Online]. Available: https://github.com/MatthewPeterKelly/OptimTraj.

[58] P. J. Bishop, A. Falisse, F. De Groote, and J. R. Hutchinson, "Predictive simulations of musculoskeletal function and jumping performance in a generalized bird," *Integrative Organismal Biology*, 2021. DOI: 10.1093/iob/obab006/6226705. [Online]. Available: https://academic.oup.com/iob/advance-article/doi/10.1093/iob/obab006/6226705.

[59] P. J. Bishop, K. B. Michel, A. Falisse, A. R. Cuff, V. R. Allen, F. de Groote, and J. R. Hutchinson, *Computational modelling of muscle fibre operating ranges in the hindlimb of a small ground bird (Eudromia elegans), with implications for modelling locomotion in extinct species*, 4. 2021, vol. 17, pp. 1–46, ISBN: 1111111111. DOI: 10.1371/JOURNAL.PCBI.1008843. [Online]. Available: http://dx.doi.org/10.1371/journal.pcbi.1008843.

[60] G. Brockman, V. Cheung, L. Pettersson, J. Schneider, J. Schulman, J. Tang, and W. Z. Openai, "OpenAI Gym," Tech. Rep., 2016.

[61] E. Todorov, T. Erez, and Y. Tassa, "MuJoCo: A physics engine for model-based control," Tech. Rep., 2012.

[62] Y. Duan, X. Chen, R. Houthooft, J. Schulman, and P. Abbeel, "Benchmarking deep reinforcement learning for continuous control," in *33rd International Conference on Machine Learning, ICML 2016*, vol. 3, 2016, pp. 2001–2014, ISBN: 9781510829008. [Online]. Available: https://github.com/.

[63] N. Heess, T. B. Dhruva, S. Sriram, J. Lemmon, J. Merel, G. Wayne, Y. Tassa, T. Erez, Z. Wang, S. M. Ali Eslami, M. Riedmiller, and D. Silver, *Emergence of locomotion behaviours in rich environments*, 2017.

[64] Y. Wu, E. Mansimov, S. Liao, R. Grosse, and J. Ba, "Scalable trust-region method for deep reinforcement learning using Kronecker-factored approximation," in *Advances in Neural Information Processing Systems*, vol. 2017-Decem, 2017, pp. 5280–5289. [Online]. Available: https://github.com/openai/baselines..

[65] S. Gu, T. Lillicrap, Z. Ghahramani, R. E. Turner, and S. Levine, "Q-PrOP: Sample-efficient policy gradient with an off-policy critic," in *5th International Conference on Learning Representations, ICLR 2017 - Conference Track Proceedings*, 2017. DOI: 10.17863/CAM.21294.

[66] J. Tan, T. Zhang, E. Coumans, A. Iscen, Y. Bai, D. Hafner, S. Bohez, and V. Vanhoucke, "Sim-to-Real: Learning Agile Locomotion For Quadruped Robots," 2018. DOI: 10.15607/RSS.2018.XIV.010. [Online]. Available: https://arxiv.org/pdf/1804.10332.pdf%20http://arxiv.org/abs/1804.10332.

[67] J. Hwangbo, J. Lee, A. Dosovitskiy, D. Bellicoso, V. Tsounis, V. Koltun, and M. Hutter, "Learning agile and dynamic motor skills for legged robots," *Science Robotics*, vol. 4, no. 26, 2019, ISSN: 24709476. DOI: 10.1126/scirobotics.aau5872. [Online]. Available: http://robotics.sciencemag.org/.

[68] D. Amodei, D. Hernandez, G. Sastry, J. Clark, G. Brockman, and I. Sutskever, *AI and Compute*, 2018. [Online]. Available: https://openai.com/blog/ai-and-compute/#modern.

[69] A. Kanazawa, J. Malik, P. Abbeel, and S. Levine, "SFV: Reinforcement Learning of Physical Skills from Videos," *ACM Trans. Graph*, vol. 37, p. 17, 2018. DOI: 10.1145/3272127.3275014. [Online]. Available: https://xbpeng.github.io/projects/SFV/index.html.

[70] E. Todorov, "Convex and analytically-invertible dynamics with contacts and constraints: Theory and implementation in MuJoCo," Tech. Rep., 2014.

[71] K. Kazerounian and Z. Wang, "Global versus Local Optimization in Redundancy Resolution of Robotic," Tech. Rep.

[72] D. E. Whitney, "Resolved Motion Rate Control of Manipulators and Human Prostheses," *IEEE Transactions on Man-Machine Systems*, vol. 10, no. 2, pp. 47–53, 1969, ISSN: 21682860. DOI: 10.1109/TMMS.1969.299896.

[73] S. M. Gatesy and A. A. Biewener, "Bipedal locomotion: effects of speed, size and limb posture in birds and humans," *Journal of Zoology*, vol. 224, no. 1, pp. 127–147, 1991, ISSN: 14697998. DOI: 10.1111/j.1469-7998.1991.tb04794.x.

[74] Y. Nakamura, H. Hanafusa, and T. Yoshikawa, "Task-Priority Based Redundancy Control of Robot Manipulators," Tech. Rep.

[75] P. E. Gill, W. Murray, and M. A. Saunders, "SNOPT: An SQP Algorithm for Large-Scale Constrained Optimization *," *Society for Industrial and Applied Mathematics*, vol. 47, no. 1, pp. 99–131, 2005. DOI: 10.1137/S0036144504446096. [Online]. Available: http://www.siam.org/journals/sirev/47-1/44609.html.

[76] Many, *Optimization (scipy.optimize) — SciPy v1.6.1 Reference Guide*, 2021. [Online]. Available: https://docs.scipy.org/doc/scipy-1.6.1/reference/tutorial/optimize.html.

[77] A. Wächter, ·. Lorenz, and T. Biegler, "Digital Object Identifier (On the implementation of an interior-point filter line-search algorithm for large-scale nonlinear programming," *Math. Program., Ser. A*, vol. 106, pp. 25–57, 2006. DOI: 10.1007/s10107-004-0559-y.

[78] M. Diehl, H. G. Bock, H. Diedam, and P.-b. Wieber, "Fast Direct Multiple Shooting Algorithms for Optimal Robot Control," *Fast Motions in Biomechanics and Robotics*, p. 28, 2005.

[79] E. Todorov, "Optimality principles in sensorimotor control," *Nature Neuroscience*, vol. 7, no. 9, pp. 907–915, 2004, ISSN: 10976256. DOI: 10.1038/nn1309.

[80] C. Wilson, M. R. Yeadon, and M. A. King, "Considerations that affect optimised simulation in a running jump for height," *Journal of Biomechanics*, vol. 40, no. 14, pp. 3155–3161, 2007, ISSN: 00219290. DOI: 10.1016/j.jbiomech.2007.03.030.

[81] M. R. A. Nabawy, G. Sivalingam, R. J. Garwood, W. J. Crowther, and W. I. Sellers, "Energy and time optimal trajectories in exploratory jumps of the spider Phidippus regius OPEN," DOI: 10.1038/s41598-018-25227-9. [Online]. Available: www.nature.com/scientificreports.

[82] W. I. Sellers, S. B. Pond, C. A. Brassey, P. L. Manning, and K. T. Bates, "Investigating the running abilities of Tyrannosaurus rex using stress-constrained multibody dynamic analysis," *PeerJ*, vol. 2017, no. 7, pp. 1–19, 2017, ISSN: 21678359. DOI: 10.7717/peerj.3420.

[83] F. Antonio, "FASTER LINE SEGMENT INTERSECTION," in *Graphics Gems III (IBM Version)*, Elsevier, 1992, pp. 199–202. DOI: 10.1016/b978-0-08-050755-2.50045-2.

[84] E. Todorov, *Emo Todorov (University of Washington): "Acceleration-based methods" - YouTube*, Jun. 2019. [Online]. Available: https://www.youtube.com/watch?v=uWADBSmHebA.

[85] J. T. Betts, *Practical Methods for Optimal Control and Estimation Using Nonlinear Programming*. Society for Industrial and Applied Mathematics, Jan. 2010. DOI: 10.1137/1.9780898718577.

[86] D. Silver, J. Schrittwieser, K. Simonyan, I. Antonoglou, A. Huang, A. Guez, T. Hubert, L. Baker, M. Lai, A. Bolton, Y. Chen, T. Lillicrap, F. Hui, L. Sifre, G. Van Den Driessche, T. Graepel, and D. Hassabis, "Mastering the game of Go without human knowledge," *Nature*, vol. 550, no. 7676, pp. 354–359, Oct. 2017, ISSN: 14764687. DOI: 10.1038/nature24270.

[87] I. Akkaya, M. Andrychowicz, M. Chociej, M. Litwin, B. McGrew, A. Petron, A. Paino, M. Plappert, G. Powell, R. Ribas, J. Schneider, N. Tezak, J. Tworek, P. Welinder, L. Weng, Q. Yuan, W. Zaremba, and L. Zhang, *Solving Rubik's cube with a robot hand*, 2019. [Online]. Available: https://openai.com/bibtex/openai2019rubiks.bib.

[88] R. J. Williams, "Simple Statistical Gradient-Following Algorithms for Connectionist Reinforcement Learning," *Machine Learning*, vol. 8, no. 3, pp. 229–256, 1992, ISSN: 15730565. DOI: 10.1023/A:1022672621406.

[89] V. Mnih, K. Kavukcuoglu, D. Silver, A. Graves, I. Antonoglou, D. Wierstra, and M. Riedmiller, "Playing Atari with Deep Reinforcement Learning," 2013. [Online]. Available: http://arxiv.org/abs/1312.5602.

[90] V. Mnih, K. Kavukcuoglu, D. Silver, A. A. Rusu, J. Veness, M. G. Bellemare, A. Graves, M. Riedmiller, A. K. Fidjeland, G. Ostrovski, S. Petersen, C. Beattie, A. Sadik, I. Antonoglou, H. King, D. Kumaran, D. Wierstra, S. Legg, and D. Hassabis, "Human-level control through deep reinforcement learning," *Nature*, vol. 518, no. 7540, pp. 529–533, 2015, ISSN: 14764687. DOI: 10.1038/nature14236.

[91] E. Todorov, "Goal Directed Dynamics," *IEEE International Conference on Robotics and Automation (ICRA)*, pp. 2994–3000, 2018. DOI: 10.1109/ICRA.2018.8462904.

Appendices

Appendix A

Equations for Model Analysis Chapter

The equations of motion for the 2 degree of freedom model are:

$$\tau_1 = l_1\, l_2\, m_2\, \sin(q_1 - q_2)\, \dot{q}_2^2 + \left(I_1 + \frac{{l_1}^2\, m_1}{4} + {l_1}^2\, m_2 \right) \ddot{q}_1 + \frac{g\, l_1\, m_1\, \cos(q_1)}{2} + g\, l_1\, m_2\, \cos(q_1) + l_1\, l_2\, m_2\, \ddot{q}_2\, \cos(q_1 - q_2)$$

$$\tau_2 = -l_1\, l_2\, m_2\, \sin(q_1 - q_2)\, \dot{q}_1^2 + \left(m_2\, {l_2}^2 + I_2 \right) \ddot{q}_2 + g\, l_2\, m_2\, \cos(q_2) + l_1\, l_2\, m_2\, \ddot{q}_1\, \cos(q_1 - q_2) \quad \text{(A.1)}$$

The forward dynamics are long equations. The acceleration of the first joint, q_1, is given by:

$$\ddot{q}_1 = \frac{A}{B} \quad \text{(A.2)}$$

where

$$A = 4\, l_1\, l_2\, m_2\, \tau_2\, \cos(q_1 - q_2) - 4\, {l_2}^2\, m_2\, \tau_1 - 4\, I_2\, \tau_1 + 4\, l_1\, {l_2}^3\, {m_2}^2\, \dot{q}_2^2\, \sin(q_1 - q_2) + 4\, g\, l_1\, {l_2}^2\, {m_2}^2\, \cos(q_1) + 2\, I_2\, g\, l_1\, m_1\, \cos(q_1) + 4\, I_2\, g\, l_1\, m_2\, \cos(q_1) - 4\, g\, l_1\, {l_2}^2\, {m_2}^2\, \cos(q_1 - q_2)\, \cos(q_2) + 2\, g\, l_1\, {l_2}^2\, m_1\, m_2\, \cos(q_1) + 4\, {l_1}^2\, {l_2}^2\, {m_2}^2\, \dot{q}_1^2\, \cos(q_1 - q_2)\, \sin(q_1 - q_2) + 4\, I_2\, l_1\, l_2\, m_2\, \dot{q}_2^2\, \sin(q_1 - q_2)$$

$$B = 4\, {l_1}^2\, {l_2}^2\, {m_2}^2\, \cos^2(q_1 - q_2) - 4\, {l_1}^2\, {l_2}^2\, {m_2}^2 - m_1\, {l_1}^2\, {l_2}^2\, m_2 - 4\, I_2\, {l_1}^2\, m_2 - I_2\, m_1\, {l_1}^2 - 4\, I_1\, {l_2}^2\, m_2 - 4\, I_1\, I_2.$$

The equation for the acceleration of the second joint, q_2, is given by:

$$\ddot{q}_2 = \frac{C}{D} \quad \text{(A.3)}$$

where

$$C = 4\, I_1\, \tau_2 + {l_1}^2\, m_1\, \tau_2 + 4\, {l_1}^2\, m_2\, \tau_2 - 4\, l_1\, l_2\, m_2\, \tau_1\, \cos(q_1 - q_2) + 4\, {l_1}^3\, l_2\, {m_2}^2\, \dot{q}_1^2\, \sin(q_1 - q_2)$$

$$
\begin{aligned}
&- 4\,g\,{l_1}^2\,l_2\,{m_2}^2\,\cos(q_2) - 4\,I_1\,g\,l_2\,m_2\,\cos(q_2) + {l_1}^3\,l_2\,m_1\,m_2\,\dot{q}_1^2\,\sin(q_1 - q_2) \\
&+ 4\,g\,{l_1}^2\,l_2\,{m_2}^2\,\cos(q_1 - q_2)\,\cos(q_1) - g\,{l_1}^2\,l_2\,m_1\,m_2\,\cos(q_2) \\
&+ 4\,{l_1}^2\,{l_2}^2\,{m_2}^2\,\dot{q}_2^2\,\cos(q_1 - q_2)\,\sin(q_1 - q_2) + 4\,I_1\,l_1\,l_2\,m_2\,\dot{q}_1^2\,\sin(q_1 - q_2) \\
&+ 2\,g\,{l_1}^2\,l_2\,m_1\,m_2\,\cos(q_1 - q_2)\,\cos(q_1)
\end{aligned}
$$

$$
\begin{aligned}
D = &-4\,{l_1}^2\,{l_2}^2\,{m_2}^2\,\cos^2(q_1 - q_2) + 4\,{l_1}^2\,{l_2}^2\,{m_2}^2 + m_1\,{l_1}^2\,{l_2}^2\,m_2 + 4\,I_2\,{l_1}^2\,m_2 \\
&+ I_2\,m_1\,{l_1}^2 + 4\,I_1\,{l_2}^2\,m_2 + 4\,I_1\,I_2.
\end{aligned}
$$

The ground reaction force acting at the foot is given by:

$$
F_v = \frac{G}{H} \tag{A.4}
$$

where

$$
\begin{aligned}
G = &\,8\,I_1\,I_2\,g\,m_1 + 8\,I_1\,I_2\,g\,m_2 - 4\,l_1\,{l_2}^2\,{m_2}^2\,\tau_1 - 4\,{l_1}^2\,l_2\,{m_2}^2\,\tau_2 \\
&+ 8\,{l_1}^2\,l_2\,{m_2}^2\,\tau_2\,\sin^2\Big(q_1 - \frac{q_2}{2}\Big) + 8\,l_1\,{l_2}^2\,{m_2}^2\,\tau_1\,\sin^2\Big(\frac{q_1}{2} - q_2\Big) + 4\,l_1\,{l_2}^2\,{m_2}^2\,\tau_1\,\cos(q_1) \\
&+ 4\,{l_1}^2\,l_2\,{m_2}^2\,\tau_2\,\cos(q_2) + 8\,I_1\,g\,{l_2}^2\,m_1\,m_2 + 2\,I_2\,g\,{l_1}^2\,m_1\,m_2 + 4\,I_2\,l_1\,m_1\,\tau_1\,\cos(q_1) \\
&+ 8\,I_2\,l_1\,m_2\,\tau_1\,\cos(q_1) + 8\,I_1\,l_2\,m_2\,\tau_2\,\cos(q_2) - 2\,{l_1}^2\,l_2\,m_1\,m_2\,\tau_2 + 2\,I_2\,g\,{l_1}^2\,{m_1}^2\,\sin^2(q_1) \\
&+ 8\,I_2\,g\,{l_1}^2\,{m_2}^2\,\sin^2(q_1) + 8\,I_1\,g\,{l_2}^2\,{m_2}^2\,\sin^2(q_2) - I_2\,{l_1}^3\,{m_1}^2\,\dot{q}_1^2\,\sin(q_1) \\
&- 8\,I_2\,{l_1}^3\,{m_2}^2\,\dot{q}_1^2\,\sin(q_1) - 8\,I_1\,{l_2}^3\,{m_2}^2\,\dot{q}_2^2\,\sin(q_2) - 6\,I_2\,{l_1}^3\,m_1\,m_2\,\dot{q}_1^2\,\sin(q_1) \\
&+ 4\,g\,{l_1}^2\,{l_2}^2\,m_1\,{m_2}^2\,\sin^2(q_1 - q_2) - {l_1}^3\,{l_2}^2\,m_1\,{m_2}^2\,\dot{q}_1^2\,\sin(q_1 - 2\,q_2) \\
&+ 4\,{l_1}^2\,l_2\,m_1\,m_2\,\tau_2\,\sin^2\Big(q_1 - \frac{q_2}{2}\Big) - 4\,I_1\,l_1\,{l_2}^2\,{m_2}^2\,\dot{q}_1^2\,\sin(q_1) - 4\,I_2\,{l_1}^2\,l_2\,{m_2}^2\,\dot{q}_2^2\,\sin(q_2) \\
&- 2\,{l_1}^2\,{l_2}^3\,m_1\,{m_2}^2\,\dot{q}_2^2\,\sin(2\,q_1 - q_2) - 4\,I_1\,I_2\,l_1\,m_1\,\dot{q}_1^2\,\sin(q_1) - 8\,I_1\,I_2\,l_1\,m_2\,\dot{q}_1^2\,\sin(q_1) \\
&- 8\,I_1\,I_2\,l_2\,m_2\,\dot{q}_2^2\,\sin(q_2) + 4\,I_1\,l_1\,{l_2}^2\,{m_2}^2\,\dot{q}_1^2\,\sin(q_1 - 2\,q_2) + 4\,l_1\,{l_2}^2\,m_1\,m_2\,\tau_1\,\cos(q_1) \\
&+ 4\,g\,{l_1}^2\,{l_2}^2\,m_1\,{m_2}^2\,\sin^2(q_1) + 2\,g\,{l_1}^2\,{l_2}^2\,{m_1}^2\,m_2\,\sin^2(q_1) - 2\,g\,{l_1}^2\,{l_2}^2\,m_1\,{m_2}^2\,\sin^2(q_2) \\
&- 3\,{l_1}^3\,{l_2}^2\,m_1\,{m_2}^2\,\dot{q}_1^2\,\sin(q_1) - {l_1}^3\,{l_2}^2\,{m_1}^2\,m_2\,\dot{q}_1^2\,\sin(q_1) - 4\,I_2\,{l_1}^2\,l_2\,{m_2}^2\,\dot{q}_2^2\,\sin(2\,q_1 - q_2) \\
&+8\,I_2\,g\,{l_1}^2\,m_1\,m_2\,\sin^2(q_1) - 2\,I_2\,{l_1}^2\,l_2\,m_1\,m_2\,\dot{q}_2^2\,\sin(2\,q_1 - q_2) - 4\,I_1\,l_1\,{l_2}^2\,m_1\,m_2\,\dot{q}_1^2\,\sin(q_1)
\end{aligned}
$$

$$
\begin{aligned}
H = &\,8\,{l_1}^2\,{l_2}^2\,{m_2}^2\,\sin^2(q_1 - q_2) + 2\,m_1\,{l_1}^2\,{l_2}^2\,m_2 \\
&+ 8\,I_2\,{l_1}^2\,m_2 + 2\,I_2\,m_1\,{l_1}^2 + 8\,I_1\,{l_2}^2\,m_2 + 8\,I_1\,I_2
\end{aligned}
$$

Appendix B

Model Equations of Motion

Subscripts are used to denote common expressions to compress the equations into a more readable form. Trigonometric functions are shortened such that sin becomes s and cos becomes c. Joint angles used in the trigonometric functions are included in subscripts. A single value in the subscript, c_m, represents the joint angle used $\cos(q_m)$. Subscripts containing two values are used to represent angle differences: c_{nm} represents $\cos(q_n - q_m)$ and s_{nm} represents $\sin(q_n - q_m)$. The products of segment lengths are shortened using subscripts as: $l_{m \cdot n}$ represents $l_m l_n$.

Equations are presented in the form as described in 1.8a:

$$\boldsymbol{H}\ddot{\boldsymbol{q}} + \boldsymbol{C}\dot{\boldsymbol{q}} + \boldsymbol{\tau}_g = \boldsymbol{\tau}$$

B.1 1 Degree of Freedom

The equation of motion for the 1 degree of freedom model is:

$$\boldsymbol{H} = m_1 {l_1}^2 + I_b \tag{B.1}$$

$$\boldsymbol{C} = 0 \tag{B.2}$$

$$\boldsymbol{\tau}_g = m_1 \, g \, l_1 \, c_1 \tag{B.3}$$

B.2 2 Degree of Freedom

The equations of motion for the 2 degree of freedom model are:

$$\boldsymbol{H} = \begin{bmatrix} I_1 + \frac{1}{4}\, m_1\, {l_1}^2 + m_2\, {l_1}^2 & m_2\, l_{1.2}\, c_{12} \\ m_2\, l_{1.2}\, c_{12} & m_2\, {l_2}^2 + I_b \end{bmatrix} \tag{B.4}$$

$$\boldsymbol{C} = \begin{bmatrix} m_2\, l_{1.2}\, s_{12}\, \dot{q}_2 \\ -m_2\, l_{1.2}\, s_{12}\, \dot{q}_1 \end{bmatrix} \tag{B.5}$$

$$\boldsymbol{\tau_g} = \begin{bmatrix} \frac{1}{2}\, m_1\, g\, l_1\, c_1 + m_2\, g\, l_1\, c_1 \\ m_2\, g\, l_2\, c_2 \end{bmatrix} \tag{B.6}$$

B.3 4 Degree of Freedom

The equations of motion for the 4 degree of freedom model are:

$$\boldsymbol{H} = \begin{bmatrix} I_1 + \frac{1}{4}\,{l_1}^2\,m_1 + {l_1}^2\,m_2 + {l_1}^2\,m_3 + {l_1}^2\,m_4 & \frac{1}{2}\,l_{1{\cdot}2}\,c_{12}\,m_2 + 2\,m_3 + 2\,m_4 & \frac{1}{2}\,l_{1{\cdot}3}\,c_{13}\,m_3 + 2\,m_4 & l_{1{\cdot}4}\,m_4\,c_{14} \\ \frac{1}{2}\,l_{1{\cdot}2}\,c_{12}\,m_2 + 2\,m_3 + 2\,m_4 & I_2 + \frac{1}{4}\,{l_2}^2\,m_2 + {l_2}^2\,m_3 + {l_2}^2\,m_4 & \frac{1}{2}\,l_{2{\cdot}3}\,c_{23}\,m_3 + 2\,m_4 & l_{2{\cdot}4}\,m_4\,c_{24} \\ \frac{1}{2}\,l_{1{\cdot}3}\,c_{13}\,m_3 + 2\,m_4 & \frac{1}{2}\,l_{2{\cdot}3}\,c_{23}\,m_3 + 2\,m_4 & I_3 + \frac{1}{4}\,{l_3}^2\,m_3 + {l_3}^2\,m_4 & l_{3{\cdot}4}\,m_4\,c_{34} \\ l_{1{\cdot}4}\,m_4\,c_{14} & l_{2{\cdot}4}\,m_4\,c_{24} & l_{3{\cdot}4}\,m_4\,c_{34} & m_4\,{l_4}^2 + I_b \end{bmatrix} \tag{B.7}$$

$$\boldsymbol{C} = \begin{bmatrix} \frac{1}{2}\,l_{1{\cdot}2}\,m_2\,\dot{q}_2\,s_{12} + l_{1{\cdot}2}\,m_3\,\dot{q}_2\,s_{12} + l_{1{\cdot}2}\,m_4\,\dot{q}_2\,s_{12} + \frac{1}{2}\,l_{1{\cdot}3}\,m_3\,\dot{q}_3\,s_{13} + l_{1{\cdot}3}\,m_4\,\dot{q}_3\,s_{13} + l_{1{\cdot}4}\,m_4\,\dot{q}_4\,s_{14} \\ -\frac{1}{2}\,l_{1{\cdot}2}\,m_2\,\dot{q}_1\,s_{12} - l_{1{\cdot}2}\,m_3\,\dot{q}_1\,s_{12} - l_{1{\cdot}2}\,m_4\,\dot{q}_1\,s_{12} + \frac{1}{2}\,l_{2{\cdot}3}\,m_3\,\dot{q}_3\,s_{23} + l_{2{\cdot}3}\,m_4\,\dot{q}_3\,s_{23} + l_{2{\cdot}4}\,m_4\,\dot{q}_4\,s_{24} \\ -\frac{1}{2}\,l_{1{\cdot}3}\,m_3\,\dot{q}_1\,s_{13} - l_{1{\cdot}3}\,m_4\,\dot{q}_1\,s_{13} - \frac{1}{2}\,l_{2{\cdot}3}\,m_3\,\dot{q}_2\,s_{23} - l_{2{\cdot}3}\,m_4\,\dot{q}_2\,s_{23} + l_{3{\cdot}4}\,m_4\,\dot{q}_4\,s_{34} \\ -l_{1{\cdot}4}\,m_4\,s_{14}\,\dot{q}_1 - l_{2{\cdot}4}\,m_4\,s_{24}\,\dot{q}_2 - l_{3{\cdot}4}\,m_4\,s_{34}\,\dot{q}_3 \end{bmatrix} \tag{B.8}$$

$$\boldsymbol{\tau_g} = \begin{bmatrix} \frac{1}{2}\,g\,l_1\,m_1\,c_1 + g\,l_1\,m_2\,c_1 + g\,l_1\,m_3\,c_1 + g\,l_1\,m_4\,c_1 \\ \frac{1}{2}\,g\,l_2\,m_2\,c_2 + g\,l_2\,m_3\,c_2 + g\,l_2\,m_4\,c_2 \\ \frac{1}{2}\,g\,l_3\,m_3\,c_3 + g\,l_3\,m_4\,c_3 \\ g\,l_4\,m_4\,c_4 \end{bmatrix} \tag{B.9}$$

Appendix C

DDPG Hyper-parameters

The hyper-parameters used in the final 25 runs of DDPG were as follows.

Parameter	Value
Actor Neural Network Hidden Layers	2
Actor Neural Network Hidden Units (per layer)	15
Critic Neural Network Hidden Layers	2
Critic Neural Network Hidden Units (per layer)	15
Learning Rate	0.01
Discount Factor	1.0
Noise Standard Deviation	0.3
Mean Attraction Constant	0.1
Training Steps	20,000

Table C.1: Hyper-parameters used for the DDPG method