Intelligent Methods for Locomotion Optimisation

by

Jonathan Wright

School of Computing

The thesis is submitted to the **University of Portsmouth**

for the degree of **Doctor of Philosophy**

November 2015

Declaration

The work presented in this thesis has been carried out in the School of Computing at the University of Portsmouth under the supervision of Dr. Ivan Jordanov.

Whilst registered as a candidate for the above degree, I have not been registered for any other research award. The results and conclusions embodied in this thesis are the work of the named candidate and have not been submitted for any other academic award.

Jonathan Wright

Abstract

This thesis presents, critical compares and develops new methods to control and optimise locomotion for a range of systems. Jumping and running locomotion skills are examined in detail, and intelligent methods are discussed and adapted to optimise for correct form of motion, and performance outcomes. Existing control techniques are summarised and compared, including traditional analytical methods, central pattern generator oscillator systems, pattern generating neural networks, rule based systems and other specialist methods.

Optimisation and learning methods presented in the literature are also summarised, and while several methods exist, modern global search methods were limited to genetic algorithms. This thesis applies particle swarm optimisation and quantum inspired evolutionary algorithms to vertical jump and walking optimisation, comparing their performance to a genetic algorithm. Improvements were developed for both binary and real-value variants of quantum inspired evolutionary algorithms, to benefit performance on the real-value problems involved in locomotion control. These improvements consisted of modifications to their rotation gate operators, including a novel scheme to reduce premature convergence in the binary methods, based on limiting the range of less significant bits.

Methods were applied in simulated environments, although they can be adapted to real world robotic control, or for reference in optimising motion in humans. A discussion of the susceptibility of simulation runs to poor physical modelling was presented, as this was a significant problem during research. Results were generally mixed, to the extent that all tested methods may be usefully examined more in future work. The central pattern generators tested generated successful patterns more often than a recurrent neural network, and the results of the optimisation algorithms did not show sufficient advantage of one over the others.

Contents

List of figures

List of tables

Glossary

Abbreviation	Meaning
3D-LIPM	Three Dimensional Linear Inverted Pendulum Model
bQIEA	Binary Quantum Inspried Evolutionary Algorithm
CMAC	Cerebellar Model Articulation Controller
CMP	Centroidal Moment Point
CoG	Centre of Gravity
CPG	Central Pattern Generator
DE	Differential Evolution
DoF	Degrees of Freedom
ECDF	Empirical Cumulative Probability Distribution Function
EDA	Estimation of distribution algorithm
FRI	Foot Rotation Indicator
GA	Genetic Algorithm
HMM	Hidden Markov model
NME	Normalised Mean Error
NN	Neural Network
OpenHRP3	Open Architecture Human-centered Robotics Platform version 3
PSO	Particle Swarm Optimisation
QIEA	Quantum Inspried Evolutionary Algorithm
RCQIEA	Real-coded Quantum Inspired Evolutionary Algorithm
RNN	Recurrent Neural Network
rQIEA	Real Quantum Inspried Evolutionary Algorithm
SP	Success Performance
SR	Success Rate
SRQEA	Stepwise Real Quantum Evolutionary Algorithm
ZMP	Zero Moment Point

1 Introduction

1.1 Motivation

A challenge for modern computer science is the development of algorithms to deal with increasingly complex optimisation problems. Such challenges include a variety of practical real-world problems, such as structural engineering [1], [2], antenna design [3], electric power systems [4], digital image watermarking [5], EEG classification [6], benchmark problems [7], or mathematical functions designed to test or challenge aspects of optimisation [8], [9].

This thesis investigates the task of optimising the control of legged mechanical or biomechanical systems (robots, human models, etc…) to produce various locomotion patterns, including walking, running and jumping, with a focus on coping with the complexity present in detailed models of a target system.

When using analytical approaches to these problems, we need to be able to express a measure of the desired outcome (such as maximum distance obtained), as a function of the parameters of the model of motion, and apply optimisation techniques to the model formulae. This is simplified if we can differentiate the system of equations in order to find minima but, in order to do this, real world applications may require an amount of simplification in the modelling process, such as in [10]. For example, predicting projectile flight distance from a throw can be done as a function of launch speed and angle. However, such simplifications reduce the range of questions that can be answered, and are susceptible to false conclusions due to omission of key factors. In this throwing example, can we say how the body should best move in order to maximise speed? Also, do we make false conclusions over the preferred angle by ignoring spin and material characteristics of the projectile, its interaction with air, etc..?

The initial inspiration for this thesis was looking at how these questions relate to sport, and a related discussion was identified in [11] including an justification for the need of optimisation in sport, the limitations of current approaches and requirements that will need to be met by new methods. The authors of [11] noted that simple models used in analytical approaches can result in three problems. Firstly, the *reliability* of the results is questionable – movement patterns derived in the simplified model may not be (even approximately) optimal in the real world system. Secondly, the simplified model may not contain the *detail* required - some sport skills have significant contributions from large and small muscle groups simultaneously [12], but it is difficult to incorporate both scales in a simple model. Finally, it is difficult to express differences between *individuals* in a simplified model.

In order to improve the predictions, our models should become more detailed. However, we soon reach the point where we cannot solve their dynamics (in the form of a system of differential equations) analytically and must determine their behaviour through real-life observation [13], or computer simulation. More detailed models of the human body have been developed in [14], [15], but these models have many degrees of freedom, and to optimise locomotion using them we need to develop techniques that can cope with their complexity. Furthermore, we need to develop control algorithms that can generate movement patterns with enough parameterisation and range of output, such that the optimisation algorithm can locate good solutions.

This introduction opened with the goal of investigation optimisation of both biological and mechanical systems. Although the subsequent discussion has focussed on human sport, movement control of legged systems is required in other disciplines. These are outlined in a later literature review, but the largest identified use of locomotion control, and optimisation of that control, is in the field of robotics. Unlike for sport applications, robotics (and potentially other fields) can use control methods directly, and therefore the questions outlined above will become reframed into objectives for the control algorithm: how do we optimise a desired goal such as maximum speed, and can we develop a control method that is consistent, resilient to changes in environment or body state, able to control many degrees of freedom, and customised to the specific design or implementation of the target system? Increasingly complicated robotic systems will require optimisation procedures that can deal with more degrees of freedom and

accurately match the demands of the system, so both the sport perspective goals and robotic goals (and other domains' goals) will impose the same requirements.

If the target system can be programmed, optimisation of control can be performed in a real-world setting. However, this process may be slow, prone to damage when optimisation candidates result in a poorly configured control method, or simply not possible if the target system cannot be programmed (as is the case with human targets). In these cases, computer simulation can be used to determine optimal control and the resulting algorithm could then be copied to the target system if it is programmable, or the movement patterns be used as an instruction guide for human targets.

1.2 Problem statement and objectives

The problem under investigation by this thesis, is to establish and critically compare methods suitable for optimisation of locomotion control in simulated physical systems designed to closely model real-world targets. Given the motivation from section 1.1, these methods should be able to work with complicated models and support detailed conclusions on viable control methods or optimal gaits/movement patterns. The modelling should be accurate enough that control algorithms can be rapidly transferred from simulated environments to real-world systems, such as legged robots, or that optimised movement patterns can be used as a template for human instruction, with a reliable expectation that the pattern will be approximately optimal for them too. The control algorithms need to be general enough, and the optimisation techniques powerful enough, to attain good results that can be assumed to be approximately optimal and not overly constrained by the framework of the control algorithm.

In an attempt to address this problem, the following objectives were stated for the research covered in this thesis:

- *Review and compare successful control methods for locomotion.* To enable experimentation, this objective seeks to identify, categorise and critical compare published control algorithms. Then suitable candidates can be selected, according to the evaluation based on applicability and optimisation potential, for use in later research conducted for this thesis.

- *Review, compare and add successful optimisation techniques for locomotion*. The fundamental goal of this thesis is to investigate an optimisation process, and

so this objective is set to identify the types of optimisation algorithm presented in the literature that have been applied to locomotion. Furthermore, this goal seeks to apply techniques not previously identified as being used in the field, to expand the knowledge of what methods can be used and how different optimisation techniques compare to each other when applied to locomotion.

- *Develop new optimisation algorithm variants and apply to locomotion.* Following the review of used optimisation algorithms, and application of previously unused techniques, this objective seeks to expand further the options and points of comparison for optimisation techniques by developing new or modified optimisation algorithms, both for locomotion and for general application.

- *Develop new fitness functions to describe motion skills not identified in the literature*. This goal aims to expand the type of locomotion skills that can be evolved, crucial in order to be able to generalise techniques to answer questions for any sport skill, to enable robots to do more tasks in an environment, or to expand the range of useful activities in any equivalent application. Skills must be encoded in a fitness function that the optimisation algorithm can then minimise (or maximise) – a task that is potentially non-trivial as poorly specified functions may promote sub-optimal solutions relative to the desired outcome, or even an unforeseen outcome that does not match the original intention of the developer.

- *Compare presented methods across these different skill objectives, and for different physical systems.* While establishing working methods is a useful endeavour, this objective also seeks to critical compare developed methods in terms of success rates relative to skill criteria and the fitness values achieved. This will enable a discussion to begin on the relative usefulness of tested control methods and optimisation techniques, in differing physical system contexts and for different movement skills.

- *Implement simulation techniques with different software platforms, emphasising their advantages and disadvantages.* It is not a goal of this thesis to develop physics simulation platforms, and so a review of existing options must be conducted, and suitable candidates chosen for experimentation. It is an

objective to use more than one platform in order establish that the methods do work across different physics software, and to make use of advantages one package may have over others.

- *Examine the process of simulating physical systems, and how model design affects outcomes with a variety of model complexity.* A range of physical models should be developed for the physics platforms that can vary the difficulty of optimisation (through the complexity of the model), type of system (ideally both bipeds and quadrupeds) and realism when compared to target systems (e.g. human body). The modelling process should examine how best to express the desired system with the physics package and how the dynamic properties, such as joint actuator models, affect the types of movement possible and suitability for optimisation.

1.3 Original contributions

The original contributions of this thesis are:

- Development of new fitness functions, in order to optimise movement for vertical and long jumping;
- Investigation and development of a new *quantum inspired evolutionary algorithms* (QIEA), adapted and tested for use with real-value optimisation problems. Adaptations were made for both binary and real-coded QIEA. The binary QIEA included a scheme to prevent premature convergence of least-significant bits in the solution strings. The modified real-coded QIEA proposes a new rotation gate formula;
- Application of *particle swarm optimisation* (PSO) and QIEA to running and jumping locomotion tasks;
- Critical analysis, comparison and evaluation of *central pattern generators* (CPG) and *recurrent neural networks* (RNN) for locomotion control;
- Critical analysis, comparison and evaluation of *genetic algorithms* (GA), PSO and QIEA for locomotion optimisation;

- Guidance on developing physical models for use in simulations, that are capable of and likely to produce the desired movement skills;
- Analysis of the ability of the studied control methods and optimisation techniques to scale from stable quadruped to unstable biped models, with increasingly realistic actuator models.

1.4 Thesis outline

This thesis is arranged into six chapters. Chapter 1 introduces this work, with the motivation for the research in section 1.1, a statement and explanation of objectives in 1.2, a list of original contributions presented in 1.3 and an outline of the thesis in section 1.4.

Although this introduction has talked about human locomotion, the literature reviews presented in chapters 2 and 3 largely covers robotics. Applicable methods typically meet the demands of legged locomotion but other systems have also been examined. Control methods which allow motion patterns to be generated from a parameter vector are presented in section 2.1 and a comparative summary of the reviewed control methods is presented in section 2.2.

A summary of optimisation techniques identified that have been applied to locomotion in the literature is given in section 3.1 and a background to three different types or classes of optimisation algorithm is presented in section 3.2. A small review of physical simulation packages in given in section 3.3 and, following a discussion on selecting suitable physics platforms, a set of physical models are described in section 3.4 for use in experiments.

To expand upon the available optimisation methods, with recognition that evolving some patterns may be a difficult task, *quantum inspired evolutionary algorithms* (QIEA) are presented in chapter 4. After an introduction in section 4.1, binary QIEA are presented in section 4.2, along with an identified convergence problem and a proposed solution. A real-coded QIEA from the literature is given in section 4.3, and a problem in the rotation gate formulae is shown, along with an alternative solution.

Methodologies for analysing the adapted QIEA are given in section 0, with results and discussion in sections 4.5 and 4.6 respectively.

Chapter 5 presents experiments in locomotion optimisation. A bipedal vertical jump skill is developed in section 5.1 for two different platforms, in detailed bipedal models with qualitative discuss and quantitative comparison between control methods and optimisation algorithms. This approach is continued for walking gaits in quadruped and bipedal models in section 5.2 and long jumping in a biped in section 5.3. Some experiments allowed a large amount of comparison between techniques, including significance tests. An important discussion is also given into problems encountered in physical simulation, how they affected experimentation and the implications for future work.

The main content of the thesis concludes with chapter 6, with a summary and discussion in section 6.1 with reference to the objectives of the thesis, suggestions for future work in section 6.2 and a list of the author's publications in section 6.3. Chapter 7 lists the references used in this document, Appendix A presents additional results from the quantum inspired evolutionary algorithm research, Appendix B gives formulae for fitness functions used in that investigation and Appendix C presents the code for modified joint actuator functions used in the locomotion optimisation experiments.

2 Review of locomotion control

Locomotion is the process of moving an organism or synthetic creature around an environment. Artificially producing locomotion is required in a range of disciplines including robot control [16]–[19], artificial limb control [20], [21], computer animation [22], [23], and biological studies [24]. Locomotion may be needed for simulated models [25] or real world systems such as robots [26].

This chapter details a number of different techniques for control of locomotion present in the literature, and compares their strengths and weaknesses. Arguments are presented as to which methods are relevant to the research topic of this thesis, to be examined in later chapters. As can be seen from their dominance in the reference list, applications to robotics formed the bulk of this review. This is a reflection of what was found in the literature search, rather than a bias towards that field in presentation.

2.1 Control methods

2.1.1 Analytical

As was explained in chapter 1, analytical approaches to biomechanics generally involve simplification. During the literature search outlined in this chapter, analytical approaches for robotic locomotion also involved simplification in the modelling stage, and so the outline here will be relatively brief.

Legged movement starts from a simple premise - if the feet are placed in a forward moving pattern, and the rest of the body remains supported without falling to the ground, then the whole mass of the system will be moved forward continuously. Motion is therefore a combination of gait and whole (and especially upper) body stability. As bipedal motion stresses the stability constraint, it presents a challenge to locomotion

control, and was found to be the most investigated form in the literature [27]. A typical procedure for constructing bipedal motion would be:

- Plan a path to determine foot placements;
- Apply stability constraints to determine the Centre of Gravity (CoG) trajectory, based on a model of the weight distribution;
- Construct a plausible gait algorithm, addressing the double support (both feet on the ground) and single support (one foot off the ground) phases;
- Solve any remaining degrees of freedom (DoF) by any sensible manner. Methods may include copying human movement, simplifying movement, or even producing something that 'looks right'.

The first and most common stability constraint is the Zero Moment Point (ZMP) [28]. It is calculated as the point under the foot where the ground reaction force will completely negate the effects of moments and forces on the foot from the rest of the body (assuming sufficient friction). If the ZMP exists under the foot, then the system is stable, but if the calculated point is outside the foot, then the ZMP does not exist and the body (robot) will topple, rotating around an edge of the foot. The ZMP equations are used to ensure that the robot remains upright as the feet are moved. They are combined with a model of the mass distribution in order to determine a trajectory for the centre of gravity (CoG). The model is often simplified to make deriving the equations simpler. A common model is the inverted pendulum model (IPM), which has a single point mass connected to the ground by a weightless rod [29]. In two dimensions this is given as:

$$\ddot{x}_{CoG} - \frac{\left(\ddot{z}_{CoG} + g\right)}{\alpha\left(z_{CoG} - z_{ZMP}\right)}\left(x_{CoG} - x_{ZMP}\right) = 0, \tag{2.1}$$

where x_{CoG} and z_{CoG} are the x and z components of the Centre of Gravity respectively, x_{ZMP} and z_{ZMP} are the x and z components of the desired Zero Moment Point, α is a constant with value 1 in the standard inverted pendulum model, and g is the gravitational acceleration.

The gait algorithm is anything that raises a foot (now labelled as belonging to the swing leg) off the ground, moves it forwards, and places it back on the ground. Often, the mass models used assume that all of the mass is contained above the hip. Therefore, the CoG trajectory, as determined by the ZMP constraint, defines the hip angle trajectory. This is calculated to produce the CoG position relative to the hip position, which in turn is specified by the foot placements and leg joint angles. The foot trajectories are used to determine the leg joint angles and the overall motion is determined by the desired ZMP. It is located under the support foot in the single support phase and transitions to the other foot during the double support phase. Solving for the final trajectories is covered in [30].

Variations to the above procedure involve alterations to the stability constraint, alterations to the mass distribution model and alterations to the gait algorithm. Some of these variations are discussed below.

Equation (2.1) is a general form of the ZMP inverted pendulum constraint, but in many experiments it is simplified with the condition that there is no vertical movement of the CoG (i.e., $\ddot{z}_{CoG} = 0$). This is often referred to as the Three Dimensional Linear Inverted Pendulum Model (3D-LIPM). The simplicity of the 3D-LIPM algorithm made it popular in [17], [18], [18], [19], [26], [30]–[32]. In [25], a comparison was made between several different mass models. The authors found that all of the models could be written in the form of equation (2.1) but with α varying depending on which mass model was used. Since different mass models will result in varying levels of accuracy to the true model, the identification of the constant α allows the mass model to be fine-tuned. By experimentally varying α, the ZMP error can be reduced. As α multiplies the height z_{CoG}, the model was called a Virtual Height Inverted Pendulum. A simple error minimisation procedure was used to find optimal values for α for different step periods. The procedure involved incrementing or decrementing α by a fixed amount, if the ZMP error was outside a threshold interval.

To further increase the accuracy, more complicated mass models can be used, at the expense of increased complexity of analysis. For example, multi-mass models were used in [33], [34]. In [34], three different bipedal control methods were evaluated and validated by comparing them to a reference multi-mass model with ZMP constraint. The compared models used polynomial interpolation between start and end states,

actuator driving in the double-support phase, and a combined approach with added toe support and shock absorption, respectively. The toe support phase is omitted in normal ZMP based methods because it is, by definition, a failure with the foot beginning to rotate about the front edge. The authors included it to allow real-time freedom of choice for placing the landing foot. They argued this would allow for better walking on uneven ground.

The ZMP can only be used to classify a state as stable or unstable. More informative alternatives include the Foot Rotation Indicator (FRI) [35], used to determine the stability margin or degree of instability, and the Centroidal Moment Point (CMP) [36], which provides information on the whole body rotational. After analysing how the various stability constraints performed for a human gait, the authors of [36] recommended a modified FRI that was more sensitive, and that the CMP and ZMP should both be used, for human-like locomotion. A reason to improve upon ZMP control is illustrated in [37], where significant differences were found between human and ZMP gaits, such as CoG trajectory, free leg trajectory, and the position of the ZMP under the foot.

With extra legs, stability becomes less of a problem, and the focus then shifts to developing gait algorithms – methods of moving the feet in order to move the system in a particular direction. Different types of gaits for quadrupeds have been developed, to be used at different speeds, such as ambling, trotting, bounding and galloping [38]–[41]. In order to improve manoeuvrability, forward and crab (perpendicular) gaits for flat and sloped terrain were developed in [42]. Other research has looked at fine tuning the gaits, including comparing different modelling assumptions on final real-world accuracy [41], and increasing efficiency by utilising the natural dynamics of the system [43].

2.1.2 Central pattern generators

In motor skills science there was a debate over whether locomotion is reflex based or is generated internally. Experiments in the second half of the 20th century demonstrated that internal generation had to be a significant part of locomotion [44]. This was proved by severing sensory neural pathways in animal subjects and observing that they could still perform locomotion. In more recent times, dissections have enabled a reverse engineering of the neural networks that control this innate locomotion. Those networks

that have been discovered, exist in spinal regions and have therefore been called ‘central pattern generators’, or CPGs [45]–[48].

A biological CPG can be defined as a neural network that produces rhythmic pattern outputs without the need for patterned input. However, a distinction should be made between CPG neural networks and more traditional NNs, which are discussed in sections 2.1.3 and 2.1.4. In reviewing the engineering use of CPGs, it was determined that it is better to see CPGs as systems of oscillators, rather than as neural networks. In biological systems, the primary unit of rhythm is built around a pair of inhibitory/excitatory neurons that produce oscillations. A detailed examination of a biological CPG from an engineering perspective was conducted in [49] but shall not be described in detail here. The workings of the neurological systems are modelled, including membrane potentials, neuron resistance and capacitance, receptor channel dynamics, among other properties. As described below, the applied literature in locomotion control generally uses more abstract mathematical forms.

The review of CPGs here focuses on application to robotic control and generalises the concept to oscillator models. For other perspectives, including historical and biological contexts, see the reviews in [50], [51]. Some approaches, that are referred to as CPGs, do not even explicitly use differential equations, but rather use oscillators with more transparent sinusoidal forms, as discussed in the next section.

A detailed mathematical modelling of biological CPGs has been performed in [49]. The first examples of artificial CPGs described here take inspiration from biology, but without directly modelling biological processes. First developed in [52], the Matsuoka oscillating system models networks of mutually inhibiting neurons and is presented in equation (2.2). In this system, x_i and y_i are the internal and external states of oscillator (or neuron) i respectively. The model includes an adaptation, or self-inhibitory, component x_i' which was shown to help produce oscillatory behaviour, when certain conditions on the constants were met. Oscillators are linked with the weight matrix a_{ij}, b and T are timing constants, and s_i is an input from outside of the network. Mutually inhibiting pairs of neurons can conveniently be arranged into units that represent extensor/flexor muscle pairs [53], and units can then be linked to others to control inter-limb coordination, all determined by the weight matrix a_{ij}.

$$\begin{aligned} \dot{x}_i &= -x_i - \sum\left(a_{ij} y_j\right) + s_i - b x_i^{'}, \\ T\dot{x}_i^{'} &= -x_i^{'} + y_i, \\ y_i &= \max\left(0, x_i\right). \end{aligned} \tag{2.2}$$

Matsuoka oscillators have often been used for locomotion [20], [52]–[63], and as well as proving successful, they have been found to elegantly produce different gaits, such as walking, trotting and pacing quadruped gaits, simply by specifying different phase relationships [53]. Furthermore, they are capable of producing smooth gait transitions, which was dramatically demonstrated in [57] where a 2D four legged robot switched between a bipedal gait for walking, and a quadrupedal gait for climbing up a slope.

A final example of a biologically inspired CPG oscillator is the Ellias-Grossberg oscillator outlined in detail in [64] and specified in equation (2.3). This has some similarities to the Matsuoka oscillator above but slightly different combinations of contribution to the change in internal state (interpreted in [64] as the membrane potential of a neuron), from the current internal state x_i, external input I_i and interlink weight matrix D_{ij}. In this equation, A, B, C, E, F1, F2, G1, G2 are constants. Besides its initial introduction in [64], and use in controlling insect locomotion in [65], this oscillator appears to be rarely used.

$$\begin{aligned} \dot{x}_i &= -Ax_i + \left(B - x_i\right)\left[f\left(x_i\right) + I_i \right] - \left(C + x_i\right)\sum_j D_{ij} g\left(y_j\right), \\ \dot{y}_i &= E\left[\left(1 - y_i\right)\left[x_i\right]^+ - y_i \right], \\ \left[w\right]^+ &= \max\left(w, 0\right), \\ f\left(w\right) &= \frac{F_1\left(\left[w\right]^+\right)^2}{F_2 + \left(\left[w\right]^+\right)^2}, \\ g\left(w\right) &= \frac{G_1\left(\left[w\right]^+\right)^2}{G_2 + \left(\left[w\right]^+\right)^2}. \end{aligned} \tag{2.3}$$

Moving away from a direct biological inspiration, the following CPG systems are based on oscillators that are not trying to emulate some biological control process. The most obvious of these are systems that explicitly use sinusoidal equations to produce oscillations. A typical example, from [66], is shown in equation (2.4), and gives an output θ_i using an offset x_i, and amplitude r_i, and a phase angle ϕ_i for oscillator i. The

constants a_r and b_r, are used in smoothing equations to change r_i to a new value R_i, with a_x and b_x doing the same for x_i with respect to X_i. The phase angle is changed at a specified rate of ω_i plus a summation produces a link with the other oscillations based on phase difference, and interlink matrices α and β.

$$\begin{aligned}
\dot{\phi}_i &= \omega_i + \sum_j \left(\alpha_{ij} r_j \sin\left(\phi_j - \phi_i\right) + \beta_{ij} r_j \cos\left(\phi_j - \phi_i\right) \right), \\
\ddot{r}_i &= a_r \left(b_r \left(R_i - r_i \right) - \dot{r}_i \right), \\
\ddot{x}_i &= a_x \left(b_x \left(X_i - x_i \right) - \dot{x}_i \right), \\
\theta_i &= x_i + r_i \cos\left(\phi_i\right).
\end{aligned} \tag{2.4}$$

Other examples of using this type of CPG include [67]–[69], where serpentine locomotion was developed, and by using a hierarchy of oscillators, bipedal locomotion was produced in [70].

Finally in this section are presented oscillators that are expressed as differential equations, but without emulating biological processes. A typical example is the *Van der Pol* oscillator [71]–[74], shown in equation (2.5). It has a simple structure with controls for dampening μ_i, amplitude p_i, frequency g_i and offset q_i for oscillator i (although there is some interaction between the parameters). Coupling with other oscillators in the system is achieved through the matrix λ. As with Matsuoka oscillators, Van der Pol oscillators have been able to produce walking, trotting, pacing and bounding gaits [72], as well as forward jumping movements [73], with smoothly gait transitions achieved by varying the control parameters.

$$\begin{aligned}
\ddot{x}_i - \mu_i \left(p_i^2 - s_i^2 \right) + g_i^2 s_i &= q_i, \\
s_i &= x_i - \sum_j \lambda_{ji} x_j .
\end{aligned} \tag{2.5}$$

Finally, the two state variable *Hopf* oscillator is shown in equation (2.6) [75], [76]. The state variables x and y interact to create oscillations, with frequency controlled by ω. In equation (2.7) a driving term $\varepsilon F(t)$ from [77] has been added, along with a coupling matrix k [78] to connect a system of oscillators, indexed by i. Additional constants γ_i and μ_i are included to control speed of recovery for perturbation and amplitude respectively. An interesting variation in CPG constructions was presented in [77], where a series of Hopf oscillators was investigated, that included feedback terms allowing learning of an input trajectory, with each sub-oscillator matching a partial of

the input. A CPG using these combined oscillators was trained with a reference bipedal locomotion pattern, thus converting a reference trajectory into a system with limit cycle properties, so that it became resilient to perturbation.

$$\begin{aligned} \dot{x} &= -\omega y + \left(1 - r^2\right) x, \\ \dot{y} &= \omega x + \left(1 - r^2\right) y, \\ r &= \sqrt{x^2 + y^2}. \end{aligned} \tag{2.6}$$

$$\begin{aligned} \dot{x}_i &= -\omega_i y_i + \gamma_i \left(\mu_i - r_i^2\right) x_i + \varepsilon F\left(t\right), \\ \dot{y}_i &= \omega_i x_i + \gamma_i \left(\mu_i - r^2\right) y_i + \sum_j k_{ij} y_i, \\ \dot{\omega}_i &= -\varepsilon F\left(t\right) \frac{y_i}{r_i}, \\ r_i &= \sqrt{x_i^{\,2} + y_i^{\,2}}. \end{aligned} \tag{2.7}$$

2.1.3 Feed Forward Neural Networks for movement control

Feed forward NNs are limited in their ability to generate movement patterns over time. They have been used to specify parameters for an analytically derived control method, in response to control inputs encoding stair height to be climbed, in order to interpolate GA derived optimal patterns (with respect to energy consumption) [79], [80].

For actual pattern generation, *radial basis function networks* (RBFN, Figure 1) have been used to generate movement actions in response to sensory input [81]. In RBFNs the output of the input layer is fed into each neuron in the RBF layer as a vector. The neurons in this layer have a reference vector and activate depending on how close the input vector is to their particular reference. Weighted sums of the RBF neuron outputs are then fed into the output layer. The sensors in [81] gave information about the hexapod robot's current state (limb positions) and the network was trained to choose a suitable action in order to maintain a walking gait. *RBFN* were used as part of the network has neurons that compare input vectors to stored reference vectors, and respond strongly when the distance between them is below a threshold value. This enables the network to classify states, and these then feed forward towards output layers for actions.

By basing output on the current state, movement patterns are limited to those in which a one to one relationship can be established between the analysed state and suitable action.

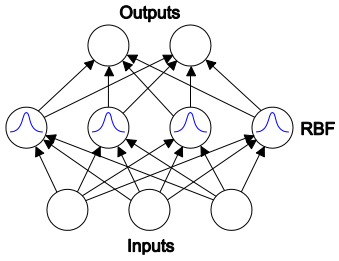


Figure 1: Radial Basis Function Network (RBFN)

By including a time signal, a feed forward NN was developed in [82] that could output actuator values as a function of time. The authors used a *Cerebellar Model Articulation Controller* or *CMAC* network (Figure 2), which is a type of associative memory network based on the cerebellum [83]. The continuous input space is divided into hyper-rectangles so that an input is located in one rectangle at any one time. Multiple layers are used with the placement of the rectangles slightly offset for each layer, so a rectangle in one layer will overlap several in the other layers. In this way one input is associated with multiple hyper-rectangles, one in each layer, but changes in the input will result in different changes in activation in each layer. Each hyper-rectangle in each layer has a weighted connection to the output neurons. The output of each node is the weighted sum of the activated rectangles, and the weights are adjusted through training.

The *CMAC* was trained to learn basic walking patterns for a hexapod, as a function of time and control variables for desired step length and walking period.

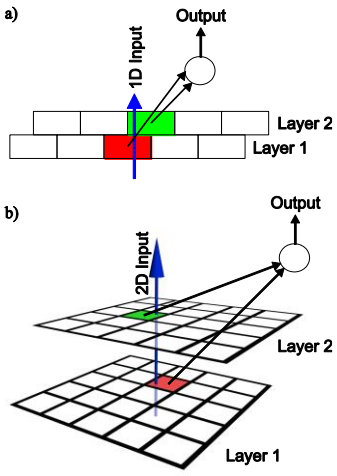


Figure 2: Cerebellar Model Articulation Controller (CMAC)

2.1.4 Recurrent Neural Network pattern generators

Recurrent neural networks (RNN) recycle information, via time delays, back into themselves, which makes them useful for processing inputs that evolve over time. Of particular interest to locomotion is that they can also exhibit limit cycle behaviour and self-generated patterns, and so are capable of producing periodic trajectories.

Upon this simple premise, more complicated networks can be developed to produce complex patterns and handle different types of input. With the RNNs outlined here there is a partnership between processing external sensor data and internally generating patterns. Inputs into the RNNs for locomotion generally consist of gait selection and sensory information. For non-recurrent networks, the inputs are vital to produce the trajectories but, for the RNN, the pattern is produced internally and modified or selected by the inputs.

A simple way of making an RNN is to recycle the output of a layer, back into the hidden layer (as additional inputs to those neurons). In *Jordan networks* (Figure 3), which were used in [84] to learn locomotion trajectories, this is done by feeding the output layer back into the network, and with *Elman networks* (used in [85] to learn trajectories for a hexapod robot), this is done by feeding the hidden layer back into itself, via time-delayed connections. In [84] accurate, fault tolerant trajectories were learned, and the system could interpolate between forms found in the training set, by varying the control inputs accordingly.

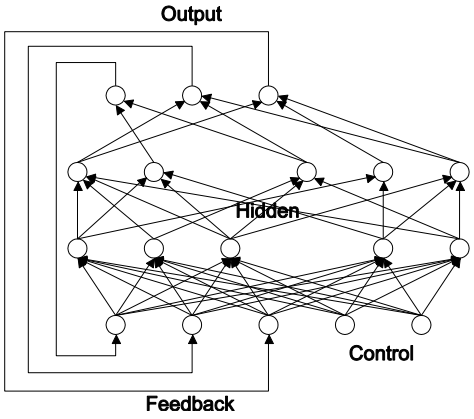


Figure 3: Jordan network

In [86]–[88], networks of *fully connected* leaky-integrator neurons were used as pattern generators (Figure 4 and Eq. (2.8)). In [86], a network of 10 neurons was used to evolve biped walking patterns. The output functions of the neurons insured changing outputs over time, even if the network was initialized with zero outputs for all neurons. In [87] a similar network was used to evolve hexapod locomotion, and in [88] another network was evolve to hunt a chemical marker in 2D space.

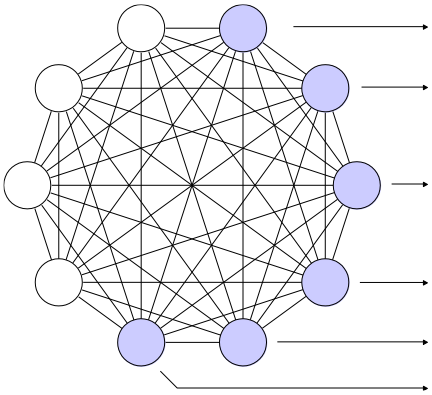


Figure 4: Fully connected neural network with six output neurons (shaded).

$$\Delta A_i = \Delta t \frac{\left(-A_i + \sum_j w_{ij} O_j \right)}{\tau_i}$$
$$O_i = \left(1 + e^{(\alpha_i - A_i)}\right)^{-1} \tag{2.8}$$

This chapter began by outlining analytically methods, utilising bespoke equations to meet the task, followed by CPGs, which although more general than analytical methods, are explicitly systems of oscillators. The neural networks presented in this and the previous section represent the most general solutions investigated for the field of locomotion control. The structure of neural networks do not impose the same constraints on function as the other methods. This is taken to the fullest extent in *reservoir networks* (Figure 5). These networks are generally large (relative to other RNNs for a given task), the structure is initialised randomly rather than having defined layers, and this combines with the size of the network to create a highly dynamic information flow. Unlike other neural networks, the weights of connections are not evolved or learned. Rather, the output neurons are evolved to 'listen' in to desired

patterns from the network. This enables linear regression methods to simplify learning. Therefore, size and processing power requirements are traded for simplicity of learning/evolving. In [89] a *reservoir network* was developed that could learn human walking trajectories captured from a motion capture system. The system was resilient to perturbation and able to interpolate between trained references, controlled by inputs.

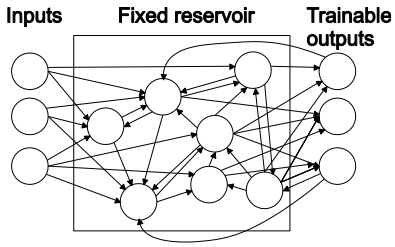


Figure 5: Reservoir neural network.

2.1.5 Rule based systems

By classifying the current state, the next action can be determined through a table lookup or rule based system. The simplicity of these approaches enables clear interpretation, and therefore the use of expert knowledge or learning/optimising techniques to specify transition rules.

If a calculation of the current state vector can be performed, with discrete values or categorisation, then a state index can be calculated and used to select an action from a lookup *table*. In [90], a transition *table* was evolved, in order to specify actions based on the current system state, for a hexapod. Body states were classified into an integer value and this was used to index a list of actions, which specified leg movements. For a given transition table, generated using a GA, the hexapod was simulated starting with each possible initial configuration, and the fitness function was the proportion of those starting positions that led to a tripod gait. The best table evolved gave stable tripod gates for 98.4% of the initial states, with the failures arising from initial symmetrical leg positions that could not lead to asymmetrical tripod gaits.

A transition *table* was also used in [91] to control a hexapod, but this time, the state value was calculated as a binary string where each bit represented either supporting (down) or not supporting (up) leg. With one leg being raised or lowered at a time, the

goal was to move through a terrain that had specified bad patches (pre-known to the algorithm). Using a graph search technique, locomotion was successfully generated and exhibited different gates.

Fuzzy logic algorithms allow decisions based on continuous data. Processing sensor information (for system state and environmental data) these systems have been used to augment analytical solutions in [33] (to modify ZMP positions to match shifting observed in human gaits) and [92] (to control four free parameters in order to improve stability and efficiency).

More directly, *fuzzy logic* controllers have been developed to fully control movement. In [93] they were used for path planning, ditch crossing, and turning for a hexapod. The controllers processed ditch distance and angle information, and outputted actions for each leg. Initially, the rules were specified by the author, but it was assumed that a subset could be more effective. To achieve this, GAs were used to prune the rules, using a fitness function developed to minimize travel time and maximize walking efficiency. Through simulation it was found that, whatever the composition of the original rule set, the GA always improved the fitness.

Others examples include bipedal stair climbing [94] and bipedal flat walking [95], as well as systems that learned during real-world performance (on-line learning) [96], [97]. A valuable property of rule based systems, including *fuzzy-logic* algorithms, is that the techniques (as expressed as rules) can easily be interpreted [94].

2.1.6 Hidden Markov models

One other specialist technique was identified in the literature – constructing *hidden Markov models* (HMM) for imitating observed movement patterns. The goal for these methods is to observe a motion patterns and then to reproduce them. Often, a human may be the source of the motion pattern to be copied and because the physical workings and capabilities of the source and target systems are different, simple copying will not work. Imitation therefore becomes a process of observation and re-synthesis. Here, observing is estimating the underlying state variables of the source system where only the output is available, and Hidden Markov Models (HMM) have been used in robotics for this task [98]–[101].

When imitating, the system first observes a movement pattern and a recognition algorithm is then used to determine if the movement is already known, in which case the pattern is used to refine the stored one. If the pattern is not recognized, then it is added to the database as a new, learned pattern. The observed kinematic or kinetic values, as well as the synthesized values, are called 'motion elements' in the HMM papers reviewed here. One motion element represents the kinematic/kinetic values at one discrete moment of time. The hidden states of the HMM provide an abstraction of the movement patterns, which can be used to re-synthesise the motion in the target system.

The first part of imitation is recognition, where a recursive algorithm calculates the probability of observing a movement pattern, if the candidate HMM was used to generate it. If this probability exceeds a threshold value (which can be varied to control grouping of similar observations), then the observation is determined to have fit the stored model and is, therefore, recognized [98], [99].

If recognition does not occur, then the next stage of imitation is learning. A new HMM is generated from the observation sequence using an expectation-maximisation algorithm (EM), such as the Baum-Welch algorithm [102].

To synthesize, or produce the movement once stored, it is usually generated stochastically from the HMM. Because of its probabilistic nature, typically, the synthesis is repeated several times, with each sequence normalized in time. The sequences are then averaged to produce a final output. Alternatively, in [99] the *Viterbi* algorithm [103] was used instead, to generate a sequence that most closely matches the observation. Finally, an error value is generated based on the difference between the synthesized and observed sequences. This error value can then be used in a learning rule to modify the matrices of the HMM and refine the stored pattern.

2.2 Summary

A continuum of approaches exists to produce locomotive movement, from those that rely fully on prior knowledge and require no training, to those that are very generic pattern generators that require intensive training or optimisation (see Figure 6 for a

relative visual depiction of algorithm optimisation demand versus constraint profile). A summary of the highlighted advantages and disadvantages of the methods presented in this chapter is given in Table 1.

For this thesis, analytical approaches are rejected, for the reasons specified in Chapter 1 – the process of simplification required for modelling reduces reliability, obscures detail and requires reworking to adapt for individual differences [11]. For similar reasons the rule based systems are also excluded as they impose a level of simplification through the selection of discrete action. Furthermore, rule based systems produce actions as a function of the current state and therefore may need additional internal state variables, or a time input (as described in section 2.1.3 for the CMAC network [82]), to have a comprehensive enough output range of patterns to meet the task goal.

Lastly, hidden Markov models are also not considered for this thesis as they have fulfilled the special task of imitation in the literature. That leaves CPGs and neural networks as applicable techniques. Because of their oscillator based design, CPGs will be best suited to cyclic behaviours. Movement skills that are non-cyclic and discrete may be difficult to realise in a CPG, unless a suitable window of time is chosen, the CPG is configured in such a way that it does not oscillate, or some multi-stage techniques are implemented. Prior to conducting the research presented in subsequent chapters, it was assumed neural networks have the potential to provide a framework than can produce both cyclic and non-cyclic movement patterns.

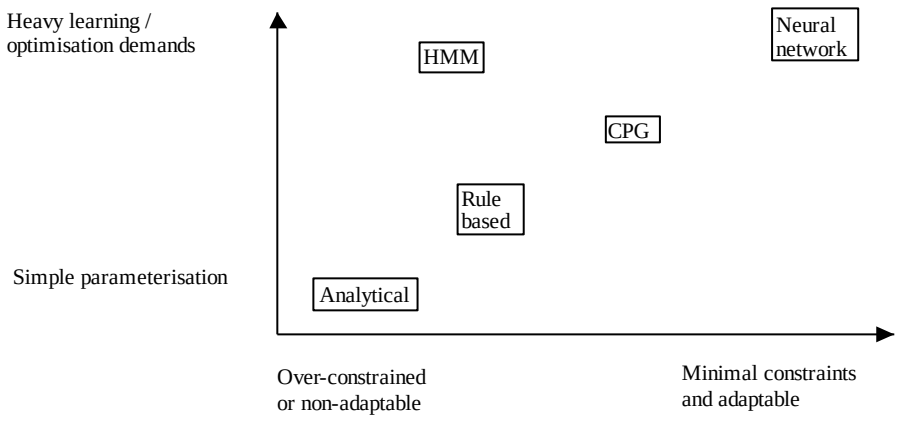


Figure 6: Constraint/adaptability and learning demands of the reviewed control methods.

Table 1: Advantages and disadvantages of the reviewed control methods.

	Advantages	Disadvantages
Analytical	• Accurate analysis can produce immediate results • Training not required so implementation in hardware can be easier • Long research history • Easy to interpret solutions	• Usually very task specific • Generally task limited with no emergent behaviours • Approximate modelling can produce inaccuracy or inefficiency • Simplification of analysis through over-constraint can lead to sub-optimal solutions • Constraints can produce 'un-natural' gaits
CPG	• Limit cycles are resilient to perturbation • Emergent behaviours • Clear parameter structure, good for optimisation • Less constrained than analytical, possible that better solutions exist	• Calibration, training or evolution often required • Less easy to interpret solutions compared to analytical • Still more constrained than some other approaches • Non-cyclic behaviour difficult to implement
Neural networks	• Least constrained approach • Capable of good limit cycle behaviour, resilient to perturbation, and able to interpolate behaviours • Emergent, new behaviours possible • Suitable for training or optimisation • Some interpretation possible with FFNNs	• Training costs can be high • Difficult to interpret method of solutions (black box solutions)
HMM	• Used for imitation tasks • Often used with a database of skills	• No emergent behaviour • Lack of resilience to perturbation • Difficult to interpret method of solutions
Rule based	• Can be very simple to implement • Easy to interpret solutions • Easy to optimise • Fuzzy systems can cope with varied environments	• Often only coarse movements can be specified • Over constrained so less optimisation potential

3 Review and discussion of locomotion optimisation and physical modelling methods

Following the presentation of locomotion control methods in chapter 2, this chapter reviews and discusses methods of parameterisation, particularly from the viewpoint of optimisation, reviewing existing techniques and presenting a background in other optimisation algorithms that could be applied to the field, in accordance with the objectives of this thesis. Then this chapter summarises physical simulation and model design used in the research presented later in the thesis. That section explains choice of software platforms, methods to design physical models and influences from the literature.

3.1 Optimisation techniques in locomotion

Although chapter 2 outlined important methods in locomotion control, little has been said about how these methods are configured for a particular application. For the non-analytical techniques, and even for several analytically derived solutions, there exist some or many parameters which will need to be specified. By evaluating a parameter set according to some constraint or fitness criteria, such parameterisation can be viewed as an optimisation process. For analytical approaches, the specification of gait form and the use of conservative stability constraints usually limits the scope of optimisation. However, there are sometimes parameters available to adjust, such as gait parameters including stride length and cycle frequency. These parameters can therefore be candidates for optimisation, according to some desired goal.

As can be seen with the formulae and structures presented in chapter 2, CPGs and NNs generally require a sizable number of parameters to be set. The goal of configuring

these parameters may simply be to produce a viable solution, or to approach an optimal solution with regards to one or more goals (e.g. energy efficiency or speed). By proper encoding of fitness functions, even the basic requirement of viability may be achieved through an optimisation process.

3.1.1 Review of optimisation goals

Achieving viable solutions is generally not a problem for analytical solutions, as the point of the analysis is to construct a successful locomotion control system. However, for the less constrained CPGs, the system may have a wider range of possible outputs than the task specific goal, and therefore configuration may be needed to produce success, such as for generating walking patterns in [60], [61]. A simpler method, if a known successful gait pattern for the physical system is available, is to train the control algorithm to match that reference. Examples for CPGs can be found in [62], [74], [77], [104].

Beyond finding solutions where the criterion is simply to produce an acceptable motion, other research has been conducted into attempting to find optimal solutions in terms of the stability of the gait, efficiency or accuracy. Improving the form, particularly dynamic stability, has been the goal for research using analytical method [25], [105], [106], and CPGs [20], [56], [59].

Energy, and the related goal of effort, feature widely as optimising them can help realising the solutions as robots in the real world. For analytical control methods, minimisation of energy expenditure was investigated in [79], [80], [105]–[108], with a CPG example in [68], and minimisation of actuating effort, determined as a function of the actuator torques, was looked at in [109], [110] for analytical control methods.

Finally, maximising speed, which is a typical performance goal usually expressed as maximum distance covered over a set time frame, was attempted in [111] for an analytical method, and in [78], [112], [113] for CPGs. Alternatively, minimised error in keeping to a desired speed can be seen in [114] for analytical, and [75] for CPG control methods.

3.1.2 Review of optimisation methods

When specifying values, the simplest technique is to input them manually. This approach has been common, for example for analytical methods [25], [105]–[107], [109]–[111], [111], [114] and for CPGs [21], [54], [55], [65], [66], [70]–[73], [115]–[119]. Analysis can provide readily interpretable parameters such as end and intermediate gait coordinates [79], [80], [108], but otherwise, successful manual specification will rely on parameter ranges being a large subset of possible inputs, there being a simple interaction of the parameters and the output (e.g., a linear relationship), a large amount of time to be available to discover suitable values, or luck, or a combination of all of those factors.

Various algorithmic search techniques have been applied to locomotion control. At the simplest level, random search for minimising speed error in an analytical method was deployed in [114] (and was found to be better in that situation than sequential programming and gradient searches), and an exhaustive search for a CPG in [68]. Next, sequential quadratic programming has been applied for analytical [105], [106], [109], [110], and CPG [74] methods.

Local search has been used in [78] where a hill climbing algorithm was used for a CPG based method, but the most common technique found was the global genetic algorithms (GA). These were applied for the analytical method in [79], [80], [108] and for CPGs in [20], [24], [59]–[62], [104], [113].

Various alternative techniques were attempted, in order to speed up the search. For an analytical method, sequential surrogate optimisation was used in [111] (using successive approximations of the objective function), and for CPGs a policy gradient search was conducted in [112] and actor/critic learning was used in [56].

Alternative techniques used include Hebbian style learning rules [75], integrated oscillator learning terms [77] (equation (2.7)), and genetic programming constructed sinusoidal systems [120] for CPGs, and Depth First Search [121] and A* search [121] for analytical path planning. Multi staged optimisation has been used to simplify the process when there is a large number of degrees of freedom [62], [104]. For example, in [42] a GA was used to find parameters for the hip joints first, then all joints in the

left leg, the whole lower body, the upper body, and then finally, the whole body (which in total had 271 parameters).

Few papers identified in the literature search compared different optimisation techniques. Although reasons were given for the choice of technique in some cases (such as actor/critic methods being used to speed up search), those reasons were untested relative to other techniques. That is why one main aim of this thesis was to compare and contrast various optimisation techniques applied to locomotion.

Using outcome measures to produce walking in an unconstrained system can be difficult. The optimisation process needs to find a workable solution from a large solution space. Using travelled distance as a fitness function was successful in [60] when combined with analysing final height, to detect possible falling, and average step length. In [61], the authors initially failed to produce a walking gait with the outcome measures of travelled distance, frequency of foot strikes, and uprightness. They presented a theory that control systems in nature may have co-evolved with the structure of the physical system. With this in mind, they added support structures to the biped that allowed it to evolve effective gaits without the risk of falling down. Although not investigated, the authors suggested that the supports could then be removed and optimisation continued.

3.2 Background on global search heuristics

Typical heuristic approaches to optimisation problems include particle swarm optimisation (PSO) [122], [123], genetic algorithms (GA) [124], [125], and differential evolution [126], [127]. However, as highlighted in the previous section, of these only GA were identified in the literature review for locomotion control. Following the objectives of this thesis, more optimisation methods were examined, with the results presented in later chapters, for use in locomotion control. This section provides a brief background on the techniques that were later examined.

3.2.1 Genetic algorithms

A genetic algorithm (GA) is a search heuristic inspired by the biological process of evolution [128]. Candidate solutions are encoded as a string of genes called a chromosome. Several chromosomes (initially randomly generated) are contained in a

population pool and each chromosome is assessed for its fitness. In the experiments presented here, a chromosome encodes the parameter set of each control algorithm, with one gene representing a parameter. The algorithm is then used in the physical simulation to test the candidate solutions and the fitness is derived accordingly to the sport skill under evaluation.

A new generation of the chromosome population is produced at each iteration. The prevalence of chromosomes and genes carried through to the next generation is related to how good their fitness was, so more information from fitter chromosomes will be present and propagated, than such from less fit chromosomes, in general. Candidate solutions are combined, inspired by breeding in nature, using cross-over reproduction operators in an attempt to build better solutions by finding combinations of good elements of multiple solution attempts. Furthermore, mutation is used to randomly alter individual genes in an additional attempt to move through and explore the search space.

3.2.2 Particle swarm optimisation

Particle swarm optimisation (PSO) [129] uses a swarm of particles to move through the search space, with each particle's position changing according to its own experience, and the experience of other particles in the swarm. By recording the best position found for each particle, and for the whole swarm or for sets of neighbouring (by index) particles, candidates for optimal solutions can be exploited. By using momentum, each particle can explore the search space and potentially find better solutions, which can then be exploited.

The candidate solutions are represented by the location of each particle in the search space with the number of dimensions equal to the number of parameters to optimise. The position $\mathbf{x}_i$ of each particle *i* is updated according to equation (3.1):

$$\mathbf{x}_i(t+1) = \mathbf{x}_i(t) + \mathbf{v}_i(t+1), \tag{3.1}$$

where $\boldsymbol{v}_i$ is the velocity of each particle *i*. The velocity for each particle *i* and dimension *j* is calculated according to equation (3.2).

$$\begin{aligned} &v_{ij}(t+1) = v_{ij}(t) + c_1 r_{1j}(t)\left[p_{ij}(t) - x_{ij}(t)\right] + c_2 r_{2j}(t)\left[l_{ij}(t) - x_{ij}(t)\right], \\ &r_{1j}(t), r_{2j}(t) \sim U(0,1), \end{aligned} \tag{3.2}$$

where p_{ij} is the best solution found so far by particle i, l_{ij} is the best solution found in the set of local (by particle index) neighbouring particles, c_1, c_2 are constants balance exploration (ability to find useful areas of the search space) versus exploitation (ability to refine solutions), and r_{1j}, r_{2j} are random numbers calculated for each iteration t. The use of a local neighbourhood characterises this as the Local Best PSO variant. The size of the neighbourhood around each particle controls how information about good solutions is communicated throughout the swarm. When using a PSO in the research presented in later chapters, it was configured as in Table 2 with standard values outlined in [130].

Table 2: PSO configuration.

c_1	c_2	$v_{ij}(t=0)$
1.49445	1.49445	0.0

3.2.3 Estimation of distribution algorithms

GA and PSO techniques evolve candidate solutions over time, sharing information across the population in order to locate regions with better fitness, and then to exploit those areas in an attempt to find the optimal solution in the search space. In contrast, *estimation of distribution algorithms* (EDA) evolve explicit probability distributions from which new candidates are sampled for each generation. A typical example of EDAs is the *population-based incremental learning* (PBIL) algorithm [131], which evolves probability strings that are sampled to produce binary candidate strings. After determining the fitness values for all of the candidates, the probability string is updated in order to increase the probability of sampling a string that was the same as the fittest candidate.

It was established in [132] that a relatively new type of optimisation algorithm – *Quantum Inspired Evolutionary Algorithms (QIEA)*, belonged to the family of EDAs. Following the objective of expanding upon the number of optimisation algorithms applied to locomotion, as outlined in the introduction, variants of QIEAs were developed and tested for real-value functions, and are presented in chapter 4. They were then compared in locomotion optimisation experiments to GA and PSO in chapter 5.

3.3 Review and specification of physical simulation

Movement optimisation can either occur in a simulated or a real environment. To apply computational optimisation to human tasks, or to robotic tasks where repeated runs are of practical difficulty, a simulated physical model is required. This section begins with a brief description of available simulation platforms, outlining their advantages and disadvantages, and concludes with a series of physical model specifications that were used in the research for this thesis.

3.3.1 Review of physical simulation platforms

The optimisation procedure adopted for this research has four components: a physical simulator where the participants, environment and props are modelled and are simulated to experience forces generated externally (e.g., gravity) and internally (e.g., muscle forces); a control algorithm that specifies desired joint torques or angles (to approximate muscle or actuator actions) for each time iteration of the physical system; an optimiser that specifies the parameters for the control algorithm; a fitness function that drives the optimisation towards a desired movement skill goal.

The control and optimisation algorithms were to be produced for this thesis, and were generally coded in C++ (for speed, and author familiarity). For visualisation many software packages exist, and different ones were used for the work in this thesis. The main choice to be made, was selection of a suitable physical simulator.

In the literature, two options were identified – a game orientated package used in [86] and a robotics platform in [31]. The game package that was used is no longer available. Game packages tend to be built for speed but are usually rigid body simulators and may set accuracy as a low priority. The robotics platform identified is useful because it has been used in other studies, and includes a bipedal robot model. Additionally, there are biomechanics packages available that include more detailed models, although the computational time will be significantly increased. A short, non-exhaustive summary of packages available is provided in Table 3.

Table 3: Non-exhaustive summary of physical simulation packages.

Type	Name	Location	Notes
Game	Newton Dynamics	newtondynamics.com	Built for speed, open source, good communication from author. Freely available
Game	Unity	unity3d.com	Contains a whole development environment so includes visualisation. Can be used for free.
Robotics	OpenHRP3	fkanehiro.github.io/ openhrp3-doc/	Used in the literature search. Includes a bipedal robot model. Stopped development in 2012. Freely available
Biomechanics	SIMM	nmbl.stanford.edu/ research/software/ index.htm	Collection of software packages. Reasonably high detail. Models provided by the user community. Freely available.
Biomechanics	LifeMOD	lifemodeler.com	High detail, soft body. High cost.

As optimisation requires multiple simulation runs, it was decided to include a game orientated physics engine for speed purposes. *Newton Dynamics* was selected as the author of this thesis was already familiar with the package, it was free to use and there was good free support available from the developer and user community. Although this research project has not tested *Newton Dynamics* for accuracy, the developer claims accurate performance due to a deterministic solver, and in turn suggests the platform can be applied to scientific research. The software has support for GPU acceleration and can be used on multiple platforms. It is designed to be integrated into other systems but sample projects exist that include visualisation, interface and logic so prototyping can be started quickly.

To offer a point of comparison with research in the literature, the popular robotics simulation platform *Open Architecture Human-centered Robotics Platform version 3* (*OpenHRP3*) [31], [133], was included in the research for this thesis. It allows external automation so can readily be used for the multiple simulations required by optimisation. It can be used for bipedal simulations and includes a humanoid model that was reasonably detailed (although distinctly robotic in nature). The main environment is typically run as a java programme hosted in Eclipse (Figure 7). This provides a user interface for visualising the simulation, loading and creating physical models, specifying control scripts, as well as other functions.

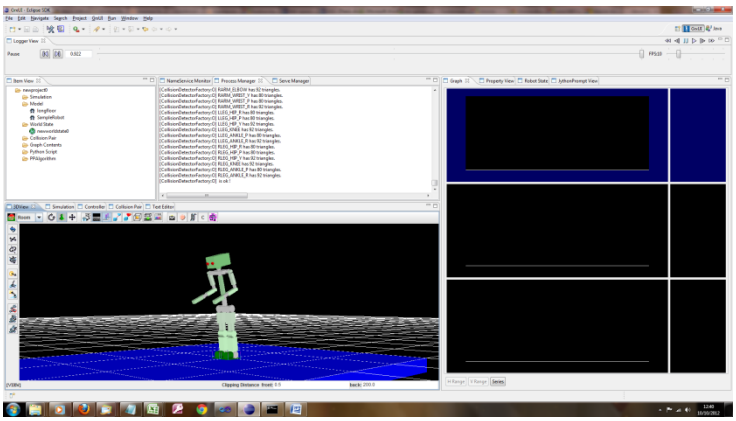


Figure 7: Open Architecture Human-centered Robotics Platform 3 (OpenHRP3).

3.4 Model design

The physics platforms used in this thesis are rigid body simulators, meaning that geometry cannot be deformed (compressed, bent, etc…). This is a good approximation for basic solid structures, especially robotic systems which are generally built using very rigid materials such as metal, but will lose some accuracy in modelling biological system which can be significantly deformed and contain tissues with viscoelastic properties [134]. Some rigid body simulators, including Newton Dynamics, can be used to simulate soft body mechanics and even fluids, using a network of particles to approximate volumes, but this advanced usage was not performed for this thesis. All models used in experiments consisted of rigid geometry parts (cuboids and spheroids) connected with mathematical models of joints.

In addition to the bipedal robot from OpenHRP3 [31], which was a relatively high degrees of freedom model, a set of models were designed to be used in Newton Dynamics. They are described in the following subsections.

3.4.1 Simple quadruped model

To provide an option that would present the least difficulty to optimise locomotion patterns for, a quadruped was designed. Quadrupeds are generally more stable than bipeds and so presumably gaits are easier to find. The system described here is statically stable (will remain still without joint activation) in its default stance, and it was observed

in testing that the model would only fall over if either the rear legs or front legs act as a pair, moving in the same direction sufficiently to flip the model. The added stability of this design provided a solid testbed with which to compare control and optimisation techniques.

The quadruped was constructed with cuboids, hinge joints and linear slider joints. The main body was situated at the top, supported by four legs at each corner, with each leg consisting of an upper and lower part. The upper and lower sections of each leg were joined with a linear slider actuator. The role of this joint was to lift the leg up and place it back down onto the floor and the legs were attached to the main body using hinge joints, allowing forward and back rotations. There was no special design for the feet, and default friction coefficients were used in the physical simulator. The visual design for the quadruped is given in Figure 8 and a specification of the physical attributes of each body part is given in Table 4.

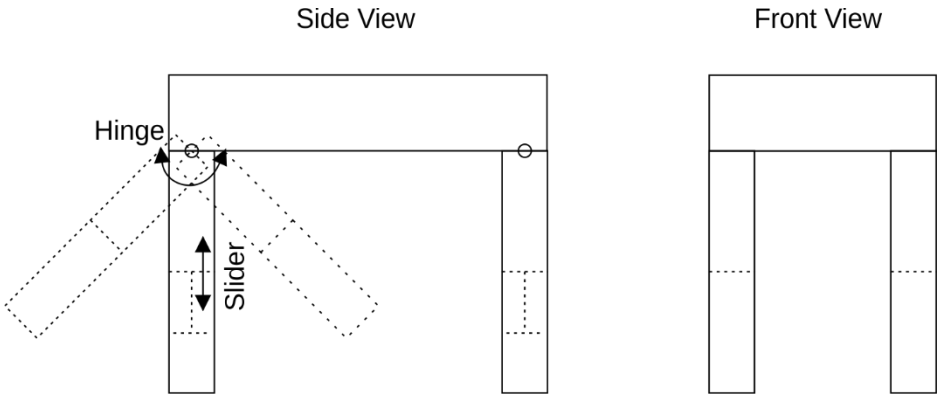


Figure 8: Schematic of the quadruped design.

Table 4: Quadruped part specification.

Part	***x* size (m)**	***y* size (m)**	***z* size (m)**	**Mass (kg)**
Main body	0.30	0.10	0.50	15.000
Upper leg	0.06	0.16	0.06	0.576
Lower leg	0.06	0.16	0.06	0.576

3.4.2 Simple biped model

Based on the biped design from [86], a simple biped model with a small number of parts was included to investigate bipedal gaits with minimal degrees of freedom. Cuboids were used for the body parts and hinge joints connected parts together. The original inspiration for the design from [86] was used for walking gaits, and included very little upper body structure. The system contains just two legs and a short torso, and used a constant density to decide masses. It was later found that the performance of the system, and ability to form suitable gaits, was heavily dependent on the joint actuation model. This is discussed in chapter 5. The schematic of the simple biped model is given in Figure 9 and part sizes are given in Table 5.

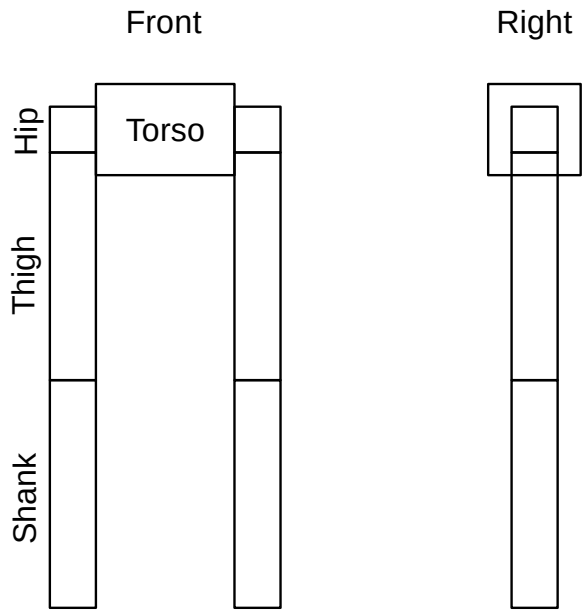


Figure 9: Simple biped schematic

Table 5: Simple biped specification.

Part	***x* size (m)**	***y* size (m)**	***z* size (m)**	**Mass (kg)**
Torso	0.300	0.200	0.200	12.00
Hip	0.100	0.100	0.100	1.00
Thigh	0.100	0.500	0.100	5.00
Shank	0.100	0.500	0.100	5.00

3.4.3 Detailed biped model

This model was designed to investigate producing movement in more complicated systems. The primary goal was to increase the number of body parts, thereby increasing

the degrees of freedom and, potentially, making movement control more difficult. Dimensions were specified in an ad-hoc fashion, with sizes constructed to give a visually humanoid structure, although this was done by eye rather than using anthropometric data. Initially masses were specified using a constant density but this criterion was soon broken as experiments into the dynamics of the body were conducted in the physical modelling software. When testing out the behaviour of joint movements, parts were added and their size and mass modified, until satisfactory movement patterns were produced when manually testing joint actuation, and no wild behaviour (unrealistic explosive movements) was observed in early test optimisation runs.

Articulation was provided through hinge joints. Additional body parts (neck, shoulders, abdomen, hips and ankles) were used to enable rotation around two axes by joining a pair of orthogonal hinges. The design is shown in Figure 10 and part sizes are shown in Table 6.

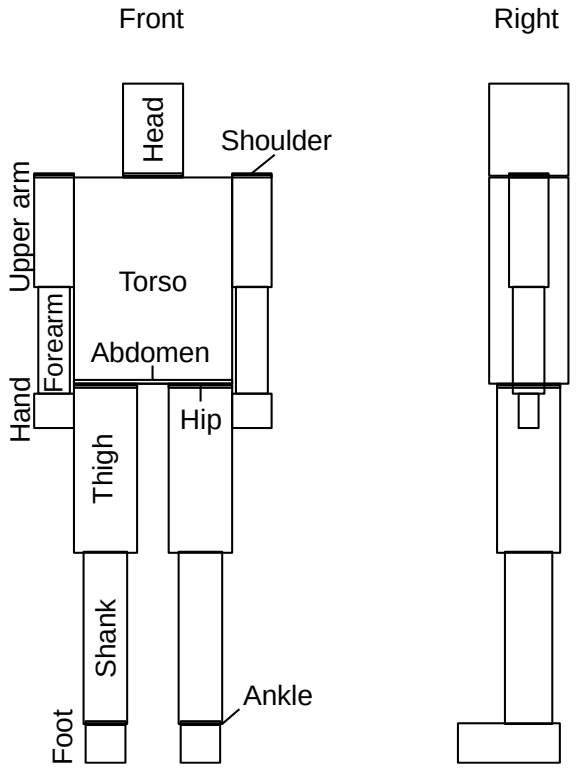


Figure 10: Detailed biped schematic

Table 6: Detailed biped specification.

Part	***x* size (m)**	***y* size (m)**	***z* size (m)**	**Mass (kg)**
Head	0.152	0.232	0.200	4.69
Torso	0.400	0.520	0.200	29.80
Upper arm	0.100	0.282	0.100	1.83
Forearm	0.080	0.269	0.080	1.13
Hand	0.100	0.086	0.050	1.00
Thigh	0.160	0.422	0.160	9.67
Shank	0.110	0.434	0.120	3.03
Foot	0.100	0.100	0.258	0.86
Neck	0.150	0.010	0.200	0.21
Shoulder	0.100	0.010	0.100	0.07
Abdomen	0.400	0.010	0.200	0.60
Hip	0.160	0.010	0.160	0.24
Ankle	0.100	0.010	0.260	0.10

3.5 Summary

A range of optimisation or learning techniques were identified in the literature. The main aim of this thesis is to attempt to locate optimal solutions, rather than find accepted solutions quickly. For that reason globally orientated searches, rather than gradient based solutions will be favoured and only GA fits this requirement from the literature review. Therefore, this thesis will examine other techniques, and compare them to GA, to establish the most appropriate techniques for the domain. A background to PSO and EDAs was presented, and they will be further discussed, developed and implemented in subsequent chapters.

A small number of physics software platforms were surveyed, encompassing fast game orientated systems, purpose specific robotics platforms and biomechanics software. A game focussed package, Newton Dynamics, was selected for being free, fast and having good support, while OpenHRP3 was selected as an example of a robotics platform popular in the literature.

To experiment with different locomotion tasks, a set of physical models were developed – a quadruped model with good stability, a simple biped model and a more detailed biped model. These models were designed to provide differing levels of difficulty for optimisation, in order to establish a relatively simple starting point, and the option to develop techniques for increasingly complex models.

4 Quantum inspired evolutionary algorithms

Given the complexity of physical models and tasks which this thesis investigates, it is possible that the search space may contain a very small area of successful parameter vectors. It was therefore one of the objectives of this thesis to apply different types of optimisation algorithms, in order to expand the knowledge of what may work best for this type of problem. In section 3.1 the well-established optimisation techniques *GA* and *PSO* were described, as well as a class of algorithms called *estimation of distribution* algorithms (*EDA*). In this research, the opportunity was taken to examine a new type of *EDA* optimisation algorithm called *quantum inspired evolution algorithm* (*QIEA*). As well as implementing versions from the literature, changes were made to improve *QIEA* performance when applied to real-value functions and engineering problems, and the background, results and discussion are presented in this chapter.

4.1 Introduction

In 2002 a new optimization algorithm was presented [135], that took inspiration from quantum computing to evolve a probability distribution, which in turn was used to search a solution space. The method used a string of quantum bits (Qbit), each storing the probability of sampling a one or a zero. Successive sampling of the string produced a series of candidate binary solutions. If any of these were found to be an improvement, the underlying Qbit probabilities were adjusted to make the candidate more likely to appear in successive samples. A detailed explanation of the algorithm is presented in section 4.2.

Originally, this quantum-inspired evolutionary algorithm (*QEIA*) was applied to the *Knapsack problem* - a binary combinatorial optimisation problem [135], and then

modified versions were applied by others to *OneMax*, *Noisy-flat* and *NK-landscapes* [132], neural-network training [136], [137], and networking [138]. In [132] it was argued that *QIEAs* should be categorised as multi-modal *EDAs*. As discussed in section 3.2.3, *EDAs* evolve probability distributions, as do *QIEAs*, and include popular examples such as *PBIL* [131], *cGA* [139] and *UMDA* [140]. A comparison of *EDAs* with respect to *QIEAs*, the multi-modal nature of *QIEAs* and the distinguishing features of *QIEAs* are given and discussed in [132].

Although some attempts have been made to apply binary *QIEA* to real-value problems [141], most applications to such tasks have used real-value *QIEA* [142]–[147]. These algorithms took, at least superficially, the concepts of superposition and quantum rotation gates that were introduced with the binary *QIEA*, and adopted them for application to real-value problems. However, when reviewing them a number of problems were encountered. Many were incompletely described and could therefore not be reproduced, one was trivial to implement [142] but performed extremely badly on a set of multimodal mathematical test functions, and of greatest concern, one paper [143] claimed superior performance to another optimization algorithm that was later found to not have performed as well as claimed [148]. A second issue, more of a philosophical concern than a practical problem, is that in making the adaptation to real-value problems, the purity of the original quantum inspiration (that are naturally applied to binary problems) may be lost. These concerns are discussed in sections 4.3 and 4.6. Various attempts at a real *QIEA* can be found in the literature, including [145]–[147], and [149] presents a review of both binary and real *QIEA*. In this investigation I have chosen [150] to build a real-coded *QIEA* upon as it performed the best in initial tests and contained features common to many real *QIEA*.

The goals of the research presented here were to see how the *Classic* version [135] of the binary *QIEA*, as well as a representative real *QIEA*, would perform on a number of recent benchmark test functions and several real-world problems, and to investigate, design, and develop modified binary and real *QIEA*, proposed to improve the performance of these approaches in terms of convergence and accuracy.

In sections 4.2 and 4.3 the binary and the real *QIEA* under investigation are presented, including the proposed modifications. Section 0 outlines the methods used for testing,

and section 4.5 presents the obtained results, with a conclusion and discussion in section 4.6.

4.2 Binary *QIEA*

This section presents the original binary quantum inspired evolutionary algorithm (*bQIEA*) [135], along with a preliminary investigation highlighting arising problems when applying it to real-value tasks. Then a modified method designed to tackle these issues is introduced.

4.2.1 Classic QIEA

The original *QIEA* [135], here labelled *Classic*, contains the core properties of *QIEA*: *Qbit* sampling; and the rotation gate operator. Unlike a traditional binary evolutionary algorithm, *Classic* stores a string of probability values called Qbits. For each individual i in a population of size p, Qbit value $Q_{ij}(t)$ gives the probability of sampling a zero or one for bit j (from a string with length of N bits) at iteration t. Through repeated iterations of sampling, the same Qbit value can be used to sample a sequence of random binary values. If a Qbit has a value of 0.5 (highest entropy), both one and zero have an equal chance of being sampled. A Qbit value near 1.0 favours sampling 1s, and a value close to 0.0 favours sampling 0s.

Even in the absence of evolution of the chromosomes, *Classic* will continue to produce different candidates for the fitness function, unlike a traditional evolutionary algorithm. The combination of probability and sampling is inspired by the quantum computing principal of superposition. Superposition is the ability of a Qbit to hold multiple states simultaneously. The string Q_i therefore provides a probability distribution function for generating candidate solutions C_i at each iteration.

While random sampling allows the solution space to be searched, the Qbits need to be changed in order to localise and refine the search. By interpreting the Qbit probability as an angle, a quantum logic gate called a rotation gate (as given in equation (4.1)), can be used to adjust the probabilities. This simply shifts the angle, and therefore the probability, one way or the other. By using the best solution found so far (called the attractor A_i) for an individual, this gate can be made to rotate towards a position that reinforces the attractor probabilities, if it is still the best solution, or away, if the current

candidate is better (Table 7). The magnitude of rotation $|\Delta\theta|$ is fixed to $\pi / 100$ and rotation saturates at the extremes, as given by equation (4.2).

Information is distributed around the population via the attractors *A*. Every *G-th* iteration, a global migration is performed, where the best attractor in the population is copied to all individuals. Every *L-th* iteration, a local migration is conducted, where the best attractor in a subset of the population is copied to the whole subset. For the investigations presented here, *G=20*, *L=1* (meaning improvements to attractors were copied to subsets at the end of each iteration), and the number of subset groups was assumed to be *5*. Pseudo-code for the *Classic* approach is given in Algorithm 1.

$$\begin{bmatrix} \alpha_{ij}(t+1) \\ \beta_{ij}(t+1) \end{bmatrix} = \begin{bmatrix} \cos(\Delta\theta) & -\sin(\Delta\theta) \\ \sin(\Delta\theta) & \cos(\Delta\theta) \end{bmatrix} \begin{bmatrix} \alpha_{ij}(t) \\ \beta_{ij}(t) \end{bmatrix}, \quad Q_{ij} = \sin^2(\alpha_{ij}),\ \beta_{ij} = \sqrt{1-(\alpha_{ij})^2}. \tag{4.1}$$

Table 7: Direction of rotation gate.

A_i^j	C_i^j	$f(A_i)$ better?	$sign(\Delta\theta)$
0	0	either	n/a
1	1	either	n/a
0	1	yes	-
1	0	yes	+
0	1	no	+
1	0	no	-

$$Q_{ij}(t+1) = \theta_{ij}(t+1) = \begin{cases} \min\left(\theta_{ij}(t) + \Delta\theta, \frac{\pi}{2}\right), & \Delta > 0 \\ \max\left(\theta_{ij}(t) + \Delta\theta, 0\right), & \Delta < 0 \end{cases}, \tag{4.2}$$

where $\Delta\theta$ is the size and direction of rotation.

Algorithm 1: Pseudo-code for the Classic algorithm.

```
Initialise each Q_i with each bit Q_ij=π/4
Initialise each A_i with random strings
Initialise each Qbit of each string to equal π/4 so that sin^2(Qbit)=0.5
while not termination condition do
    for all i∈[1,p]
        sample new C_i from Q_i
        evaluate fitness of C_i using a binary to real mapping
        for each t∈[1,N]
            if f(A_i) is better than f(C_i) then
                select a rotation direction that would
                reinforce A_ij
            else
                select a rotation direction that would
                move away from A_ij
            end if
            update Q_ij with rotation gate
        end for
        if f(C_i) is better than f(A_i) then
            A_i= C_i
        end if
    end for
    every L iterations perform local migration
    every G iterations perform global migration
end while
```

4.2.2 Application to real-value problems and convergence issues

In this investigation, for the binary optimization algorithms, real values were encoded using a simple scheme. Binary strings of length 24 bits were used to generate numbers in the $[0, 2^{24}-1]$ range, which were then linearly mapped onto the domain for the fitness function being optimized.

An initial application of *Classic* to real-valued problems highlighted a convergence issue. A plot of a typical evolution is shown in Figure 11a, where bits for one real value are shown with most significant bits to the left. Numbers shown are iterations indices. Light values are unsaturated, dark blue are saturated near to zero, red are saturated near to one. The plot shows that the least significant Qbits (LSBs) were saturating before the most significant Qbits. Once a Qbit saturates, it will no longer evolve because sampling will continuously produce ones or zeros, depending on which end of the scale the Qbit has saturated to. This means that the LSBs had become randomly fixed relatively early on in the optimization, thus preventing fine scale exploitation.

For reasonably smooth search spaces, the early stages of the search should focus on finding the general locations of extrema, rather than refining solutions to a precise position. During this phase, the fitness function will be affected more by large movements than by small ones. With a binary representation, this will manifest in the most significant bits (MSBs) dominating the search, as changes to them are likely to find larger improvements to the fitness than changes to the LSBs.

Therefore, in the early stages, the LSBs provide little selection pressure, and so random values for these bits will be tolerated, while the MSBs are optimised. The early evolution of the LSBs can therefore be modelled as random walk processes. Given that, it would be reasonable to expect several bits to have deviated substantially from the neutral middle probability position. Furthermore, the process is reinforced - as an attractor adopts a solution, shifts in the random walks will make producing a zero or one more likely, and this will lead to the random walk being attracted to a probability of zero or one respectively. By the time the MSBs have been optimised, it is likely therefore that the LSBs have saturated their probabilities.

4.2.3 Improved bQIEA convergence performance for real value problems – HSB (Half Significant Bit).

One possible solution to these convergence problems was presented in [141], where the rotation gate operator had limits imposed that were slightly within the zero to one range. This meant that, even late in the evolution, it is always possible to sample new bit values as the Qbits never completely saturate.

This type of scheme was tested but the obtained results were mixed at best. As an alternative, presented here is a method that directly addresses the problem of LSB premature convergence. When rotating a Qbit, a limit is imposed upon the range that it can move to, based on the current value of a more significant bit, so that it cannot move to a more extreme value. This has the effect of delaying large movements in the LSBs until the MSBs have saturated.

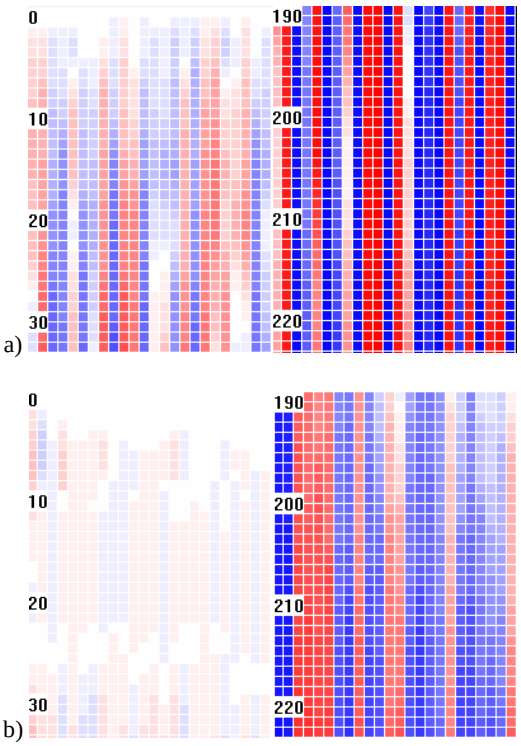


Figure 11: Evolution of Qbit probabilities on Griewank function using (a) Classic and (b) HSB algorithms.

Initially, using the more significant immediate neighbour bit as the limiting condition was tested, but eventually it was found that picking a bit index that was half the position value of the Qbit being rotated (assuming bit index zero is most significant) was better. This is a somewhat less aggressive limiting condition presenting a compromise between premature convergence and overly slow convergence. The adjusted formula for the rotation is given in equation (4.3), with the general algorithm code staying the same as for *Classic* (Algorithm 1). This modified algorithm is called *HSB* (*Half Significant Bit*) in this thesis, and preliminary results of an evolution are shown in Figure 11b.

$$Q_{ij}(t+1) = \theta_{ij}(t+1) = \begin{cases} \min\left(\theta_{ij}(t) + \Delta\theta, \pi/4 + |\theta_{ih}(t) - \pi/4|\right), & \Delta > 0 \\ \max\left(\theta_{ij}(t) + \Delta\theta, \pi/4 - |\theta_{ih}(t) - \pi/4|\right), & \Delta < 0 \end{cases}, \quad \text{where } j > 0, h = \text{floor}(j/2). \tag{4.3}$$

4.3 Real *QIEA*

In order to apply *QIEA* to real-value problems, numerous attempts have been made to develop real *QIEA* (*rQIEA*) [149] and so were included in this investigation. A simple attempt at this is shown in [142] where the rotation angles from the *Classic bQIEA* are re-interpreted as actual solutions. This approach was very simplistic and arbitrary, and completely failed in all of our initial testing. Searching for other *rQIEA* in the literature was made difficult by poorly described methods or suspect data. However, one algorithm called *RCQIEA*, presented in [150], was well defined and so was included in this study, along with a modification.

4.3.1 The RCQIEA algorithm

Whereas *Classic* produces fresh solutions at each generation, *RCQIEA* stores and updates a candidate solution. *Classic* takes the inspiration of superposition and uses it to evolve a probability density function (pdf), as described by the probability angles for each bit. By not doing this directly, *RCQIEA* begins to move away from the original quantum metaphor. However, as described below, the generation of new candidates through creep mutation, can be seen as using the candidate as a string of mean values for an evolving pdf.

At each iteration, a set of offspring O_j is generated from each individual's candidate C_i using creep mutation with variances stored in a string V_i. The values in V_i are stored as angles and transformed into a pair α_i and β_i in the same way as for the *Classic*. The offspring are generated in two subsets: one using α_i for the variances; and one using $\beta_i/5$, to allow for both fine and coarse searching. The offspring are tested for fitness and if one is found to be better than the current candidate, it replaces that candidate. Otherwise, a rotation gate is applied to the variance angles in the same way as in *Classic*, but with a rotation step given with equation (4.4).

$$\Delta\theta = \mathrm{sgn}(\alpha\beta)\,\theta_0 \exp\left(\frac{|\beta|}{|\alpha| + \gamma}\right), \tag{4.4}$$

where α and β are the angles as defined in equation (4.1), and θ_0 and γ are constants.

A cross-over operator is also applied during the evolution. For this investigation, it was applied four times during the course of each run ($G=N/4$). The pseudo-code for the *rQIEA* presented here, is given in Algorithm 2.

4.3.2 Problems with rotation gate

In [150], the constants for equation (4.4) were specified as $\theta_0=0.4\pi$ and $\gamma=0.05$. In the testing, the *RCQIEA* performed well for many functions. However, values of large magnitudes for $\Delta\theta$ were detected, which suggested a problem with the behaviour of the rotation gate. For example, if the angles are $\alpha=0.01$ and $\beta=0.99995$ (satisfying $\alpha^2 + \beta^2=1$) then equation (4.4) produces a value for $\Delta\theta$ in excess of 2.0e7. As a rotation angle in this context, such a magnitude for $\Delta\theta$ does not make sense, as it represents many complete rotations in one iteration. In effect this leads to somewhat random updates of the angle variables, and in turn, the variances for the creep mutation.

4.3.3 SRQEA – fixing RCQIEA

To alleviate this problem, a modified version of the rotation gate was developed, keeping the rest of the *RCQIEA* algorithm (Algorithm 2), called *Stepwise Real QEA* (*SRQEA*). The change rotates the angles by a constant magnitude in the rotation gate, as shown in equation (4.5).

$$\Delta\theta = \mathrm{sgn}(\alpha\beta)\,\pi/250. \tag{4.5}$$

This was motivated by making the update similar to the constant step size used in *Classic*. A step size analysis was performed with the results presented in Table 8. A range of parameter values were used to optimise test functions presented in section 4.4.1 and the parameter value that produced the minimum value for each function is listed in the table. Some functions had the same minimum value with different parameter settings, and these were separated using the SP metric outlined in section 4.4.3. The results gave a spread of values, and so a compromise choice was made of $\pi/250$.

Algorithm 2: Pseudo-code for RCQIEA

Initialise each C_i with random values

Initialise each V_i with random values

Evaluate fitness $f(C_i)$ for each individual

while not termination condition **do**

for all $i \in [1,p]$

construct two sets of offspring O_j from C_i using creep mutation from a normal distribution with variances V_i. One set uses the α_i angles, one uses the β_i angles, scaled for coarse and fine search respectively

for each offspring

if $f(O_j)$ is better than $f(C_i)$ **then**

replace C_i with C_j

else apply rotation gate to V_i

end if

end for

end for

adjust coarse and fine search scale factors over course of run to move towards finer search at the end of the simulation

every G iterations perform crossover mutation

end while

4.4 Numerical Simulation

Each algorithm was tested against several fitness functions. In accordance with the procedures outlined in [9], functions were tested with 10, 30 and 50 dimensions (except for the real-world problems which had specific dimension requirements), and each optimization run was performed 51 times. The number of function evaluations was limited in each run to 10000 x number of generations. Given that more than one function evaluation per generation was performed for the *rQIEA*, their generations per run were adjusted accordingly.

Table 8: Step size analysis for SRQEA

Function – 30 dimensions	SRQEA / π
Sphere	0.001 (SP)
Rotated high conditioned elliptic	0.0064
Rotated bent cigar	0.0010
Rotated discuss	0.0019
Different powers	0.0013 (SP)
Rotated Rosenbrock	0.0010
Rotated Schaffers F7	0.0064
Rotated Ackley	0.0010
Rotated Weierstrass	0.0052
Rotated Griewank	0.0070
Rastrigin	0.001 (SP)
Rotated Rastrigin	0.0010
Non continuous rotated Rastrigin	0.0010
Schwefel 7	0.0040
Rotated Schwefel 7	0.0019
Rotated Katsuura	0.0046
Lunacek bi-Rastrigin	0.0031 (SP)
Rotated Lunacek bi-Rastrigin	0.0052
Rotated expanded Griewank Rosenbrock	0.0061
Rotated expanded Schaffers F6	0.0010

4.4.1 Test functions

Firstly, a set of traditional, basic functions, was taken from the first 13 presented in [8] (Appendix B). A second set of more complicated functions was added from the first 20 functions defined in the CEC-2013 specification [9] (Appendix B). These are based on the traditional functions but are highly modified and transformed, including application of rotations. It should be noted that both sets share one function in common – the *Sphere* function. For this function, presentation of the results is duplicated in order to be consistent when comparing to other published results. Finally, real-world problems from CEC-2011 [7] were added: *frequency modulated sound wave matching*; *atom configuration*; and *radar waveform parameter optimisation*.

The frequency modulated sound wave matching problem optimises the constants of equation (4.6), so that the output of the wave, measured for integer t = [0 , 100], where $\theta=2\pi/100$, matches the output of equation (4.7).

$$y(t) = a_1 \sin\left(\omega_1 t\theta + a_2 \sin\left(\omega_2 t\theta + a_3 \sin\left(\omega_3 t\theta\right)\right)\right), \tag{4.6}$$

$$y(t) = 1.0 \sin\left(5.0 t\theta - 1.5 \sin\left(4.8 t\theta + 2.0 \sin\left(4.9 t\theta\right)\right)\right), \tag{4.7}$$

where α and ω are the constants to be optimised.

The *Lennard-Jones* atom potential configuration problem, aims to minimise the potential energy V_N of a set of N atoms with position $p_i = \{x_i, y_i, z_i\}$ according to equation (4.8).

$$V_N(p) = \sum_{i=1}^{N-1} \sum_{j=i+1}^{N} \left(r_{ij}^{-12} - 2r_{ij}^{-6} \right), \quad r_{ij} = \left\| p_j - p_i \right\|_2 . \tag{4.8}$$

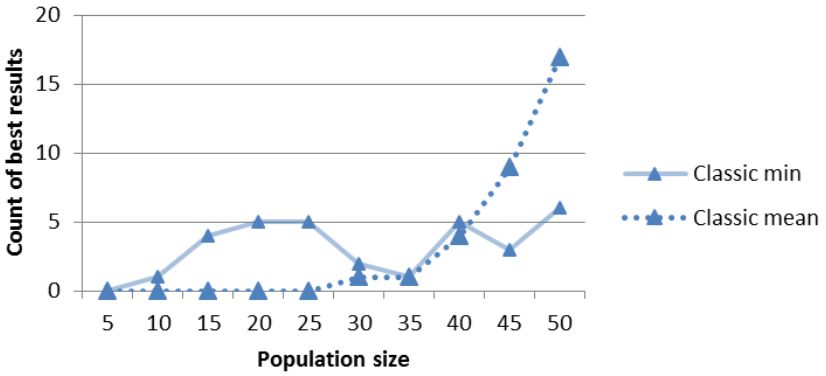
Finally, *the radar polyphase pulse design problem* seeks to minimise a function *f(x)* based upon set of *n* parameters $x=\{x_1, \ldots, x_n\}$ according to equation (4.9).

$$\begin{aligned} & f(x) = \max\{\phi_1(x), \ldots, \phi_{2m}(x)\}, \\ & \phi_{2i-1}(x) = \sum_{j=i}^{n} \cos\left(\sum_{k=|2i-j-1|+1}^{j} x_k \right), i = 1, \ldots, n, \\ & \phi_{2i}(x) = 0.5 + \sum_{j=i+1}^{n} \cos\left(\sum_{k=|2i-j|+1}^{j} x_k \right), i = 1, \ldots, n-1, \\ & m = 2n-1. \end{aligned} \tag{4.9}$$

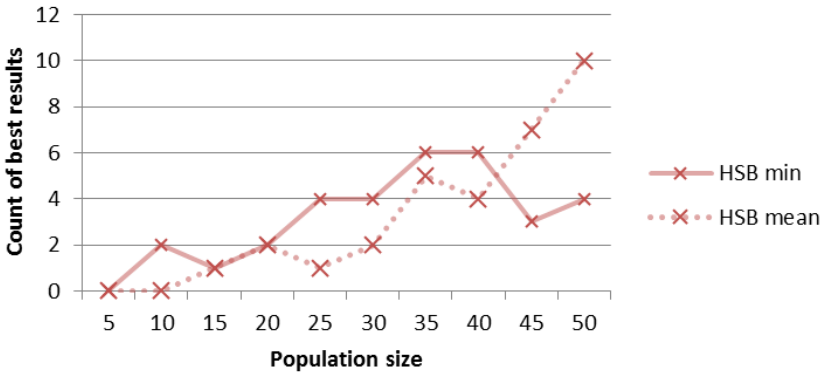
4.4.2 Population size analysis

Before conducting an extensive evaluation of the proposed methods, an investigation into choosing a suitable population size was conducted. An initial run for 30 dimensions was performed for the optimisation algorithms on the non-real world functions, with a series of different population sizes being used. The number of individuals ranged from 5 to 50, in increments of five, but the total number of functions evaluations was kept to 300000. After running the simulations, the number of times an algorithm had a best performance (assessed just for that algorithm) was counted for each population size. A best performance was assessed according to minimum values and mean values for two separate analyses, and occurred when the value was the lowest, or equal lowest, for that fitness function across all of the different population sizes tested for the optimisation algorithm. The results according to the best minimum and mean values found are shown in Figure 12. Results for the *bQIEA Classic* and *HSB* are shown in Figure 12a and Figure 12b respectively, and results for the *rQIEA RCQIEA* and *SRQEA* are shown in Figure 12c and Figure 12d respectively.

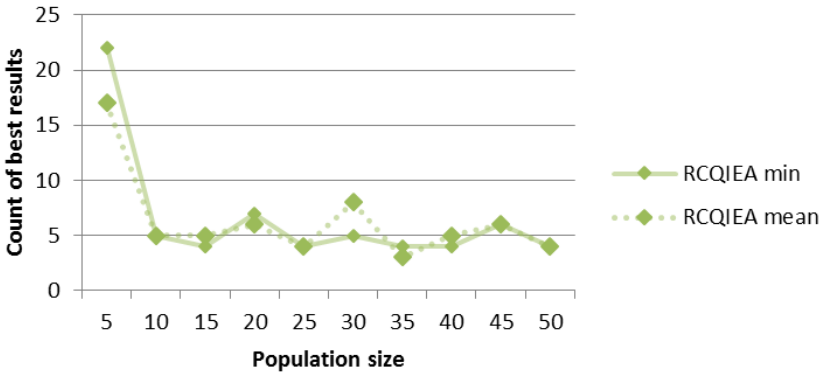
a)



b)



c)



d)

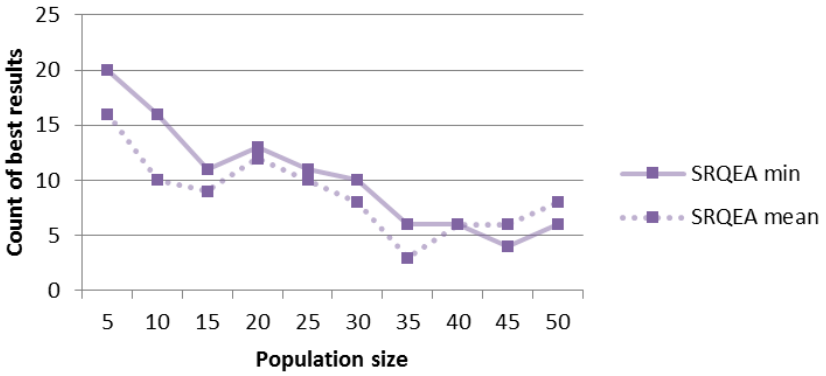


Figure 12: Population analysis for the QIEA: a) Classic; b) HSB; c) RCQIEA; d) SRQEA.

Generally, the *bQIEA* performed better with higher population sizes, while the *rQIEA* were better with smaller population sizes. For *Classic* (Figure 12a), the best minimum values were found more often with a population size of 50, with an additional peak at 20/25, while *HSB* (Figure 12b) had a peak at 35/40 but reasonable performance from 25 to 50. When looking at the mean performance, both *bQIEA* improved with increasing population size, with the best being 50 for both. After combining these results, it was chosen to proceed with 50 individuals for both *bQIEA* algorithms in the later simulations and analysis. These results suggest bQIEA are biased towards exploitation and therefore require a larger population size to achieve good exploration.

For both *rQIEA*, the results (Figure 12c and Figure 12d) were very clear – a population size of five performed the best for both minimum and mean values, and so was used for subsequent investigations. *RCQIEA* had a sharp drop-off in performance above five, while *SRQEA* had a smoother decline with increasing population size. In contrast to the findings for *bQIEA*, this suggests that the *rQIEA* have relatively good exploration, so benefit from a small population in order to improve exploitation by increasing the number of function evaluations per individual.

4.4.3 Performance metrics

1) Summary statistics

To present a basic analysis and compare across publications, summary information is generated from error values (from the known minimum value) or absolute values if the global minimum is unknown. From the raw data, simple statistical measures such as minimum, mean and standard deviations are calculated and summarised, with lower values for each being preferred in the comparisons.

2) Success Rates

Using metrics introduced in [151], a success rate and measure of time taken by the run to succeed (converging to a minimum) are calculated. Success Rate (SR) is calculated as the number of successful runs divided by the total number of runs. A run is regarded as successful if it finds an error below a threshold (as defined in results section, which can vary depending on the particular analysis or comparisons being made).

3) Success Performance

To measure the speed at which an algorithm obtains good results, a metric called Success Performance (SP) is calculated. This is defined as SP = (SNFEs)*(number of total runs)/(number of successful runs), where SNFEs is the average number of function evaluations required by each successful run to reach the tolerance. A lower value of SP is preferred because it indicates a better combination of speed and consistency for the algorithm.

4) Timeline plots

In order to analyse the behaviour of the algorithms, graphical representations of their evolution are produced for every test function. Across all runs, for each iteration the mean error is calculated and plotted. The time is normalised with respect to the number of function evaluations in the [0, 1] range so that the performance of each algorithm can be compared directly.

5) Empirical cumulative probability distribution

Performance across all functions is summarised using the empirical cumulative probability distribution function (ECDF) presented in [152]. An ECDF is constructed by firstly determining the performance of each algorithm on each test function, by comparing its mean error *ME* with the mean error achieved by the best algorithm, and formulating a normalized mean error *NME* (equation (4.10)). Then, the distribution is formed by counting, for each value χ in the domain of the distribution, how many normalized means (across all test functions) were obtained below χ (equation (4.11)). Normalizing and plotting these values produces a graph where superior algorithms reach the top of the chart faster than less well performing algorithms. In this analysis, all the test functions were included, as well as additional graphs for subsets (traditional, CEC-2013 and real-world). The NME and ECDF are given by:

$$NME_{A,f} = \frac{ME_{A,f}}{ME_{best,f} + 1}, \tag{4.10}$$

$$ECDF(x) = \frac{1}{n_A \times n_f} \sum_{i=1}^{n_A} \sum_{j=1}^{n_f} I\left(NME_{i,j} \le x\right), \tag{4.11}$$

where indices A and f denote the optimisation algorithm and the test function respectively, and n_A and n_f are the number of algorithms and test functions respectively.

4.5 Results and Discussion

Examples of methods used to optimise CEC-2013 problems include *Particle Swarm Optimization* [123], *Adaptive Differential Evolution* [127], [153], [154], *Mean Variance Mapping* [155] and *GA* [125]. The methods for optimisation of the traditional test functions, covered in this work, include *Evolutionary Programming* [8], *Particle Swarm Optimization* [156], *GA* [157], and *Hybrid Bee Colony/QEA* [158]. This section presents the obtained *bQIEA* and *rQIEA* results.

4.5.1 Functionality of the tested QIEA

First examined was the suitability of the four tested *QIEA* to be used as optimisation algorithms for real-value problems. In order to be useful, they must find solutions close to the optimum, as seen by reaching small error values. The first results examined are the performance on the traditional test functions, with minimum, mean and standard deviation data presented in Table 9, and Table 26 and Table 27 in Appendix A for 10 and 30 dimensions respectively.

These functions (listed in Appendix B) are reasonably smooth, at least locally, and therefore obtaining a good error score will require good exploitation abilities of the algorithm. Section 4.2 highlighted the difficulties for the *Classic* method in optimising the LSBs, and so it would be reasonable to expect that this would be reflected in poor minimum values as the exploitation would be hampered. Most solutions had errors of magnitude above 1e-01, the only exceptions being the 10 dimensional *Schwefel* 2.26, *Griewank* and *Penalised1* functions. Interpreting raw error values is difficult because it relates to the numerical properties of the fitness function. For example, the *Rosenbrock* has a constant factor of 100 (Appendix B), so the 10-dimension result of 7.39e02 is relatively not as bad as it would first appear. However, as the number of dimensions increases, the performance becomes obviously poorer, with four minima with magnitude of 1e07 at 50 dimensions. Means performance for all of the tested dimensions tested follows the same pattern in general, although there are some large discrepancies for the 10-dimension batch. These are a mean of 2.10e05 compared to a minimum of 7.39e02 for *Rosenbrock*, 2.73e04 versus 1.40e02 for *Quartic*, and 5.38e04 versus 1.99e00 for *Penalised2*. This implies, at least for the low dimensions, that running *Classic* several times is a necessity.

Table 9: Summary statistics for the 13 traditional test functions with 50 dimensions. Bold are best.

Traditional test functions 50 Dimensions	***bQIEA***						***rQIEA***					
	Classic			***HSB***			***RCQIEA***			***SRQEA***		
Function	**Min**	**Mean**	**Std dev**	**Min**	**Mean**	**Std dev**	**Min**	**Mean**	**Std dev**	**Min**	**Mean**	**Std dev**
01 Sphere	3.32E+03	5.54E+03	7.22E+02	7.89E+02	1.71E+03	3.85E+02	3.01E-04	5.81E-04	1.79E-04	**0.00E+00**	**0.00E+00**	**0.00E+00**
02 Schwefel 222	2.17E+02	3.08E+02	2.79E+01	8.59E+01	1.28E+02	1.57E+01	7.98E-02	1.08E-01	1.30E-02	**0.00E+00**	**0.00E+00**	**0.00E+00**
03 Schwefel 12	1.72E+06	2.89E+06	5.98E+05	2.77E+05	7.09E+05	1.86E+05	2.03E-01	4.26E-01	2.13E-01	**0.00E+00**	**0.00E+00**	**0.00E+00**
04 Schwefel 221	4.26E+01	4.80E+01	2.80E+00	3.05E+01	3.80E+01	2.41E+00	1.81E-01	3.05E-01	4.89E-02	**2.00E-02**	**3.29E-02**	**7.54E-03**
05 Rosenbrock	8.12E+07	3.42E+08	9.89E+07	9.02E+06	4.92E+07	2.19E+07	9.37E+00	1.27E+02	5.77E+01	**4.49E-02**	**4.34E+01**	**3.09E+01**
06 Step	3.59E+03	5.26E+03	8.57E+02	1.14E+03	1.77E+03	2.99E+02	**0.00E+00**	**0.00E+00**	**0.00E+00**	**0.00E+00**	**0.00E+00**	**0.00E+00**
07 Quartic	4.50E+07	7.79E+07	1.73E+07	1.59E+06	9.86E+06	5.05E+06	**0.00E+00**	**0.00E+00**	**0.00E+00**	**0.00E+00**	**0.00E+00**	**0.00E+00**
08 Schwefel 226	3.42E+02	5.80E+02	7.91E+01	8.64E+01	1.83E+02	4.35E+01	3.17E-05	6.60E-05	2.20E-05	**0.00E+00**	**0.00E+00**	**0.00E+00**
09 Basic Rastrigin	9.28E+01	1.33E+02	1.11E+01	6.74E+01	8.26E+01	6.64E+00	1.56E-04	2.89E-04	8.13E-05	**0.00E+00**	**0.00E+00**	**0.00E+00**
10 Basic Ackley	2.01E+01	2.01E+01	9.47E-03	1.89E+01	1.96E+01	2.09E-01	1.02E-02	1.66E-02	3.10E-03	**0.00E+00**	**5.63E-07**	**3.26E-06**
11 Basic Griewank	3.73E+01	5.03E+01	6.97E+00	8.13E+00	1.54E+01	2.89E+00	4.01E-04	**8.45E-03**	**9.66E-03**	**0.00E+00**	1.48E-02	2.59E-02
12 Penalised 1	1.13E+07	5.69E+07	3.00E+07	2.36E+04	2.60E+06	2.00E+06	**0.00E+00**	**0.00E+00**	**0.00E+00**	**0.00E+00**	**0.00E+00**	**0.00E+00**
13 Penalised 2	5.72E+07	1.56E+08	4.87E+07	3.80E+06	1.41E+07	6.33E+06	**0.00E+00**	**0.00E+00**	**0.00E+00**	**0.00E+00**	**0.00E+00**	**0.00E+00**

Table 10: Summary statistics for the 20 CEC-2013 test functions with 50 dimensions. Bold are best.

CEC-2013 test functions 50 Dimensions	***bQIEA***						***rQIEA***					
	Classic			***HSB***			***RCQIEA***			***SRQEA***		
Function	**Min**	**Mean**	**Std dev**	**Min**	**Mean**	**Std dev**	**Min**	**Mean**	**Std dev**	**Min**	**Mean**	**Std dev**
14 Sphere [duplicated]	3.32E+03	5.54E+03	7.22E+02	7.89E+02	1.71E+03	3.85E+02	3.01E-04	5.81E-04	1.79E-04	**0.00E+00**	**0.00E+00**	**0.00E+00**
15 R HC elliptic	6.52E+07	1.23E+08	2.30E+07	4.66E+07	7.09E+07	1.32E+07	5.84E+06	1.15E+07	3.01E+06	**1.55E+06**	**2.96E+06**	**6.43E+05**
16 Rotated bent cigar	4.50E+10	7.15E+10	1.08E+10	1.45E+10	2.46E+10	3.30E+09	1.18E+03	6.45E+06	2.26E+07	**4.98E-02**	**1.58E+05**	**1.08E+06**
17 Rotated discus	9.87E+04	1.59E+05	1.62E+04	**9.61E+04**	**1.28E+05**	**1.27E+04**	1.38E+05	1.92E+05	2.88E+04	1.14E+05	1.75E+05	2.47E+04
18 Different powers	1.20E+03	2.99E+03	8.85E+02	3.18E+02	6.81E+02	1.35E+02	1.57E-03	4.14E-03	1.94E-03	**0.00E+00**	**0.00E+00**	**0.00E+00**
19 Rotated Rosenbrock	2.81E+02	4.74E+02	6.76E+01	1.35E+02	2.41E+02	4.52E+01	2.98E+01	4.51E+01	**3.43E+00**	**2.38E+01**	**4.19E+01**	7.54E+00
20 Rotated Schaffers F7	1.71E+02	2.15E+02	1.70E+01	**1.28E+02**	**1.76E+02**	**1.41E+01**	1.79E+02	2.54E+02	1.22E+02	1.47E+02	2.46E+02	9.28E+01
21 Rotated Ackley	2.10E+01	2.12E+01	3.96E-02	2.10E+01	2.11E+01	3.88E-02	2.10E+01	2.11E+01	**3.74E-02**	**2.10E+01**	**2.11E+01**	4.68E-02
22 Rotated Weierstrass	4.75E+01	5.65E+01	2.47E+00	**4.72E+01**	**5.22E+01**	**2.35E+00**	5.71E+01	6.34E+01	3.36E+00	6.16E+01	6.74E+01	3.76E+00
23 Rotated Griewank	9.90E+02	1.43E+03	1.95E+02	4.96E+02	7.13E+02	1.01E+02	1.54E+00	2.25E+00	3.02E-01	**2.71E-02**	**1.31E-01**	**4.96E-02**
24 Rastrigin	1.86E+02	2.37E+02	2.04E+01	7.87E+01	1.04E+02	1.05E+01	5.83E-04	1.10E-03	3.38E-04	**0.00E+00**	**0.00E+00**	**0.00E+00**
25 Rotated Rastrigin	4.26E+02	5.34E+02	4.14E+01	**2.31E+02**	**4.00E+02**	**3.99E+01**	3.58E+02	6.08E+02	1.25E+02	4.60E+02	6.85E+02	1.38E+02
26 NC rotated Rastrigin	5.69E+02	6.63E+02	**3.95E+01**	**4.41E+02**	**5.54E+02**	4.39E+01	4.71E+02	6.30E+02	9.74E+01	5.01E+02	6.89E+02	1.01E+02
27 Schwefel 7	1.60E+03	2.29E+03	2.72E+02	3.88E+02	8.10E+02	1.46E+02	**2.96E-02**	**8.57E-02**	**2.41E-02**	9.99E-02	6.71E-01	2.94E-01
28 Rotated Schwefel 7	6.26E+03	7.88E+03	**4.70E+02**	4.63E+03	6.47E+03	5.08E+02	**4.37E+03**	6.25E+03	7.15E+02	4.69E+03	**6.22E+03**	6.23E+02
29 Rotated Katsuura	9.79E-01	1.67E+00	2.62E-01	**8.09E-01**	**1.39E+00**	**2.11E-01**	8.74E-01	1.64E+00	3.49E-01	8.93E-01	1.83E+00	4.41E-01
30 Lunacek bi-Rastrigin	3.78E+02	4.57E+02	3.80E+01	1.46E+02	1.95E+02	2.29E+01	3.82E-02	9.98E-02	3.51E-02	**0.00E+00**	**1.96E-04**	**1.40E-03**
31 R Lunacek bi-Rastrigin	8.64E+02	1.04E+03	8.09E+01	6.00E+02	7.50E+02	**5.49E+01**	**3.05E+02**	**4.80E+02**	7.87E+01	4.53E+02	6.12E+02	9.30E+01
32 RE Griewank Rosen.	4.75E+02	1.61E+03	6.75E+02	1.30E+02	2.75E+02	9.06E+01	**5.73E+01**	**1.45E+02**	**4.66E+01**	1.46E+02	2.91E+02	6.26E+01
33 RE Schaffers F6	1.77E+01	2.09E+01	1.21E+00	**1.58E+01**	**1.87E+01**	1.13E+00	2.05E+01	2.43E+01	**5.97E-01**	1.90E+01	2.44E+01	7.68E-01

For every test function in the traditional batch, *HSB* had better minimum values than *Classic*, apart from the 10-dimension *Penalised1* function. Although the magnitudes were generally similar, some were substantially better, e.g., 30-dimension *Rosenbrock*, *Penalised1* and *Penalised2*, and 50-dimension *Penalised*1. The consistently better performance suggests both that the LSB problems of *Classic* hampered its performance, and that the tested improvement of limiting the LSB probability saturation was successful.

Despite apparent functional performance by the *bQIEA*, the two *rQIEA* were substantially better. The worst performance for the *rQIEA* was for *RCQIEA* on the 30-dimension *Rosenbrock* with a minimum of 1.65e01, but most minima had magnitudes of less than 1e-01. *RCQIEA* found smaller than 1e-08 solutions (clamped to 0.00 in the results) for *Step, Quartic*, *Penalised1* and *Penalised2* in all tested dimensions. Despite *RCQIEA* performing well on these test functions, it was eclipsed by *SRQEA*. With the exception of *Schwefel* 2.21 and *Rosenbrock*, it obtained clamped 0.00 results for all of the functions, in all dimensions. Even for *Schwefel* 2.21 and *Rosenbrock* it had the best performance across the *QIEA* tested. The superior performance of the real algorithms over their binary counterparts is unsurprising, given the application to real-value problems, and the superior performance of *SRQEA* justifies the modification of the rotation gate function for these functions.

As CEC-2013 are a set of real-value problems, some being modified versions of the functions from the traditional set tested here, it was predicted that a similar pattern of results would be generated, with the *rQIEA* dominating the *bQIEA*. Although *HSB* outperformed *Classic*, and *SRQEA* outperformed *RCQIEA*, the performance of the *bQIEA* compared to the *rQIEA* was very different from its previous performance (Table 10 for 50 dimensions, and Table 28 and Table 29 in Appendix A for 10 and 30 dimensions respectively).

For several of the test functions - *Rotated Discus*, *Rotated Schaffers F7*, *Rotated Weierstrass*, *Rotated Rastrigin*, *Non-continuous Rotated Rastrigin*, *Rotated Katsuura*, *Rotated Expanded Grienwank, Rosenbrock and Rotated Expanded Schaffers F6*, one of the *bQIEA* had the best performance for one or more dimensions tested. When the *bQIEA* performed best, the *rQIEA* approached a similar order of magnitude, but when one of the *rQIEA* gave the best result, it sometimes considerably outperformed the

bQIEA (for example, on the *Rotated Bent Cigar*, *SRQEA* achieved 4.98e-02 compared to 4.50e10 and 1.45e10 for *Classic* and *HSB* respectively). Nevertheless, the positive results of the *bQIEA* are significant and surprising, given that they can outperform the *rQIEA* on some real-value benchmark functions.

The CEC-2013 functions are highly manipulated versions of traditional basic functions (many based on the traditional test functions used in this chapter). The manipulations include rotations, scalings and non-linear transforms. It may be that these transformations allow the *bQIEA* to perform well in one of two possible ways. Firstly, the transformations may increase the nonlinear interactions between dimensions, producing a fitness landscape that is very rough, and therefore more resembling a discrete space at scales above the very small. These search spaces may be suited to the binary methods presented here, possibly possessing similarities to the combinatorial problems that *bQIEA* have been successful with (e.g., *Knapsack* [135]). Alternatively, the search pattern may be the key. In the *rQIEA*, the search space is traversed using creep mutations with distances drawn from a normal distribution, while the movement in the *bQIEA* is performed using multi-scaled jumps as the bits flip between zero and one and move the search to an adjacent binary partition at the scale of the significance of the bit. This binary space partitioning could reflect, to some degree, the underlying structure of the search spaces.

For the *CEC-2013* set of test functions, the *bQIEA* achieved several minimum scores with a magnitude of 1e02 or less and, given that the test functions often contain large constants (1e06), it could be argued that they performed better on the more difficult test functions than on the traditional set of functions. It would be interesting to see if this scales, so that the *bQIEA* have increasingly better relative performance as the fitness landscape becomes more complex.

Although *SRQEA* was the best performer, in terms of number of best minimum values found and the ability to find threshold zero error values for some functions (which none of the other algorithms managed to do), when looking at the general performance across all of the functions and algorithms, the picture was somewhat more mixed. A heat map of best minimum values, scaled relatively from the best performing algorithm to the worst on each test function, is presented in Figure 13, with a green (zero) rectangle indicating best performance, and a light-green (one) rectangle indicating worst

performance. In this plot, judging by the number of darker rectangles, *RCQIEA* performs well, arguably outperforming *SRQEA*. From the raw data in Table 10, it can be seen that when the performances of the *rQIEA* are close, *SRQEA* produces better results than *RCQIEA*, but this is not generally noticeable in the heat map, where the larger degrees of magnitude produced by the *bQIEA* obscure the *rQIEA* differences. Summarising the raw data and the heat map, it can be said that *RCQIEA* had a slightly better average performance but *SRQEA* was able to produce much better individual scores for some functions. The more random nature of the rotation gate of *RCQIEA* may produce desirable search characteristics for the *CEC-2013* test functions, at the expense of more exploitation.

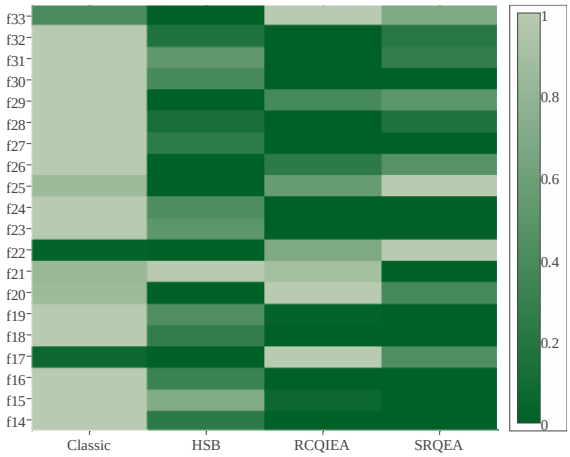


Figure 13: Heat map of best convergence to a minimum by the QIEA on the CEC-2013 test functions.

For the *CEC-2011* real-world problems, converging to the minima was best for the *rQIEA* (Table 11) - *RCQIEA* generated the best minimum for the *Radar Polly Phase* problem, while *SRQEA* had the best minimum results for the other three functions. However, for the mean values, both *Classic* and *HSB* outperformed the *rQIEA* for the *Frequency Modulation* and *Radar Polly Phase* problems. The nested functions present in both of these benchmarks suggest a highly nonlinear search space, so these results are consistent with the findings and interpretations of the performance of the *bQIEA* on the *CEC-2013* functions.

Finally, a summary of algorithms' mean performance across multiple test functions in Figure 14. The plots show a cumulative normalised count, for each algorithm, of how many functions possess a normalised mean performance below the horizontal

coordinate value. The sooner the plot reaches 1.0 in the vertical axis, the better the algorithm performs (as this indicates a high probability of achieving low mean error values).

The best performance on the traditional test functions (Figure 14a) is dominated by the two *rQIEA* methods, which can also be seen for all of the test functions taken together (Figure 14d), with *Classic* performing poorly for both of those cases. For the *CEC-2013* functions *HSB* is much closer (Figure 14b), catching up sooner with the *rQIEA* in the plot, although it starts with poorer results, indicating a low probability of producing very low mean scores across the function set. The performance of *RCQIEA* compared to *SRQEA* for CEC-2013 is in line with the results presented in the heat map (Figure 13). *SRQEA* outperforms *RCQIEA* for low mean values, but takes a slight lead for normalised means between 0.2 and 0.4. For the real-world test functions (Figure 14c), the situation is completely reversed, with *Classic* performing the best, followed by *HSB*.

In summarising the ECDF and the results given in the tables, it can be concluded that, although the *rQIEA* have superior best performance (minimum values found), the *bQIEA* algorithms do have good mean performance, often superior to their real-value counterparts. Again, it is with the more complicated *CEC-2013* and real-world *CEC-2011* functions that the *bQIEA* perform at their best, often outperforming the *rQIEA*.

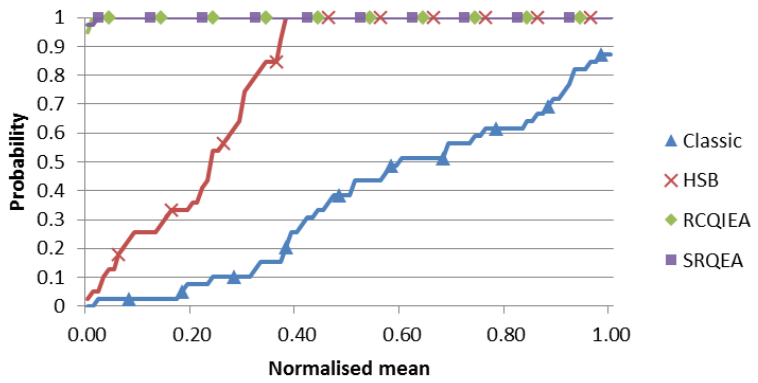
Table 11: Summary statistics for CEC-2011 real world problems. Bold are best.

	bQIEA						***rQIEA***					
	Classic			***HSB***			***RCQIEA***			***SRQEA***		
Function	**Min**	**Mean**	**Std dev**	**Min**	**Mean**	**Std dev**	**Min**	**Mean**	**Std dev**	**Min**	**Mean**	**Std dev**
Frequency Modulation	1.79E+00	1.29E+01	**3.98E+00**	7.40E-02	**7.22E+00**	4.73E+00	2.71E-04	1.57E+01	5.78E+00	**0.00E+00**	1.70E+01	4.68E+00
Lennard Jones 5 atoms	-1.07E+01	-9.23E+00	6.66E-01	-1.21E+01	-1.05E+01	**5.97E-01**	-1.27E+01	-1.18E+01	1.03E+00	**-1.27E+01**	**-1.21E+01**	1.02E+00
Lennard Jones 10 atoms	-1.91E+01	-1.50E+01	**1.34E+00**	-2.27E+01	-1.77E+01	1.50E+00	-3.08E+01	-2.26E+01	3.87E+00	**-3.18E+01**	**-2.41E+01**	4.23E+00
Radar Polly Phase	1.60E+00	1.98E+00	1.25E-01	1.65E+00	**1.85E+00**	**8.72E-02**	**1.50E+00**	2.00E+00	2.31E-01	1.59E+00	2.11E+00	2.10E-01

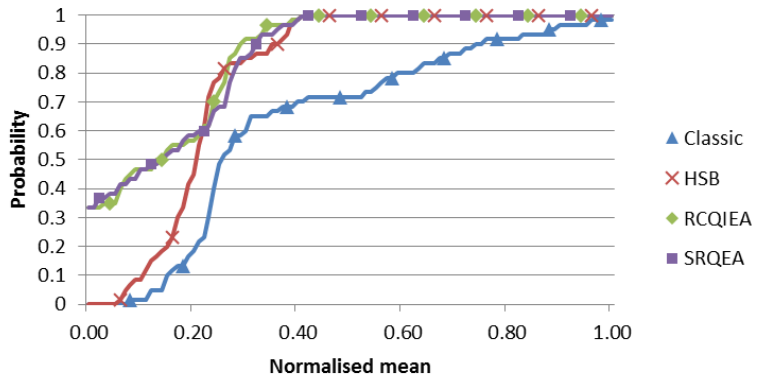
4.5.2 Evolution properties of the QIEA

Mean error values per generation (averaged across the 51 runs) are shown for four functions in Figure 15. For most functions, *Classic* outperformed *HSB* early on the evolution, but tends to stall earlier and is generally overtaken by *HSB* at around the 30% (of the total number of generations) time point (for example, see the *Sphere* and *Rotated Rosenbrock* function timelines in Figure 15a,b). This gives additional support to the argument that *Classic* was prematurely converging when applied to real-value problems, and justifies the approach when formulating the *HSB* adaptation. It should also be noted though that *HSB* also usually approaches an approximately zero gradient relatively early on (50% of time or less), implying there is further need to improve premature convergence.

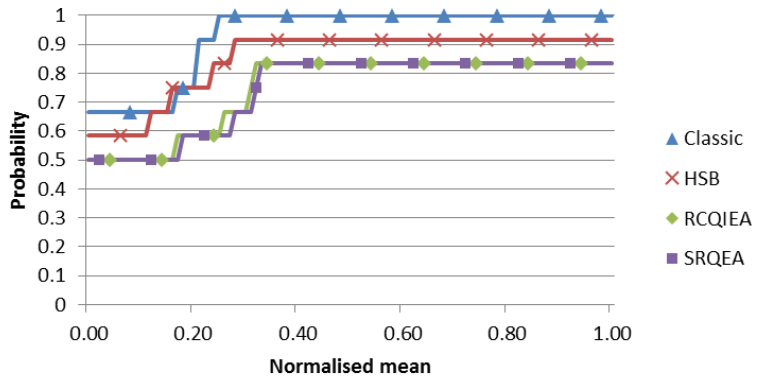
a)



b)



c)



d)

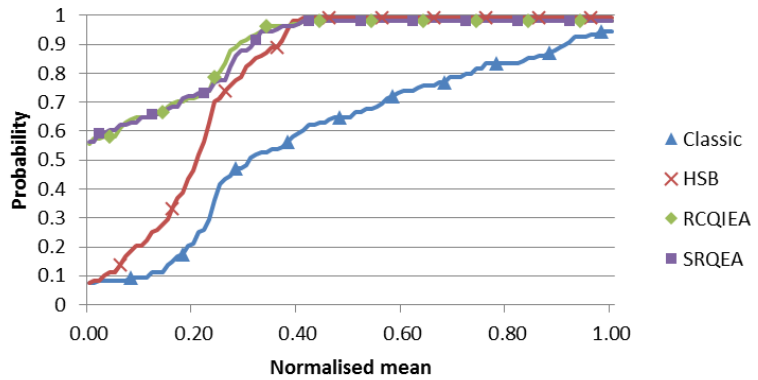
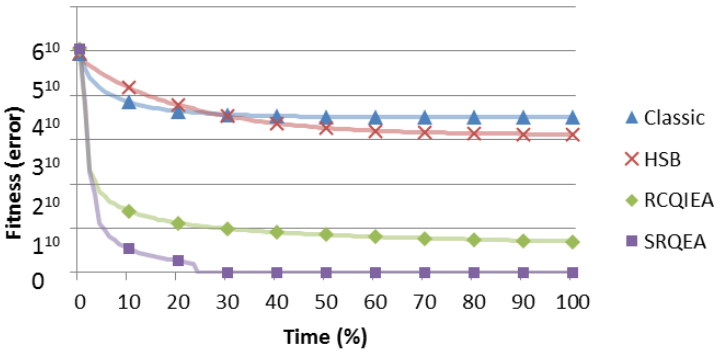
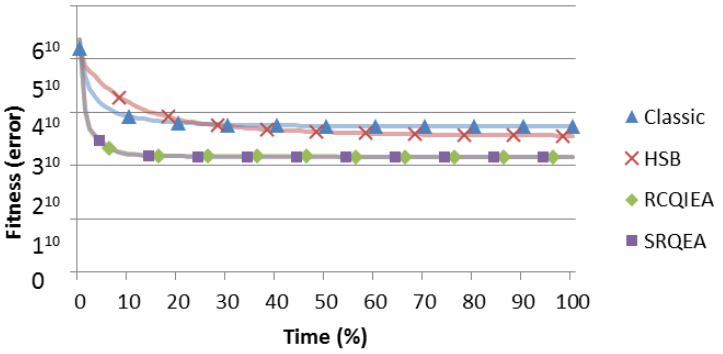


Figure 14: Empirical cumulative probability distribution function of mean errors across a) traditional, b) CEC-2013, c) real-world and d) all test functions, comparing the four QIEA.

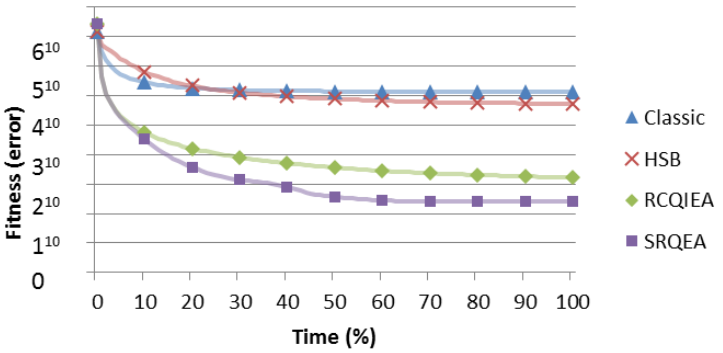
a)



b)



c)



d)

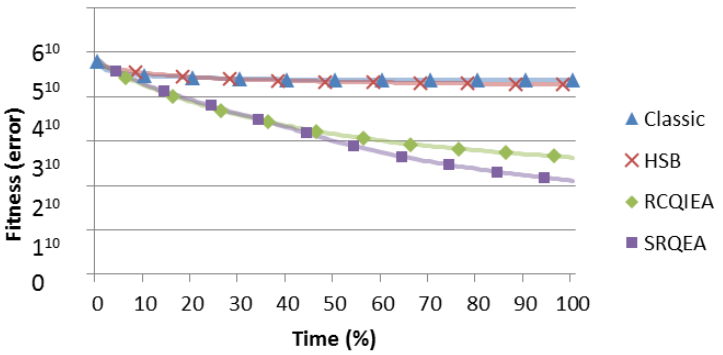


Figure 15: Timeline evolution of mean error values for a) Sphere, b) Rotated Rosenbrock, c) Rotated Griewank and d) Schwefel 2.21.

For the majority of cases where *SRQEA* outperformed *RCQIEA*, their early performances were very similar, but *SRQEA* would establish a lead from typically the 10-30% time mark (*Rotated Griewank* in Figure 15c). This can be interpreted as indicating that the corrected rotation formula allowed a more refined search in later stages. Both *rQIEA* demonstrated a clear non-zero gradient at the end of the timeline in several of the plots (such as Figure 15c,d). This suggests they are capable of finding significantly better results if the algorithm is run for longer. As the plots display the fitness to the 10^{th} root, this is relevant for fine convergence to the optimal value, indicating room for improvement of precision.

4.5.3 Comparison of QIEA with published results

As the best performing *QIEA* on the traditional test functions, *SRQEA* was chosen to compare with two other algorithms – *FEP* [8] and *MADE* [151] (Table 12). Comparison is made difficult by varying numbers of function evaluations across the published methods, but in general, *SRQEA* outperformed *FEP* except for the *Rosenbrock*, *Ackley* and *Griewank* functions where *FEP* had a superior mean and standard deviation. *MADE* was better than *SRQEA* for *Schwefel 2.21*, *Rosenbrock*, *Ackley* and *Griewank*, but *SRQEA* beat *MADE* for *Quartic* and matched it for all of the other functions. Unfortunately, best minimum values found were not published for either algorithm, but since *MADE* produced several zero means, it is clear those results would have been good as well.

The exploitation ability of *RCQIEA* and *SRQEA* was compared to data published on a set of differential algorithms (*DE*) [151] and is presented in Table 13, using the success rate (*SR*) and success performance (*SP*) metrics, using a success threshold of 1E-08, except for 1E-02 for Quartic in order to compare to published data. In general, the *DE* algorithms achieved success more often, and quicker, than the *rQIEA*. The *SRQEA* is overall superior to *RCQIEA* for these metrics (with a better *SR*), but when *RCQIEA* was successful it tended to achieve success more quickly than *SRQEA* (better *SP*). These results represent the weakest performance for the *QIEA*, and indicate room for improvement in their search and exploitation abilities for the traditional test functions, although success rates were based on very low thresholds (usually 1e-08) and therefore may not be important in practical cases. Unfortunately *MADE* was not applied to the

CEC-2013 functions, so it cannot be said if these conclusions hold for the more complicated test functions.

Table 14, and Table 30 and Table 31 in Appendix A, show the performance of *SRQEA* against two algorithms that were applied to the *CEC-2013* fitness functions [9]. The two algorithms compared are a particle swarm optimization algorithm *SPSO-2011* [123] and a genetic algorithm *GA* [125]. *SRQEA* was chosen for comparison as, overall, it was the best performing *QIEA* tested here, in terms of minimum values found.

Looking at all dimensions, all three algorithms achieved some best performances. *SPSO-2011* performed least well, having fewer best minimum results, and most of those being joint equal with one or both of the other algorithms. The main competition for *SRQEA* came from the *GA*. For 10 dimensions it achieved 16 best performances, with *SRQEA* only achieving seven. For 30 dimensions *GA* scored 12 best performances, while the *SRQEA* reached 8, but for 50 dimensions, *SRQEA* took the lead with 11 compared to 9 best results for the *GA*. This demonstrates better scaling with increased number of dimensions for *SRQEA* than for the *GA*. Mean performance was similarly distributed across all dimensions but *SRQEA* showed improved standard deviation performance again for 50 dimensions, outperforming the other algorithms substantially. This shows a more consistent relative performance at higher dimensions for *SRQEA* as well as better minima and means.

Table 12: Comparison between SRQEA, Fast Evolutionary Programming (FEP) [8], and MADE [151] on the traditional test functions. Bold are best.

30 Dimensions	***SRQEA***				***FEP***			***MADE***		
Function	**F Evals**	**Min**	**Mean**	**Std dev**	**F Evals**	**Mean**	**Std dev**	**F Evals**	**Mean**	**Std dev**
1 Sphere	300000	0.00E+00	**0.00E+00**	**0.00E+00**	150000	8.10E-03	7.70E-04	150000	**0.00E+00**	**0.00E+00**
2 Schwefel 222	300000	0.00E+00	**0.00E+00**	**0.00E+00**	200000	8.10E-03	7.70E-04	150000	**0.00E+00**	**0.00E+00**
3 Schwefel 12	300000	0.00E+00	**0.00E+00**	**0.00E+00**	500000	1.60E-02	1.40E-02	200000	**0.00E+00**	**0.00E+00**
4 Schwefel 221	300000	3.51E-03	6.16E-03	1.56E-03	500000	3.00E-01	5.00E-01	500000	**0.00E+00**	**0.00E+00**
5 Rosenbrock	300000	1.04E-02	8.86E+01	1.80E+02	2000000	5.06E+00	5.87E+00	500000	**3.97E-01**	**1.63E+00**
6 Step	300000	0.00E+00	**0.00E+00**	**0.00E+00**	150000	**0.00E+00**	**0.00E+00**	500000	**0.00E+00**	**0.00E+00**
7 Quartic	300000	0.00E+00	**0.00E+00**	**0.00E+00**	300000	7.60E-03	2.60E-03	300000	1.24E-03	3.78E-04
8 Schwefel 226	300000	0.00E+00	**0.00E+00**	**0.00E+00**	900000	1.50E+01	5.26E+01	200000	**0.00E+00**	**0.00E+00**
9 Basic Rastrigin	300000	0.00E+00	**0.00E+00**	**0.00E+00**	500000	4.60E-02	1.20E-02	300000	**0.00E+00**	**0.00E+00**
10 Basic Ackley	300000	0.00E+00	9.20E-01	4.01E+00	150000	1.80E-02	2.10E-03	150000	**0.00E+00**	**0.00E+00**
11 Basic Griewank	300000	0.00E+00	2.06E-02	2.25E-02	200000	1.60E-02	2.20E-02	200000	**0.00E+00**	**0.00E+00**
12 Penalised 1	300000	0.00E+00	**0.00E+00**	**0.00E+00**	150000	9.20E-06	3.60E-06	300000	**0.00E+00**	**0.00E+00**
13 Penalised 2	300000	0.00E+00	**0.00E+00**	**0.00E+00**	150000	1.60E-04	7.30E-05	300000	**0.00E+00**	**0.00E+00**

Table 13: Comparison of success rates (SR) and speed of convergence (SP), between RCQIEA, SRQEA and 4 differential evolution algorithms, for the 13 traditional test functions with 30 dimensions. Bold are best.

Function	***RCQIEA***		***SRQEA***		***jDE***		***SDE***		***JADE***		***MADE***	
	SP	***SR***	***SP***	***SR***	***SP***	***SR***	***SP***	***SR***	***SP***	***SR***	***SP***	***SR***
1 Sphere	—	0	2.48E+05	**1**	5.93E+04	**1**	3.91E+04	**1**	3.04E+04	**1**	**2.29E+04**	**1**
2 Schwefel 222	—	0	7.19E+05	**1**	8.16E+04	**1**	5.31E+04	**1**	5.61E+04	**1**	**3.64E+04**	**1**
3 Schwefel 12	—	0	3.56E+05	**1**	3.37E+05	**1**	—	0	**7.17E+04**	**1**	1.34E+05	**1**
4 Schwefel 221	—	0	—	0	2.99E+05	**1**	4.72E+05	0.44	—	0	**1.27E+05**	**1**
5 Rosenbrock	—	0	—	0	5.89E+06	0.08	—	0	**1.22E+05**	**0.92**	1.97E+05	**0.92**
6 Step	7.77E+04	**1**	1.20E+05	**1**	2.27E+04	**1**	1.44E+04	**1**	1.16E+04	**1**	**7.89E+03**	**1**
7 Quartic	1.37E+05	**1**	1.80E+05	**1**	1.12E+05	**1**	8.34E+04	**1**	2.97E+04	**1**	**2.83E+04**	**1**
8 Schwefel 226	—	0	2.12E+05	**1**	7.85E+04	**1**	**5.50E+04**	**1**	1.00E+05	**1**	6.00E+04	**1**
9 Basic Rastrigin	—	0	2.53E+05	**1**	1.17E+05	**1**	6.14E+05	0.36	1.31E+05	**1**	**1.14E+05**	**1**
10 Basic Ackley	—	0	1.54E+06	0.63	9.02E+04	**1**	5.95E+04	**1**	4.75E+04	**1**	**3.55E+04**	**1**
11 Basic Griewank	—	0	8.50E+05	0.31	6.21E+04	**1**	4.07E+04	**1**	3.30E+04	**1**	**2.41E+04**	**1**
12 Penalised 1	5.61E+04	**1**	9.46E+04	**1**	5.40E+04	**1**	3.66E+04	**1**	2.95E+04	**1**	**2.03E+04**	**1**
13 Penalised 2	3.85E+04	**1**	7.15E+04	**1**	5.76E+04	**1**	3.77E+04	**1**	2.95E+04	**1**	**2.19E+04**	**1**

Table 14: SRQEA compared to SPSO-2011 and a GA algorithm for the CEC-2013 functions with 50 dimensions. Bold are best.

50 Dimensions	***SRQEA***			***SPSO-2011*** [123]			***GA*** [125]			
Function	**Min**	**Mean**	**Std dev**	**Min**	**Median**	**Std dev**	**Min**	**Median**	**Mean**	**Std dev**
14 Sphere [duplicated]	**0.00E+00**	**0.00E+00**	**0.00E+00**	**0.00E+00**	**0.00E+00**	3.18E-13	**0.00E+00**	**0.00E+00**	**0.00E+00**	**0.00E+00**
15 R HC elliptic	1.55E+06	2.96E+06	6.43E+05	3.79E+05	6.80E+05	**1.87E+05**	**1.74E+05**	**4.28E+05**	**4.76E+05**	2.14E+05
16 Rotated bent cigar	**4.98E-02**	**1.58E+05**	**1.08E+06**	2.00E+07	4.37E+08	9.47E+08	2.55E+06	**3.44E+07**	1.06E+08	1.49E+08
17 Rotated discus	1.14E+05	1.75E+05	2.47E+04	3.22E+04	5.10E+04	8.72E+03	**4.90E-01**	**2.25E+00**	**3.33E+00**	**4.88E+00**
18 Different powers	**0.00E+00**	**0.00E+00**	**0.00E+00**	**0.00E+00**	**0.00E+00**	5.41E-05	**0.00E+00**	**0.00E+00**	4.77E+04	1.70E+05
19 Rotated Rosenbrock	2.38E+01	**4.19E+01**	**7.54E+00**	**1.84E+01**	**4.35E+01**	2.41E+01	3.66E+01	4.36E+01	4.72E+01	1.40E+01
20 Rotated Schaffers F7	1.47E+02	2.46E+02	9.28E+01	5.61E+01	8.64E+01	**1.53E+01**	**1.51E+01**	**3.97E+01**	**4.17E+01**	1.83E+01
21 Rotated Ackley	**2.10E+01**	**2.11E+01**	4.68E-02	2.10E+01	**2.11E+01**	4.25E-02	2.11E+01	2.12E+01	2.12E+01	**3.98E-02**
22 Rotated Weierstrass	6.16E+01	**6.74E+01**	**3.76E+00**	**4.52E+01**	**5.40E+01**	6.74E+00	5.21E+01	7.53E+01	7.43E+01	3.97E+00
23 Rotated Griewank	**2.71E-02**	1.31E-01	**4.96E-02**	1.00E-01	4.00E-01	2.38E-01	2.71E-02	**9.36E-02**	**1.05E-01**	7.09E-02
24 Rastrigin	**0.00E+00**	**0.00E+00**	**0.00E+00**	1.50E+02	2.30E+02	4.18E+01	1.49E+01	**5.37E+01**	5.57E+01	2.23E+01
25 Rotated Rastrigin	4.60E+02	6.85E+02	1.38E+02	1.62E+02	2.35E+02	4.87E+01	**5.07E+01**	**9.75E+01**	**9.83E+01**	**2.45E+01**
26 NC rotated Rastrigin	5.01E+02	6.89E+02	1.01E+02	3.20E+02	4.28E+02	6.22E+01	**1.04E+02**	**1.86E+02**	**1.93E+02**	**5.30E+01**
27 Schwefel 7	**9.99E-02**	**6.71E-01**	**2.94E-01**	5.51E+03	7.26E+03	8.53E+02	1.06E+03	**2.30E+03**	2.55E+03	1.14E+03
28 Rotated Schwefel 7	**4.69E+03**	**6.22E+03**	**6.23E+02**	5.68E+03	**7.92E+03**	1.14E+03	6.20E+03	8.24E+03	9.84E+03	3.19E+03
29 Rotated Katsuura	**8.93E-01**	**1.83E+00**	4.41E-01	1.40E+00	**2.00E+00**	**3.87E-01**	2.23E+00	3.76E+00	3.68E+00	3.88E-01
30 Lunacek bi-Rastrigin	**0.00E+00**	**1.96E-04**	**1.40E-03**	2.08E+02	3.11E+02	6.62E+01	8.25E+01	**1.13E+02**	1.15E+02	2.00E+01
31 R Lunacek bi-Rastrigin	4.53E+02	6.12E+02	9.30E+01	1.70E+02	2.91E+02	**6.24E+01**	**8.83E+01**	**1.32E+02**	**1.68E+02**	1.02E+02
32 RE Griewank Rosen.	1.46E+02	2.91E+02	6.26E+01	1.70E+01	3.72E+01	1.20E+01	**3.60E+00**	**9.02E+00**	**8.92E+00**	**3.17E+00**
33 RE Schaffers F6	**1.90E+01**	2.44E+01	**7.68E-01**	1.99E+01	**2.27E+01**	1.19E+00	1.99E+01	2.36E+01	**2.35E+01**	8.02E-01

Table 15: Comparison of performance on real-world problems between SRQEA and three differential evolutionary algorithms. The starred value has been clamped to zero as it was below the threshold of 1E-08 (used in the QIEA simulations). Bold are best.

	SRQEA			**MADE-WS**			**EA-DE-Memetic**			**Adaptive DE**		
Function	**Min**	**Mean**	**Std dev**	**Min**	**Mean**	**Std dev**	**Min**	**Mean**	**Std dev**	**Min**	**Mean**	**Std Dev**
FM Wave match	**0.00E+00**	1.70E+01	4.68E+00	**-**	**8.81E-01**	**2.47E+00**	**0.00E+00***	3.81E+00	5.21E+00	**0.00E+00**	4.85E+00	6.69E+00
Lennard-Jones 5 atoms	**-1.27E+01**	**-1.21E+01**	1.02E+00	**-**	-9.09E+00	**8.83E-02**	-	-	-	-	-	-
Lennard-Jones 10 atoms	**-3.18E+01**	-2.41E+01	4.23E+00	**-**	-2.66E+01	**8.64E-01**	-2.84E+01	-2.59E+01	2.24E+00	-2.80E+01	**-2.68E+01**	2.11E+00
Radar Polly Phase	**1.59E+00**	**2.10E+00**	2.09E-01	**-**	-	-	2.20E+02	2.20E+02	**0.00E+00**	2.20E+02	2.20E+02	**0.00E+00**

The poorer performance of *SPSO-2011* (Table 14) and the better performance of the *GA* may suggest that the recombinatorial properties of the cross-over operator may aid the search pattern for the *CEC-2013* functions. This is consistent with both of the presented hypotheses for why the *bQIEA* performed relatively well against the *rQIEA* – either treating the rougher space as more discrete and looking for recombination, or navigating through hops (swapping genes in the case of *GA*, and flipping bits in the case of the *bQIEA*). Although overall *SRQEA* was better, it would be interesting to see how *bQIEA* perform against *rQIEA* and other algorithms on even more complex search spaces.

A heat map of the relative performance by the three compared algorithms on 50-dimensional *CEC-2013* test functions is shown in Figure 16. *SRQEA* has the highest number of best performances, but *GA* has fewer worst performances. Again, this indicates better exploitation properties for *SRQEA* at the expense of exploration.

Figure 16: Heat map of best minimum performance by SRQEA compared to published PSO and GA algorithms on the CEC-2013 test functions.

A comparison between *SRQEA* and two alternative algorithms, when applied to the real-world problems is shown in Table 15. For the *frequency modulation wave matching problem*, *MADE-WS* [151] had the best mean and standard deviation. Unfortunately, the authors did not report a minimum value. *SRQEA* outperformed the hybrid algorithm [159] and *the DE* algorithm [160], in terms of mean and standard deviation, while equalling the best minimum performance. The mean and standard deviation were worse but comparable with the *MADE-WS* results.

For the *Lennard-Jones* problems, *SRQEA* again established the best minimum values, but *MADE-WS* did not have a comparable values published. *SRQEA* did have the best mean value for *Lennard-Jones5* but only outperformed the hybrid algorithm for *Lennard-Jones10*.

For the *radar waveform parameter specification problem*, *SRQEA* was the clear winner. The published results [159] and [160] both gave a suspiciously poor value though, and it may be worth considering whether there were issues in using shared code for the function evaluations. The problem was directly tackled in [161] where a *variable neighbourhood search algorithm* gave a minimum value of 8.58e-01 which was better than that achieved by the *SRQEA*.

4.6 Conclusion

When applied to real-value optimization tasks, all of the *QIEA* tested and validated in this investigation produced usually excellent results for basic test functions, and acceptable to excellent results for the more complicated benchmarks. Binary *QIEA* are a direct implementation of the quantum computing metaphor, which is built around repeated sampling of binary strings, analogous to superposition of states on a set of quantum bits. The Qbit probabilities define a probability distribution that elegantly specifies both the region of the best solution found so far, and the variance of the search radius. As the probabilities saturate, the mean position of the search becomes clearly defined, and the variance of the search narrows. Although the original *Classic* algorithm performed relatively poorly on the optimization tasks examined here, the modified *HSB* did substantially improve the results. In many instances it outperformed *RCQIEA*, especially for the more difficult *CEC-2013* test functions. The timeline plots highlighted the premature convergence of *Classic* (Figure 15a), giving further justification for the choice of modification, which was developed in response to the analysis of individual bit evolution. By explicitly limiting the saturation of less significant bits to the magnitude of saturation of more significant bits, *HSB* avoids the problems that *Classic* encountered for real-value problems, although horizontal lines in the latter half of some timeline plots suggest there is still room for improvement. The population size results (Figure 12a and Figure 12b) also suggest exploration issues, as

the *bQIEA* benefit from a larger population size for a fixed number of function evaluations.

The best results came from the *rQIEA*, especially from the modified version - *SRQEA* (Table 9, Table 10 and Table 11) but the *rQIEA* specifications require a compromise with respect to the quantum metaphor. In most *rQIEA*, and certainly the ones presented here, the Qbits and the quantum rotation gate give a mechanism for adjusting the radius of search throughout the evolution, such as through the creep mutation operators of the rQIEA presented here, or in the velocity equations in PSO algorithms [147]. Although that is not a problem of itself, it may be useful to view these quantum inspired algorithms as operator algorithms used within other optimization methods. Presented on their own, *rQIEA* can largely resemble other techniques. For example, the *RCQIEA* algorithm used here looks similar to simulated annealing, with the rotation gate adjusting the variance for neighbour selection.

The modification to the rotation gate produced superior results, particularly with regards to the exploitation (Table 10), although as it can be seen from the heatmaps of Figure 16 and Figure 13, the average performance across the functions is slightly compromised. This suggests the superior exploitation may come at the expense of some exploration limitation. As well as being beneficial in this specific implementation, it would be interesting for future work to explore the possibility of using the modified rotation in other algorithms, as a way of adjusting search variance.

When compared to other published results, the modified algorithms were superior for the more complex *CEC-2013* functions (Table 14). For the traditional test functions, they were generally outperformed by other published results (in particular, the *DE* algorithms [151], Table 13). However, timeline plots (Figure 15) suggest the *rQIEA* may continue to improve if left for longer. It would therefore be interesting to see if these algorithms are suitable for increasingly complicated test functions, where longer processing times are to be expected.

Surprisingly, the *bQIEA* appeared to perform better for the more complex *CEC-2013* and the real-world test functions (Table 10 and Table 11). It can be speculated that this may be because either the transferred search space begins to resemble the binary space portioning that the *bQIEA* generate, or that the search hops at different scales (depending on bit significance) may result in more suitable search patterns when

compared to *rQIEA* or other algorithms. The ability of *bQIEA* to combine different scales, through bit manipulation, may explain their improved performance on these more sophisticated tasks. As more complex fitness functions are published in the future, it would be worth including *bQIEA* (and perhaps other binary optimisation algorithms) in attempting to optimise them.

QIEA provide a good starting point for optimization. Deficiencies, when compared to competing algorithms, were largely down to fine exploitation, with results being of a similar degree of magnitude in error (Table 12). Future work would be beneficial on improving exploration for *SRQEA*, or further reducing the premature convergence for *HSB*. This may be achieved through an analysis of the effect of changing algorithm parameters (as discussed below), or by including the *QIEA* in hybrid algorithms with a two-stage exploration and exploitation process. Using the configuration of step size and other parameters presented here, the two *rQIEA* are more orientated towards exploration than exploitation. This is demonstrated by the populations analysis (Figure 12), which showed they both benefitted from a small population size for a given number of function evaluations (thereby increasing the number of iterations per individual). The *bQIEA* in contrast performed best with a larger population size and so appear to be balanced more towards exploitation than exploration, thus needing larger population size to effectively explore the search space.

One final advantage of *QIEA* is the low number of parameters they require for the main part of their implementation. Generally, only the number of individuals and step size for the rotation gate are needed. The *rQIEA* presented here also include a parameter for the number of children produced in each generation. For all of the investigated algorithms, the number of individuals and rotation gate step magnitude need specifying. The *bQIEA* also have parameters for local and global update rates, while *rQIEA* have crossover rates. How these affect the overall performance was not evaluated. The *rQIEA* also add a parameter for the number of offspring spawned at each iteration. Again, changing this was not analysed and further investigation into the optimisation of these parameters would be worth conducting.

5 Comparative experiments in evolution of locomotion

In this chapter, a series of experiments is presented, examining the evolution of vertical jump, long jump and walking gaits. All of the skills were developed for biped models, and walking was developed for a quadruped model. Different models, platforms, control algorithms and optimisation algorithms were compared and the results are presented, along with problems encountered in simulation, with conclusions and directions for future work concluding the chapter.

5.1 Vertical jump skill

The first experiments presented here sought to produce vertical jumps in a biped model, optimising maximum height achieved. A counter-movement element in a vertical jump has been identified as important in human jumping [162]. One reason for this is called a stretch shortening cycle, which increases performance due to storage of elastic energy and increased muscle activation [163]. These advantageous properties will be missing in a physical simulation consisting of rigid bodies, which is the case for the systems used here as soft body dynamics are more complicated to simulate. However, the increased acceleration phase made possible by bending the knees before jumping still favours a counter-movement jump for maximum height, and so an additional criterion will be used in evaluation - optimisation of a vertical jump skills will be regarded as successful if a counter-movement is observed.

5.1.1 Vertical jump in detailed biped model

The vertical jump was first used to test the process of optimisating movement and the first experiment is described briefly here, in a qualitative way as no comparative

experiments were performed at this stage. However, the results demonstrate a proof of concept, and a description is given of issues encountered. *Newton Dynamics* was selected for speed, but since a vertical jump is a discrete skill (short duration, not continuous) it was decided to use the detailed biped model as optimisation was assumed to be less taxing in this scenario. Visualisation was provided by *Ogre3D* and software was written in C#.

A Van der Pol based CPG was used for control and a genetic algorithm was used for optimisation, these being described in detail in sections 2.1.2 and 3.2.1 respectively. The fitness function was simply maximum height obtained in a fixed time period, although it was later modified to only accept one jump phase, as some evolutions produced higher jumps using multiple jump phases. Evolution was often successful, as per counter-movement criterion, and an example animation is presented in Figure 17.

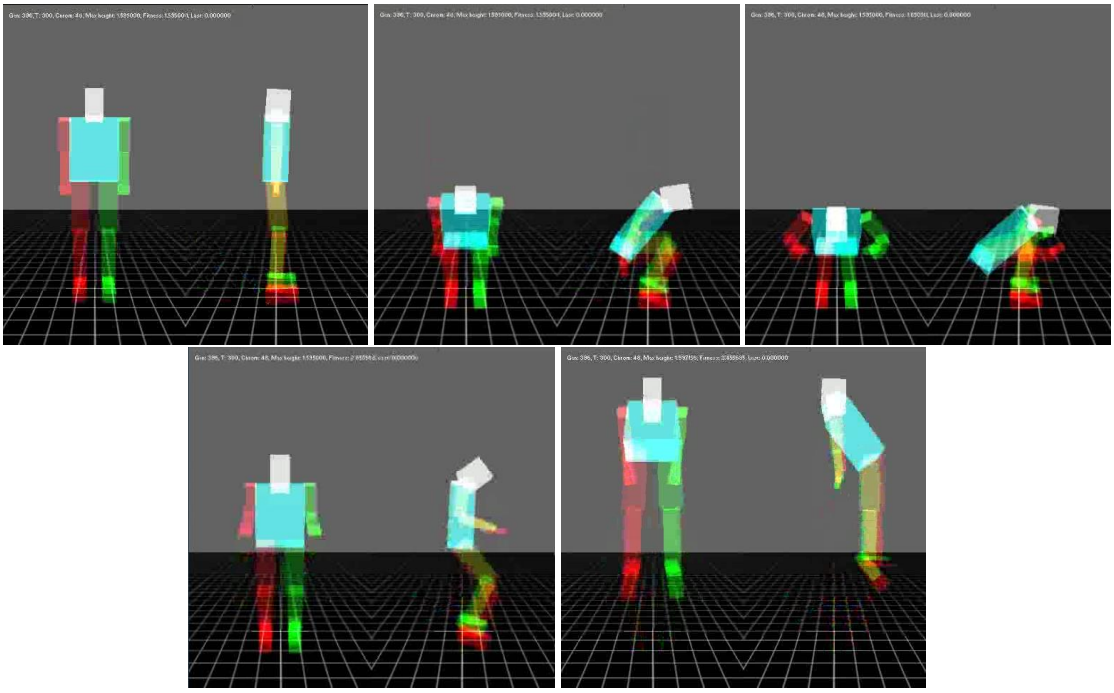


Figure 17: Still frames from an animation of a vertical jump evolved for a biped model, in Newton Dynamics Engine, from left to right, top to bottom.

Care had to be taken in specifying the physical model. As noted in section 3.4.3 it was constructed using cuboids and hinge joints, for simplicity. However, results were sensitive to the control properties, especially maximum joint speeds. Unrealistic, explosive movements were produced when maximum speeds were allowed to be too high. Although this is somewhat obvious, it does highlight the need for proper

calibration even in theoretical experiments (not with a real-world target system in mind).

5.1.2 Vertical jump in detailed robot biped model

Following the proof of concept work, a quantitative comparative study was conducted into evolving and optimising vertical jumps. Given the success from the previous section, it was felt that the slower OpenHRP3 software could be used, in order to use a platform that was common in the literature, and meeting the thesis objective of testing methods on multiple physics platforms.

The control algorithms outlined below (a CPG and a RNN) output desired joint angles at each time iteration. They are then converted into joint torque values, which in turn are fed into the physical simulation. This was done using a proportional–integral–derivative controller (PID controller) [164]. A PID controller outputs control values based on the error between the current measured position and desired position (expressed for this experiment in terms of the joint angles). It is given in discrete time form by equation (5.1):

$$u(t_k) = K_p e(t_k) + K_i \sum_{i=1}^{k} e(t_i)\Delta t + K_d \left(\frac{e(t_k) - e(t_{k-1})}{\Delta t} \right) \quad (5.1)$$

where t_k is the time at iteration k, $u(t_k)$ the output torque for joint k, $e(t_k)$ the error (difference between target joint angle and measured joint angle), and K_p, K_i, and K_d are tuning parameters representing proportional, integral and derivative gains respectively. Additionally, in order to cope with situations where the control algorithms changed their output too rapidly the output joint torque was limited to a maximum value.

As well as developing a vertical jump in a more complicated model, two methods from the literature were compared – a *Van der Pol* oscillator based CPG and a fully connected leaky integrator recurrent neural network.

5.1.3 CPG – Van der Pol

For the central pattern generator controller, a *Van der Pol* oscillator model (see section 2.1.2) was selected. It was chosen because the literature search had highlighted it as a popular choice, plus it was relatively simple to implement. A discrete time version is presented in equation (5.2) using the simple Euler method to perform the numerical integration. Using the Euler method is simple to implement, and so was chosen, but more precise experiments could have been achieved with an improved numerical integration, such as *Runge-Kutta* [165].

$$\begin{cases} \dot{x}_i(t) = \dot{x}_i(t-1) + \Delta t\left(\alpha_i\left(p_i^{\,2} - x_i(t)^2\right)\dot{x}_i(t-1) - \omega_i^{\,2}\left(x_i(t) + \sum_{j=1}^{n}\lambda_{ij}x_j(t)\right) - k_i\right) \\ x_i(t+1) = x_i(t) + \dot{x}_i(t)\Delta t \end{cases} \tag{5.2}$$

where $x_i(t)$ is the output of oscillator i at iteration t, α_i, p_i, ω_i and k_i are constants to be tuned by optimisation, controlling shape, amplitude, frequency and amplitude respectively, λ is the interconnection matrix (with $\lambda_{ii} = 0$), Δt is the length of time between iterations, and n is the number of oscillators.

The parameter ranges used in this investigation, and encoding scheme used during optimisation, are given in Table 16 and Table 17.

Table 16: CPG parameter ranges.

Initial oscillator phase x	Set equal to the initial joint positions in the standing pose
Initial oscillator speed $\dot{x}$	[-4.0,4.0]
α	[0.0,4.0]
p^2	[0.0,8.0]
k	0
ω^2	[0.0,40.0]
λ_{ij}	[-1.0,1.0]

Table 17: CPG encoding scheme.

n constants α_i	n constants $p_i^{\,2}$	n constants ω^2	$n(n-1)$ matrix λ_{ij}

5.1.4 RNN – Fully connected leaky integrator

To compare with the CPG, a RNN from [86]–[88] with topology given in Figure 4 and described in equation (2.8) was chosen. This network was selected due to being well described in the literature, suitably generic and easy to implement. In [86]–[88] it was used for evolving continuous skills such as walking gaits so this experiment had the potential to establish a larger capability for this topology, if it were capable of discrete jumping skills.

Parameter ranges used during optimisation are given in Table 18.

Table 18: Parameter ranges for RNN

Initial oscillator phase x	Set equal to the initial joint positions in the standing pose
τ_j	[0.001,5.0]
w_{ij}	[-16.0,16.0]
α_j	[-4.0,4.0]

5.1.5 Optimisation

In this investigation a standard GA [128] was compared with a particle swarm optimisation (PSO) algorithm [129], for their ability to successfully develop counter-movement jumps (expressed as success rates). As mentioned in section 3.1.2, GA has often been used to optimise locomotion, and was used in [86] to generate walking gaits with the RNN. In contrast, no uses of PSO for this field were identified, and therefore it would be useful to compare the two techniques to start the process of establishing the most useful techniques for locomotion.

For the GA, the genes of each chromosome consisted of real values in the interval [0.0 , 1.0]. These specified the parameters for the control algorithms, being linearly mapped onto the desired parameter ranges. The configuration of the GA is shown in Table 19 with constants adapted from [86] after initial testing. The PSO was configured according to section 3.2.2.

Table 19: GA configuration

Population size	50 chromosomes
Chromosome length	Dependent on control algorithm
Randomisation	Normal distribution $\mu = 0.5$, $\sigma^2 = 0.5$
Selection	Tournament selection, $p = 0.9$
Crossover	Genes crossed $p = 0.01$
Mutation	Creep mutation from $N(0.5, 0.5)$, $p = 0.04$
Elitism	Best 4 chromosome copied unaltered

5.1.6 Fitness function

The fitness function for the vertical jump was kept as simple as possible, with the emphasis being placed on the methods to produce a solution. However, it was found that including some a-priori knowledge greatly helped results. The specification for the modified function is shown in Table 20.

Termination criteria can be used to frame the skill in time, to quickly stop failing attempts such as falling over, or to avoid undesirable movements such as jumping while attempting running. In this case, only the time frame was considered as important. It was found that a fitness function based on chest height was preferable to one based on waist height. This was because the waist based function ignored toppling of the upper body. The results using the modified version emphasising counter-movement is discussed in the next section.

Table 20: Fitness function for vertical jump.

Skill	**Fitness score**	**Termination criteria**
Vertical jump	Maximum height attained by chest Modified to favour counter-movement jumps by measuring the maximum height achieved by the chest after the waist had lowered below 0.65m	3 seconds of simulated time

5.1.7 Results

All control and optimisation combinations tested were able to produce satisfactory vertical jumps. An optimisation run was deemed to be successful if the final output

consisted of a counter movement jump. The frequency of successes and best jump heights varied across the different combinations of control and optimisation algorithms. The results from 25 optimisation runs are summarised in Table 21.

Table 21: Comparison of vertical jump evolution between GA and PSO, using CPG and RNN control methods.

Optimser	GA		PSO	
In 25 runs	Successes	Best height after 100 generations	Successes	Best height after 100 iterations
CPG	25	2.26m	25	2.55m
RNN	4	2.23m	7	1.54m

CPG runs were always successful but RNN controlled runs often failed to produce a success. When failure occurred, optimisation tended to hit a local maximum – typically using ankle plantarflexion to jump up a little. In an attempt to combat this, the fitness function was modified to include a threshold criterion. The waist had to lower below 0.65m before maximum height was measured. This favoured counter-movement jumps over ankle jumps. Even so, RNN control had a very poor success rate. Using a normal distribution approximation there was little evidence supporting a difference in the success rates of RNN optimisation between GA and PSO ($p>0.3$). However, this approximation is unreliable as the GA optimised RNN had fewer than 5 successes.

The results are suggestive of PSO optimisation producing more successes for RNN controlled jumps, with GA better at refining solutions to give greater heights. More simulations are needed to establish strong evidence of the PSO having better exploration (finding successful solutions) and the GA better exploitation (refining those solutions to get a near optimal height), and in contrast to the results for the RNN, PSO found the best height for the CPG controller.

The use of this modified fitness function includes a priori expert knowledge – that a counter-movement jump is preferential to maximum height. This is problematic as expert knowledge for other skills may be missing or wrong, and highlights the need to further develop the control and optimisation algorithms so that they can consistently find global maxima. In this experiment, only the CPG was able to find suitable solutions without expert knowledge.

The higher success rate of the CPG can be attributed to its explicit oscillatory form. Counter movements (cycling knee flexion to knee extension) are generally present in the first, random, population. Optimisation is then simply a process of identifying and refining these patterns. For the RNN, the optimisation task is more complicated as many configurations of the network do not produce any periods of oscillation at all.

Although very successful, the ease of optimising the CPG created instances of solutions that would not be appropriate for real life – even though they scored very high jumps. These solutions involved one or more preparatory mini-jumps before the final big jump. For real world applications such as robotic control or templates for human action, further modification to the fitness function may be needed to measure only the first jump, otherwise the form may be too energy demanding, time demanding, or aesthetically unpleasing. Both CPG and RNN algorithms were capable of producing unnecessary movements after the launch phase. This was often pronounced for the CPG as oscillations tended to continue after the launch phase. To improve the form, there would need to be some cut-off or transition control added to the CPG to control for these movements, and these form elements to be accounted for in the fitness function.

Lastly, it was observed that the CPG was capable of producing very explosive movements. This was possible because large amplitudes of oscillation were allowed (but clipped to control joint ranges). The movement pattern was probably not realistic but attempts to constrain the range more appropriately were less successful. Further work is needed to address this problem.

For use in a sport context, the simulation software can be used to produce a video of the optimised movement (Figure 18). This video can then be shown to coaches and athletes to help them visually understand the optimised pattern, or to be used as a reference in video analysis.

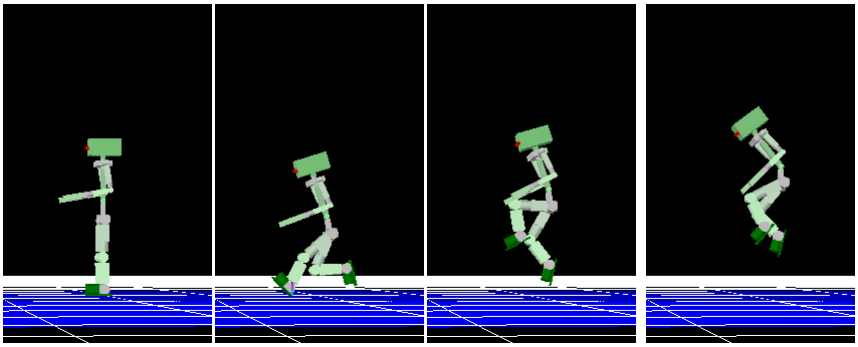


Figure 18: Frames from the animation of a successfully optimised vertical jump.

Alternatively, the raw output of the control algorithms (joint angles over time – Figure 19), or the calculated torques over time can be used in training. To do this, the traces can be compared to data collected using motion tracking techniques.

Figure 19: CPG output trace for the right knee angle during a vertical jump

Finally, the researcher could extract key features from either the video or control algorithm output data. Key features may include extrema positions of joints, timing information, and movement sequence description. Identification of important features can be done by comparing to pre-existing technique and highlighting major differences.

Fitness functions have to be carefully chosen because they can affect the ability of the system to optimise and fundamentally define the form of the final solution. In the vertical jump choosing to measure height achieved by the waist is successful in producing vertical jumps but, in general, the upper body tends to rotate towards the horizontal. Measuring height achieved by the chest corrects this problem. For a more realistic jump skill, it would be appropriate to add horizontal plane factors to the fitness function. This is to ensure that the direction of the jump is task appropriate.

In this experiment, the CPG consistently outperformed the RNN. It was always successful whereas the RNN sometimes failed to produce satisfactory results. However, the CPG does have issues with unsuitable movements (such as multiple jump patterns) that are less of a problem with RNN controlled movement. For GA optimised RNN jumping, the success rate was 1 in 10. This compares to a similar success rate for GA optimised RNN controlling a walk gait found in [86]. Although the movement skills are different the vertical jump can be viewed as possessing one cycle of an oscillation, and so the similar success rates may represent a similar difficulty in finding oscillatory patterns in the RNN using GA. The PSO found acceptable solutions for the RNN more often than the GA, but the GA generated the best height across the test for the RNN.

5.2 Walking gaits

This section presents investigations into producing and optimising walking gaits for quadrupeds and bipeds. Physical model design, control algorithms and optimisation techniques are explained, tested and contrasted.

Experiments were conducted using *Newton Dynamics* version 3.13 for physics simulation (driven with a time step of 0.02 seconds), *wxWidgets* for the user interface and *OpenGL* for graphical rendering.

5.2.1 Evolving quadruped walking gaits

To start with a less demanding walking task, the quadruped model was selected. As mentioned in section 3.4.1 it is statically stable and making it fall over is difficult. It was assumed that this property would make optimisation of gaits easier.

To control the quadruped, a sinusoidal *CPG* based on a body phase driven set of oscillators [70] was compared to a fully connected neural network ([86] and section 2.1.4). The CPG is given in equation (5.3).

$$\begin{aligned} \dot{\theta}_B &= \omega_B, \\ \dot{\theta}_i &= \omega_i + \lambda_i \left(\omega_B - \omega_i \right), \\ y_i &= x_i + a_i \sin\left(\theta_i \right), \end{aligned} \tag{5.3}$$

where θ_B and ω_B are the body phase angle, and phase speed respectively, θ_i is the angle of oscillator i, ω_i is the phase speed of oscillator i and λ is an entrainment factor. The output of the oscillator is given by y_i, with x_i and a_i being an offset and amplitude respectively.

The design of the *CPG* gives a clear interpretation for each parameter, and therefore the effects of changing them can be predicted to some degree, giving the potential for applying expert knowledge/reasoning to the configuration. The goal of this thesis was to develop methods that do not include expert knowledge, as the assumptions may be wrong, or may limit the solutions to sub-optimal results, but the opportunity was taken here to compare expert and non-expert adjusted configurations, in order to determine the added optimisational complexity of fully specifying the results by the optimisation process.

Therefore, two versions of the *CPG* were tested – one with the full configuration specified by the optimisation algorithm, and one where some parameters were hand coded. These were initial body phase (set to zero), body phase speed (set to one radian per second), oscillator offset (set to 0.5) and oscillator amplitude (set to 0.5). The justification for these manual adjustments was that initial body phase is somewhat arbitrary, a phase speed dictates the walking pace and so can be chosen freely, and that actuator movement would probably be best if full range and symmetrical.

The genetic algorithm (*GA*) and particle swarm optimiser (*PSO*) from chapter 5 were used, with the addition of *SRQEA* from chapter 4. The *GA* and *PSO* had 50 chromosomes or individuals respectively, and the *SRQEA* had a population size of 5 but performed 40 fitness evaluations per generation, as per the child spawning process outlined in chapter 4. The fitness function was simply distance travelled along the z-axis after 20 seconds of simulated time (1000 physical simulation iterations). Since the optimisation algorithms were configured to minimise the fitness function, this favoured movement in the negative direction. The body and hinges were aligned in the z-axis and had reflection symmetry in the x-axis so requiring positive or negative movement was arbitrary.

An additional termination condition was added that stopped simulation if the body tilted too far. This was done to stop processing simulations where the body flips. Initial experiments indicated this could save time, but it did not seem to be a necessary

condition for successful gait production. Nevertheless, it was included as a time saving measure.

5.2.2 Quadruped walking results

All methods produced walking gaits that transported the quadruped negatively along the z axis. However, some of these movement patterns relied upon friction to drag the body along, with some other legs moving in an opposing fashion. A well-functioning gait was one that moved diagonally opposite legs in unison and co-ordinated the two pairs of opposite legs. Rotating backwards, a pair of legs propel the model forward, while the other two legs retract and rotate forwards. This maintains dynamic stability while creating forward movement.

Detecting this behaviour would allow a 'success' status to be applied to runs, and in turn, would allow metrics based on success rates to be compared. Although an algorithm was not implemented to detect this directly, it was visually determined that distances of 2 metres in the negative z direction (a fitness score of -2.00) were generally associated with a strongly recognisable gait, as described above. Therefore, a run was regarded as successful if the fitness was less than or equal to -2.

The results of applying the *GA*, *PSO* and *SRQEA* optimisers to the expert adjusted *CPG* (labelled *simple CPG*), fully specified *CPG* (labelled *full CPG*) and *RNN* are presented in Table 22. Data presented are minimum, maximum, mean, median and standard deviation values, as well as SP and SR metrics described in section 4.4.3. Simulations were run for 200 iterations/generations and repeated for 100 runs, with no early termination criteria.

Only the *CPG* was able to produce a walking gait that travelled at least 2 metres (screen captures of a typical successful walk are shown in Figure 20) but managed this in both the *simple* and *full* versions. The *RNN* struggled to establish cyclic behaviour. This was achieved in [86] but only for a small percentage of runs. Additionally, the network was tested for its ability to copy sinusoidal waveforms (unpresented), which confirms the version coded in this work was capable of cyclic behaviour. It must therefore be regarded as a failure of the optimisation process that a *RNN* controlled walk of more than 2 metres was not obtained. As the failure occurred for all three optimisation algorithms, it is more likely to be a problem of optimiser configuration, than choice of

algorithm. This is discussed further when looking at time evolutions of the optimisation processes. When comparing the three control methods, it can clearly be seen that *simple CPG* outperforms the other methods, achieving a best score of -2.60m, followed by *full CPG* which achieved a best of -2.09m, with *RNN* having the worst performances with a best fitness of -1.09m.

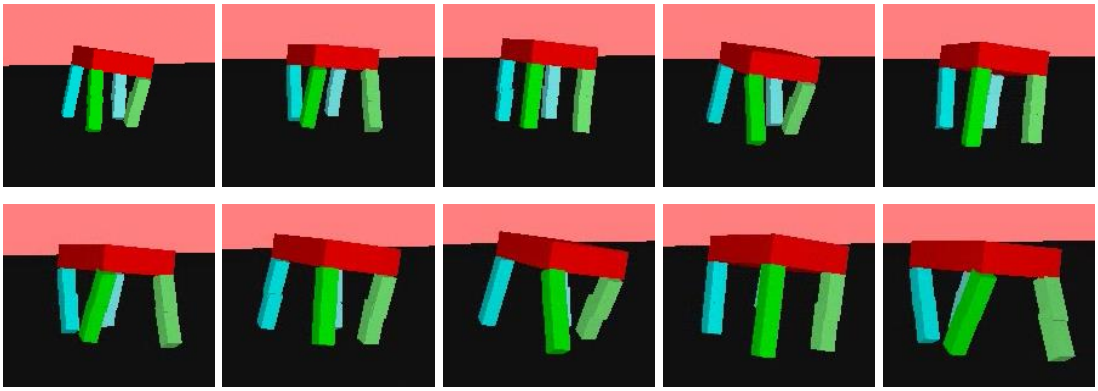


Figure 20: Typical evolved successful quadruped gait.

Overall the *GA* produced better results than *SRQEA* and *PSO*, achieving the best score for *simple Sin* and *full Sin*, but *PSO* scored the best for *RNN*. In general, the performance of the three optimisation algorithms was similar, with the exception of *PSO* for *full Sin* where the best value (minimum) was substantially worse than for the other two algorithms. However, looking at the average values for *full Sin*, good runs may be regarded as rare events and, as discussed below when looking at the time evolutions, may be an indication that more runs, over a longer period, may reduce the differences between the optimisers. Furthermore, *PSO* achieved the best minimum on the *RNN*, which probably presents the greatest optimisation challenge because of its open architecture, and this suggests that *PSO* is perhaps not fundamentally bad with *full Sin* but rather that the rare good results did not appear in the 100 runs performed for this work. The *GA* always had the best, or equal best, average performance and, as it also found the best values overall for two of the three controller configurations, should be regarded as the best performer in these tests.

Table 22: Comparison of GA, SRQEA and PSO optimisation of a quadruped gait controlled by a sinusoidal based CPG in an expert opinion assisted simple version (Simple Sin) and fully specified version (Full Sin), and a fully connected recurrent neural network (RNN).

		Simple Sin	Full Sin	RNN
GA	Min	-2.60	-2.09	-1.04
	Max	-1.50	-0.79	-0.60
	Mean	-2.39	-1.25	-0.84
	Median	-2.45	-1.18	-0.84
	Std	0.22	0.32	0.08
	SR	0.94	0.03	0.00
	SP	1379.26	6350.00	-
SRQEA	Min	-2.59	-1.98	-0.92
	Max	-1.16	-0.68	-0.40
	Mean	-2.29	-1.07	-0.67
	Median	-2.38	-1.05	-0.68
	Std	0.26	0.23	0.12
	SR	0.88	0.00	0.00
	SP	3658.52	-	-
PSO	Min	-2.46	-1.48	-1.09
	Max	-1.91	-0.68	-0.66
	Mean	-2.23	-0.94	-0.84
	Median	-2.25	-0.93	-0.83
	Std	0.10	0.15	0.07
	SR	0.99	0.00	0.00
	SP	1621.21	-	-

Most of the differences between the configurations were found to be statistically significant, with the test results shown in Table 23 using Kruskal-Wallis 1-way ANOVA in SPSS 23. The non-parametric Kruskal-Wallis test was used as many of the distributions were found to be non-normal. The data for *Simple Sin* were not found to be significantly different across the optimisation algorithms, but PSO did underperform GA for *full Sin*. SRQEA was statistically significantly worse for the *RNN* when compared to both GA and PSO.

Table 23: Pairwise significance tests between each combination of controller and optimiser, with another. Significant results are in bold, and numbers are adjusted p-value generated by SPSS.

	Simple Sin GA	Simple Sin SRQEA	Simple Sin PSO	Full Sin GA	Full Sin SRQEA	Full Sin PSO	RNN GA	RNN SRQEA	RNN PSO
Simple Sin GA		1.000	0.091	**0.000**	**0.000**	**0.000**	**0.000**	**0.000**	**0.000**
Simple Sin SRQEA	1.000		1.000	**0.000**	**0.000**	**0.000**	**0.000**	**0.000**	**0.000**
Simple Sin PSO	0.091	1.000		**0.000**	**0.000**	**0.000**	**0.000**	**0.000**	**0.000**
Full Sin GA	**0.000**	**0.000**	**0.000**		1.000	**0.000**	**0.000**	**0.000**	**0.000**
Full Sin SRQEA	**0.000**	**0.000**	**0.000**	1.000		1.000	**0.000**	**0.000**	**0.000**
Full Sin PSO	**0.000**	**0.000**	**0.000**	**0.000**	1.000		0.184	**0.000**	0.149
RNN GA	**0.000**	**0.000**	**0.000**	**0.000**	**0.000**	0.184		**0.000**	1.000
RNN SRQEA	**0.000**	**0.000**	**0.000**	**0.000**	**0.000**	**0.000**	**0.000**		**0.000**
RNN PSO	**0.000**	**0.000**	**0.000**	**0.000**	**0.000**	0.149	1.000	**0.000**	

Variants of *SRQEA* and *PSO* were also tested for *full Sin* and the results are presented in Table 24. *SRQEA10* had a population size of 10 but 4 offspring, compared to 5 individuals and 8 offspring for the default configuration. This maintained the number of evaluations per iteration at 40. *SRQEA10,20* performed 20 cross-overs per run compared with 4 for the default version. Overall differences between the SRQEA versions were small, although the mean performance was improved for both variants. *SPSO-2011* [123] is the same algorithm as contrasted in chapter 4, and produced sizable improvements for minimum and mean performance, but increased variance between runs.

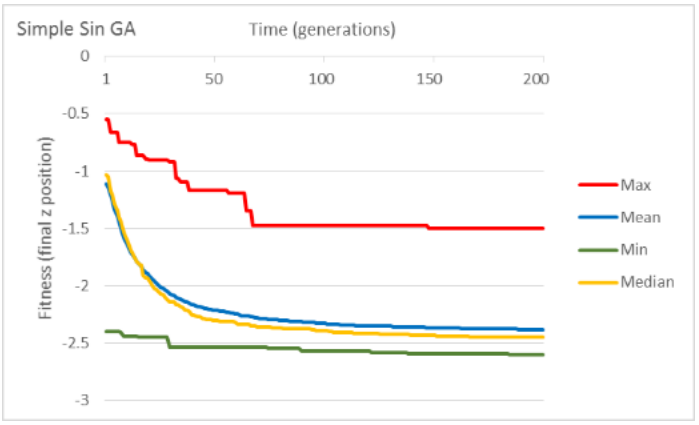
Table 24: Alternative versions of SRQEA and PSO applied to the full Sin controller for a quadruped walking gait.

	Full Sin				
	Min	Max	Mean	Median	Std
SRQEA	-1.98	-0.68	-1.07	-1.05	0.23
SRQEA10	-1.83	-0.72	-1.14	-1.11	0.21
SRQEA10,20	-1.89	-0.71	-1.13	-1.13	0.24
PSO	-1.48	-0.68	-0.94	-0.93	0.15
SPSO2011	-1.84	-0.54	-1.22	-1.18	0.25

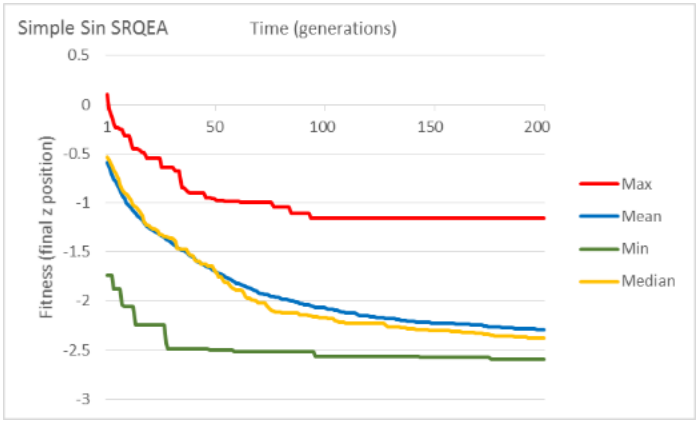
The time evolutions of each optimisation/control algorithm pair, are shown in Figure 21, where the horizontal time axis shows the generation/iteration, ranging from 1 – 200. The vertical axis shows the fitness score which measures z-direction distance travelled,

with lower scores being better. Quite a few of the plots show that the minimum value (green) is still decreasing close to the end of the evolution. This suggests that these algorithms would benefit from being run for longer. The PSO may especially benefit from a longer time as substantial improvements in minimum score were seen in the later stages for *full Sin* (where it underperformed relative to GA and SRQEA) and RNN (where it was the best performer). The evolution of SRQEA shows a continual, albeit slow, improvement of minimum value throughout the evolution, with a higher SP value suggesting modifications to speed convergence may be beneficial for this application of the optimisation algorithm.

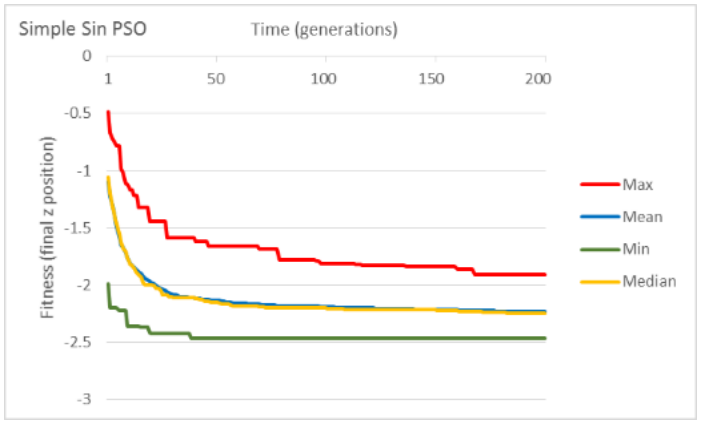
a)



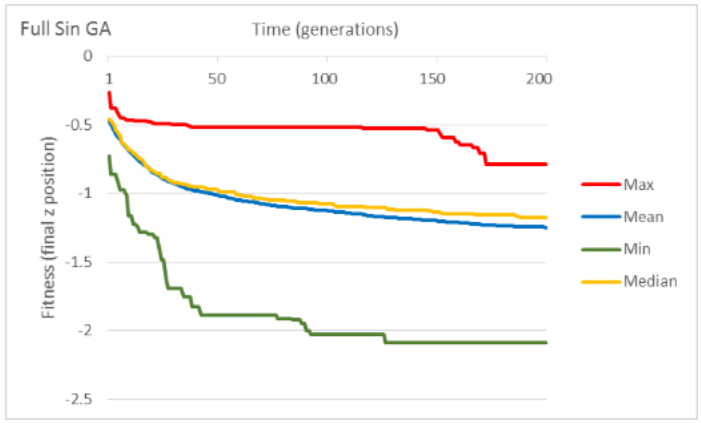
b)



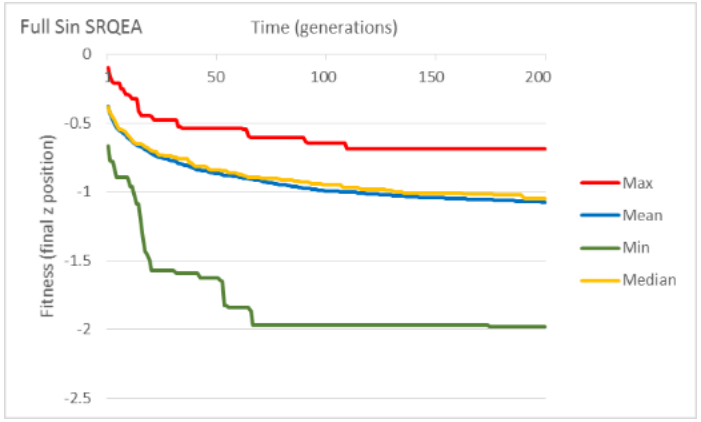
c)


Simple Sin PSO
Time (generations)
Fitness (final z position)
0
1
50
100
150
200
-0.5
-1
-1.5
-2
-2.5
-3
Max
Mean
Min
Median

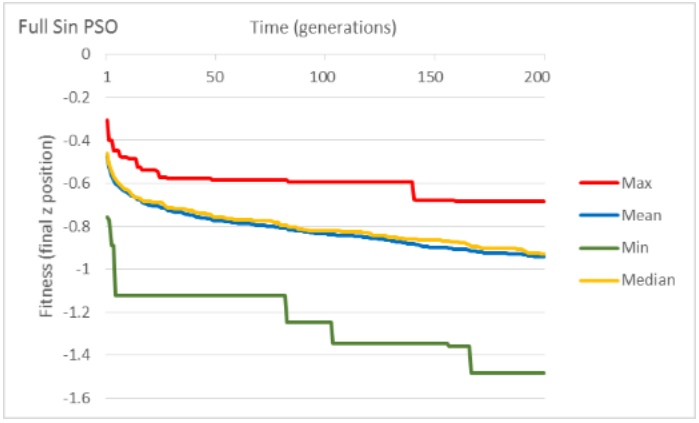
d)


Full Sin GA
Time (generations)
Fitness (final z position)
0
1
50
100
150
200
-0.5
-1
-1.5
-2
-2.5
Max
Mean
Min
Median

e)


Full Sin SRQEA
Time (generations)
Fitness (final z position)
0
1
50
100
150
200
-0.5
-1
-1.5
-2
-2.5
Max
Mean
Min
Median

f)


Full Sin PSO
Time (generations)
Fitness (final z position)
0
1
50
100
150
200
-0.2
-0.4
-0.6
-0.8
-1
-1.2
-1.4
-1.6
Max
Mean
Min
Median

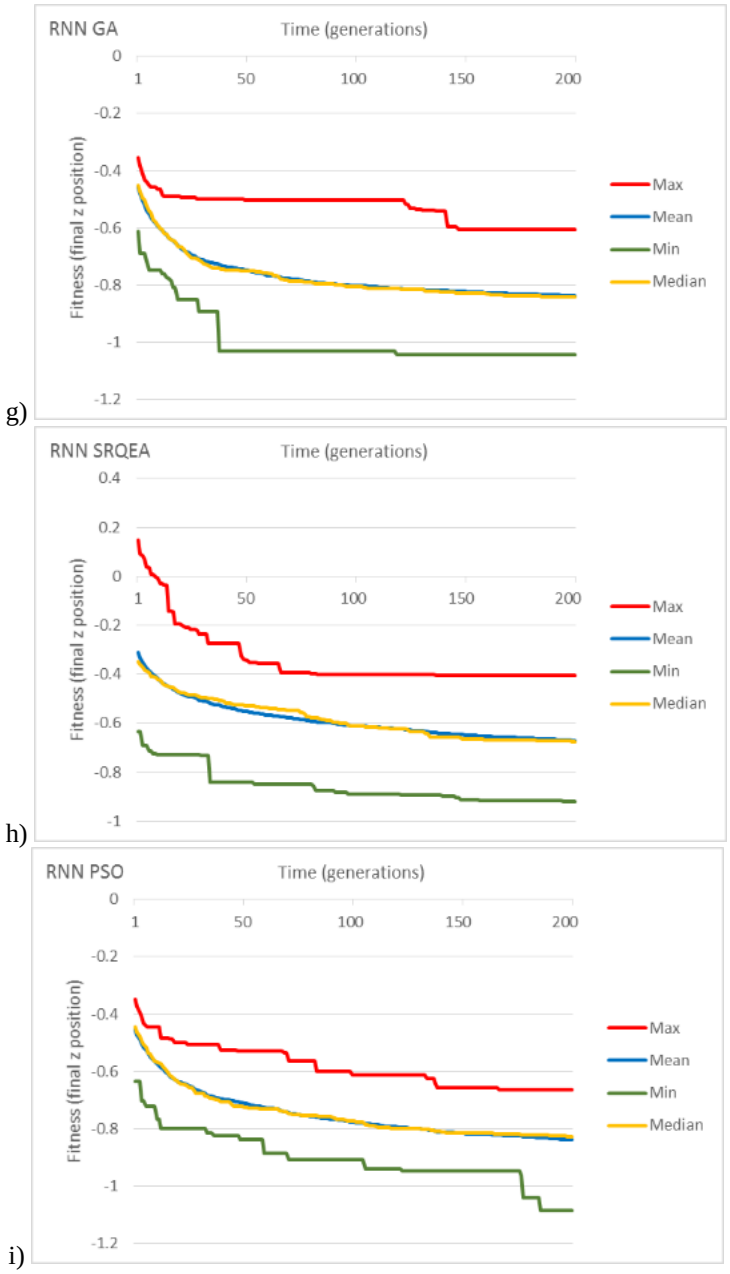


Figure 21: Time evolutions of each optimisation algorithm for each controller. Results for GA, SRQEA and PSO are shown, respectively, for Simple Sin (a – c), Full Sin (d – f) and RNN (g – i).

5.2.3 Evolving biped walking gaits

To make methods more applicable to human and many robot systems, biped models need to have their movement controlled and optimised. At the beginning of this chapter a vertical jump skill was successfully evolved, so in contrast, the continuous cyclical skill of walking was examined here. The simple biped model from section 3.4.2 was used as it was assumed the optimisation task would be more difficult than for a vertical jump, so a model with less degrees of freedom was used.

5.2.4 Difficulties

For a long time the investigation was unsuccessful at producing a convincing walking gait in a biped model. Optimisation would produce stumble and fall patterns, a series of hops, or somersaulting in an attempt to improve distance travelled. This was finally solved by altering characteristics of the physical model. The solution was to essentially copy the joint actuator algorithm from [86], which converts desired joint angles to torques based on the difference between the current angle, and the desired angle, with a degree of smoothing. This replaced the original hinge code that came with the physics simulation package (appendix C.1) and can be seen in appendix C.2. With this change, successful walking gaits were produced, albeit in a small percentage of runs.

Frames from an animation of a successful walk sequence are shown in Figure 22. Another problem was that foot contact with the ground formed point contacts which could not prevent rotations around the normal to the floor plain. This allowed the feet to twist in place and, strangely, the effective walking gaits started with a 90 degree spin before walking away. A few different feet designs were tried, such as sphere feet, multi-sphere and cuboid feet but all had the problem of producing a single point contact at certain times. In the real-world, feet will deform when touching the floor and create an area of contact, which in turn will provide friction against these rotations around the normal. This problem was not solved in this experiment, but future work is suggested in the conclusion section to address this. Towards the end of the simulation time for the gait the model appeared to be becoming unbalanced. This was quite possibly due to accumulated error and therefore a minor detail rather than a fundamental problem with the solution. To improve upon this, a finer evolutionary strategy could be employed to precisely specify the gait, or sensory feedback could be incorporated to help maintain balance, but was left for future work.

5.2.5 Bipedal walking results

The fitness function was the negative square of distance travelled in the *x*-*z* plane when termination occurred, favouring walking in any direction. A run was terminated if the simulation time had elapsed, or if the body fell below a threshold height. It is difficult to present a quantitative analysis here as fitness scores were often due to erroneous

behaviour. For example, although distance travelled could be substantial, the form used was sometimes not a walking gait – hopping or spinning gaits were often observed. However, once the new joint model was implemented, a set of reliable data could be produced, and basic information presented that could be relied upon.

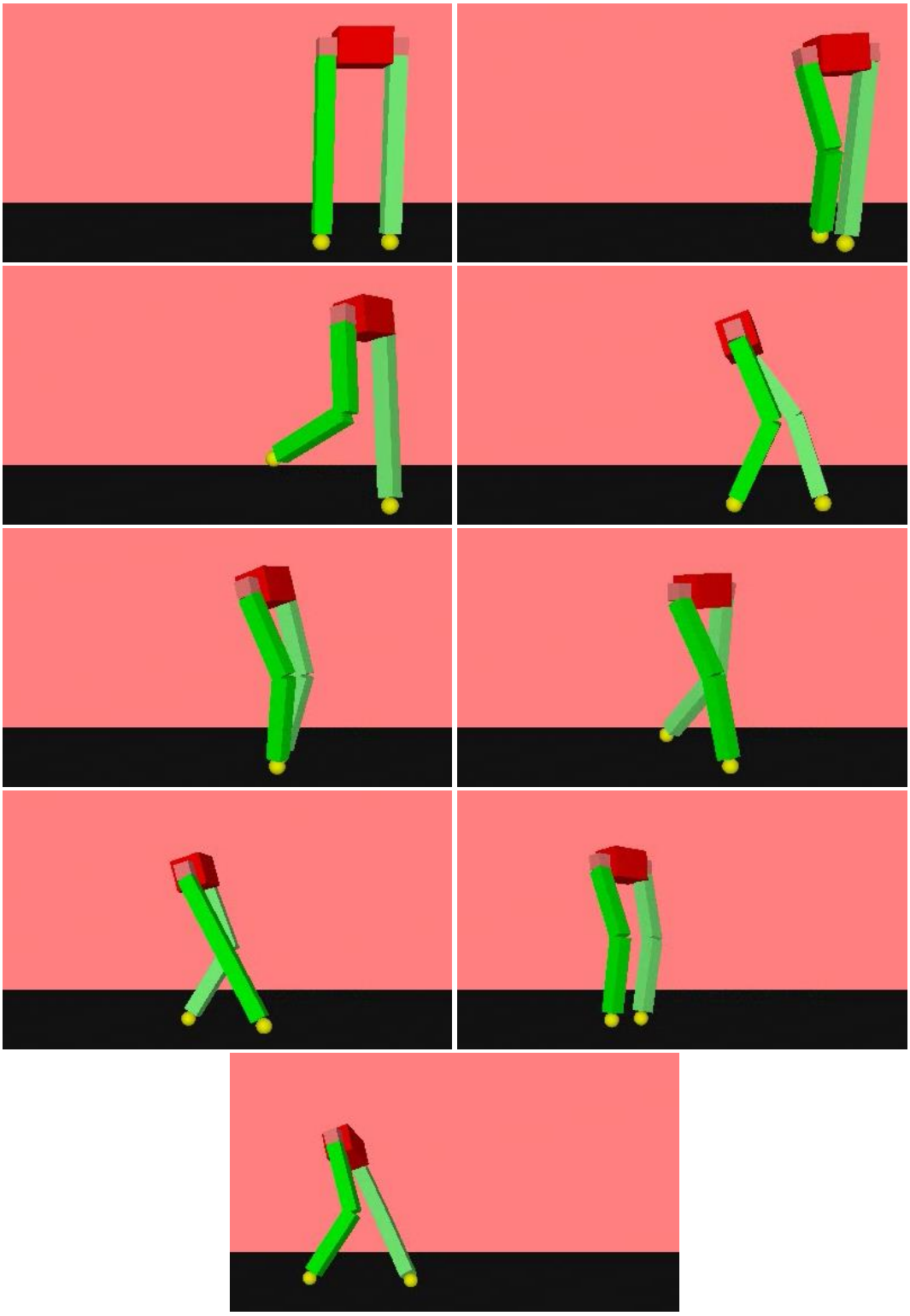


Figure 22: Successful bipedal walking gait.

In Table 25 data are shown for successful evolutions of a biped gait for a *CPG* (configured the same as *full Sin* previously) and the fully connected *RNN*. A threshold of -3.00m was taken as successful, as by inspection it appeared that this was associated with a proper walking gait. The authors of [86] reported a success rate of approximately 0.10 so the results of 0.15 for the *CPG*, and 0.06 for the *RNN* demonstrate a similar order of magnitude of success. Not surprisingly, the *CPG* achieved the correct, cyclic behaviour more often than the *RNN*, and had a better minimum score.

Table 25: Walking evolution performance for biped model.

		CPG	RNN
GA	Min	-4.46	-3.41
	Mean	-2.36	-2.06
	SR	0.15	0.06

5.3 A long jump skill

To conclude the investigation into biped control, a brief examination of a new skill – long jumping was conducted. This gave an opportunity to demonstrate a wider range of skill production, and also the results highlighted difficulties with correctly expressing fitness functions and how optimisation may exploit characteristics in the physical model to produce unrealistic movements.

The fitness function for the long jump was the furthest distance travelled in the negative z direction by either foot at termination time. Termination time was either when a foot hit the ground beyond a start line (0.1m in front of the system along the negative z direction) or when the upper body hit the ground. The intention was to evolve a jump and land phase, favouring distance travelled in the air before landing. This was partially successful but the evolved movement technique had some undesirable features.

Shown in Figure 23 the long jump began with a counter-movement, bending and then extending the knees. Although some propulsion will have come from this movement, most came from a flinging action by one of the legs. It quickly swung forwards and up, lifting and propelling the model forward. As well as being unrealistic compared to expected movement from humans, this action created a spinning effect on the body. Finally, the other leg pointed forwards to achieve a good fitness score. The odd form, compared to human jumping technique, is probably due to two factors. Firstly, the

physical model is not very accurate, and probably favours ballistic single limb movements, rather than co-ordinating body parts to propel the system. Secondly, the fitness function may benefit from having certain form elements included, in order to prevent spinning (although, if arms were present they may make contact with the ground, so automatically would penalise these patterns).

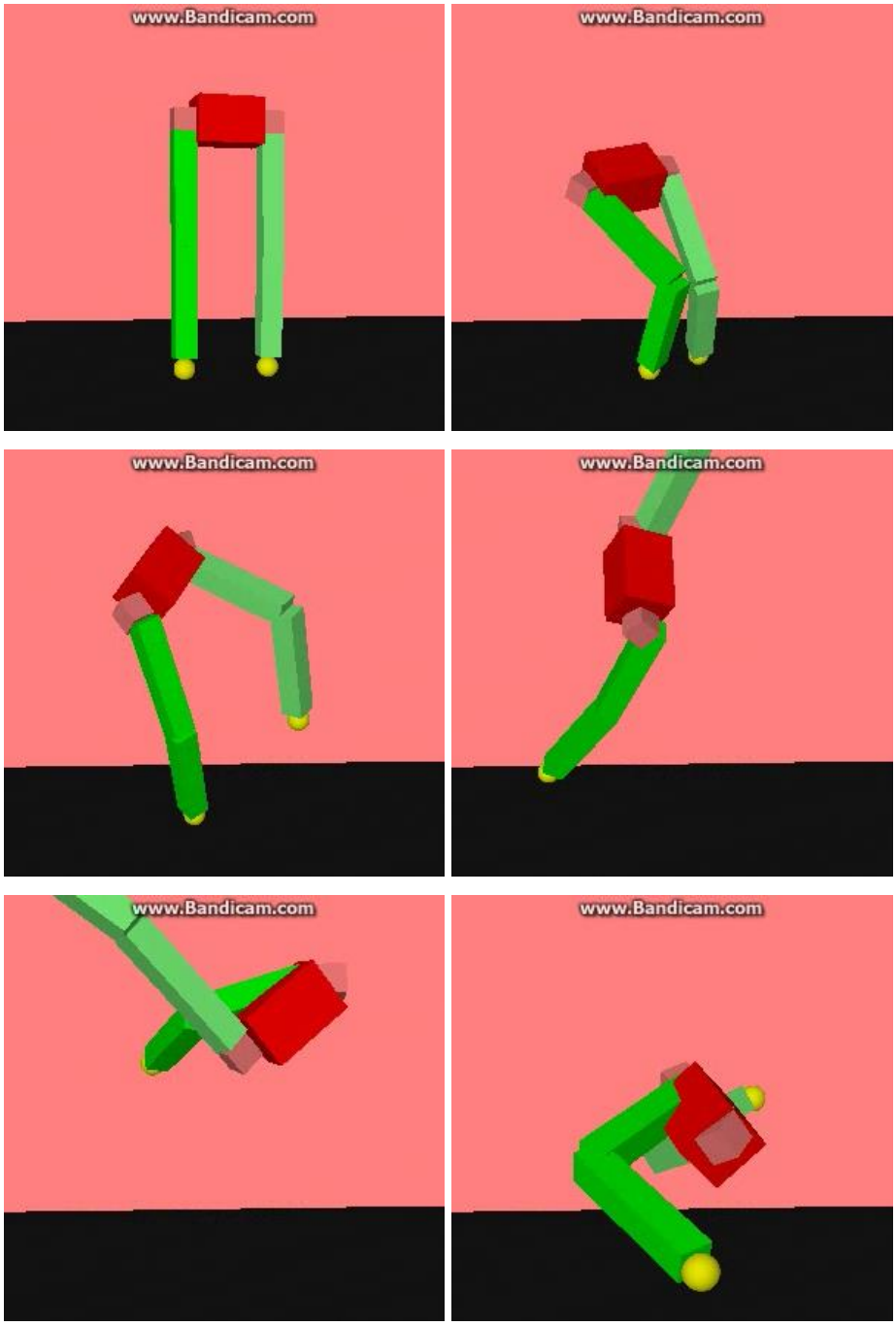


Figure 23: Typical long jump evolution.

5.3.1 Conclusion

In this chapter research results have been presented applying three different optimisation techniques to locomotion – GA, PSO and SRQEA. Results were mixed, with GA achieving the best results in walking tasks and some of the vertical jump and long jump tests, but PSO demonstrated superior results for the vertical jump in terms of best height obtained and the number of successful evolutions when using a RNN control method, and the best results for RNN controller quadrupedal walking. SRQEA did not perform as well in general, never attaining the best results, although it did outperform PSO and the SPSO-2011 algorithms for quadruped walking with a fully specified Sin controller.

It should be noted that GA, PSO and SRQEA algorithms have several variables and sub-functions that can be changed. This will all have an impact on exploitation versus exploration. For example, some versions of PSO systematically vary their parameters over the course of the run, so that they favour exploration in the beginning but exploitation towards the end. Therefore, conclusions on the overall properties of the optimisation algorithms tested here in comparison to each other cannot be made just on the experiment presented in this thesis.

Evolving vertical jumping gaits was quite successful, in both Newton Dynamics and OpenHRP3. Even in these discrete tasks, the CPG proved very successful. This can be understood by interpreting the counter-movement jump as one cycle in a potentially continuous process, with the coordination of various limbs acting out across that cycle.

When using the RNN controller for the vertical jump. Expert knowledge was required to ensure the counter-movement property. The process of including expert knowledge risks constraining the problem to sub-optimal solutions. Using a sinusoidal based controller, evolving a walking gait was relatively easy for quadrupeds. The fitness function only analysed distance travelled which is desirable, as it is an outcome measure and assumes nothing about desired form. However, best performance came when manually specifying some of the parameters of the sinusoidal CPG, which violates the principal of making minimal assumptions. Performing 100 runs typically took more than 10 hours to complete, but extending running time may be beneficial in finding good solutions, as seen by the time evolution plots. A two phase or hybrid technique

may be employed to aggressively search early on, as the rare-event nature of *full Sin* suggests adequate solutions occupy a small region of the search space when optimising more open control architectures, and then refine the results with a search focussed on exploitation.

The biped model is inherently less stable than the quadruped, and therefore movement control should be expected to be more difficult. The success of the simulations was also sensitive to the specification of the physical model and future work should include more accurate modelling to overcome these issues. Success rates for bipedal walking was low, but similar to other published research [86]. It was easier to evolve walking gaits with the CPG, which is not surprising given the oscillatory nature of the gait. Long jumping had some elements of success but was hampered by poor form – which would lead to unachievable or destructive movement in real-world applications such as robotics.

To expand upon the biped research presented here, firstly the foot contact problem should be solved. This may be done by creating a temporary hinge joint while the foot is in contact with the floor, but one that allows translation away from the floor plane so that the foot can lift away. Following that, additional optimisation and control techniques should be investigated to establish the more successful method combinations.

Once a suitable solution has been found, it can be utilised by either copying the control algorithm to the real-word robotic host, or taking information that may be communicated in order to train a human. Video of the evolution is easy to generate from the simulation software, and sample frames have been included in this chapter as illustration. Alternatively, joint trajectories or body part trajectories can be plotted, although for training purposes these will probably require significant transformation or interpretation to be understood.

6 Conclusions and future directions

This chapter presents a conclusion based on how the work presented in this thesis meets the objectives outlined in the introduction. It also presents recommendations for future work to further these objectives, and concludes with a list of publications produced during the development of this work.

6.1 Conclusions

The first objective of this thesis was to review published methods for locomotion control, and this process identified several techniques (chapter 2). These were traditional analytical methods, CPGs, neural networks, rule based systems, and HMMs. As optimisation was the primary theme of the thesis objectives, analytical methods were not used in subsequent research because of the constraints imposed on parameterisation. Also, the simplification imposed by the modelling process would limit analytical techniques in their application to detailed questions on optimisation of complex systems. For similar reasons, rule based systems were also rejected. In principal they could be detailed but their use in published literature demonstrated a sizeable degree of simplification. HMMs were used solely for imitation tasks and so were also not included in experimentation. This left CPGs and NNs as suitable candidates to test, as they provided extensive parameter optimisation potential and had demonstrated successful results in the literature.

To meet the thesis objective of comparing different control methods, which is generally lacking in the literature, experiments were conducted using a range of techniques. A fully connected RNN, Van der Pol CPG and sinusoidal CPG were compared. These proved to be popular in the literature, and the sinusoidal CPG was easy to modify to include expert knowledge regarding symmetry constraints.

Many optimisation techniques to accompany the use of the control methods in locomotion had been identified in the literature (section 3.1) but these were either simple optimisation algorithms, specialised learning algorithms to use in real-world experiments, or *GA*. The optimisation objectives of this thesis were to review, compare and add new methods. The review clearly identified a gap in application of different optimisation algorithm types so it was decided to add different methods and compare to GAs in experimentation. These were a standard *PSO*, and a type of *EDA* called *SRQEA*.

SRQEA was developed for this thesis (chapter 4) to meet the objective of developing new optimisation variants, from a multi-modal *EDA* called a quantum inspired evolutionary algorithm. In developing this, a binary variant was also produced called *HSB*, as *QIEAs* were originally presented as binary algorithms. Both of the new *QIEAs* were tested on real-value problems as the objectives of this thesis required parameterisation of real-valued control methods.

Binary *QIEA* were examined (chapter 4) because the inspiration behind this class of algorithm is most coherently expressed when using a binary encoding. The problem with applying *bQIEA* to real-value problems is that premature convergence of the least significant bits tends to occur (due to reinforced random walk processes). A modification was proposed and tested that restricted evolution of these bits until the more significant bits had saturated their probabilities of producing ones or zeros. This modification improved the performance of the algorithm and, although generally outperformed by *rQIEA*, had some interesting successes that may be due to the way a binary algorithm partitions and transverses the search space.

The real *QIEA* identified in the literature often suffered from poor specification in their presentation, but a representative algorithm, *RCQIEA*, was examined. The formula for its rotation gate (the method by which the underlying search probability function is updated) produced wild, and effectively random updates. Replacing this with a simple step function improved performance and the resulting *SRQEA* method performed well against other published algorithms (including *GA*, *PSO* and *DE* methods), achieving superior performance for the *CEC-2013* benchmarks for the larger dimensions.

The process of optimisating locomotion is performed typically (although not exclusively in the literature) in a virtual environment, as faster than real-time simulation

speeds up optimisation, and control failure does not cause real damage. An objective of this thesis was to review physical simulation platforms and develop physical models for experimentation. The literature search identified OpenHRP3 as a common platform, but otherwise, platforms were not possible to determine, or were no longer available. A small review (section 3.3) of software available online identified game orientated packages Newton Dynamics and Unity (which also provided and application platform), robotics platform OpenHRP3, and biomechanics software SIMM and LifeMod. The biomechanics packages would be useful when techniques progress to being able to handle the level of detail in their models (with associated high degrees of freedom), but were not included for the research in this thesis. OpenHRP3 was selected in keeping with its profile in the literature search, and Newton Dynamics was selected as a fast and free alternative.

OpenHRP3 provided a robotic biped model which was used to evolve a vertical jump. Additional models designed and presented in section 3.4 were a simple quadruped model, a simple biped and a more detailed biped, all for use in the Newton Dynamics platform. They were designed with specific goals in mind. The simple quadruped was the most stable, being statically stable in most conditions, and was included for use when researching difficult gaits before moving on to more complicated systems. It was used to evolve a walking gait and enabled extensive quantitative comparison between different control methods and optimisation techniques. The simple biped addressed a similar need for bipedal investigation. Although not as stable as the quadruped, it had a low number of degrees of freedom and was used for walking and long jump skills. Finally, a more detailed biped model was used, with more degrees of freedom and detailed upper as well as lower body modelling in comparison to the simple biped model. It was used for the vertical jump skill.

A significant discovery was the sensitivity of the overall optimisation process to the quality of the physical simulation. The approach used in this thesis combines physical modelling through simulation, a control algorithm and an optimisation technique. In principal, any of these could be a weak link and prevent successful evolutions, but the experience of the bipedal model when used for walking and long jumping suggests that the physical simulation can provide the biggest problems. As these techniques are applied to real-world modelling, this should hopefully become less of a problem, as the physical modelling will be more validated, rather than using the informal model design

approach employed here. Nevertheless, the importance of correct physical simulation was highlighted across all tests on the Newton Dynamics platform. Problems encountered included explosive movements due to joint activation speeds becoming too high, and poor foot-floor interaction with point contacts not preventing rotations that real-world area contacts would do.

In general, the literature review identified only transportation locomotion skills (walking, running, swimming). To meet the thesis objective of developing fitness functions for new skills, a vertical jump and a long jump skill were developed in chapter 5, these two skills having not been observed during the literature review. In general, encoding the fitness functions was quite simple – an outcome of maximum height in a set time, impact point of lead leg or maximum distance in a set time in vertical jumping, long jumping and walking respectively. However, in some instances, expert knowledge needed to be included, such a criterion to develop a counter-movement jump or symmetry simplifications in walking, to enable successful evolution with some skill/control method/optimisation technique combinations. As noted in chapter 5 the inclusion of expert knowledge risks development of sub-optimal solutions as they effectively impose constraints on the process. However, when applying the patterns, constraints may be useful to reject certain patterns on aesthetic, practicality or safety grounds.

For jumping and walking skills the *CPGs* were the most consistent performers. Their structures are custom designed for oscillatory behaviour so should be expected to perform well for continuous walking control, but they were also successful in producing vertical jumps (and partially long jumps) which are discrete short action skills. Counter-movements are often advantageous to skills, whether continuous or discrete and so some form of oscillation is clearly beneficial. The *RNN* investigated had low success rates, lower mean and fewer best performances in general, and the inclusion of expert knowledge appeared necessary in some of the experiments. There may be skills that are more suited to this type of structure, but alternative neural network designs may be worth investigating in the future.

The model design decisions were suited to the type of evolution tasks performed. As expected, the vertical jump has capable in being evolved in the more detailed biped models both on Newton Dynamics and OpenHRP3. The walking tasks were more

difficult but the stable quadruped model allowed extensive testing and quantitative comparison between methods. Experiments suffered from poor specification of actuator behaviour, and friction models used for feet-floor interactions. It was not until the actuator models were improved that evolution could be successful for vertical jumping and walking. Long jumping had some elements of success but was hampered by poor form – which would lead to unachievable or destructive movement in real-world applications such as robotics. It is expected that improvements to the physical modelling would correct these problems.

For optimisation, *GA* outperformed the other techniques in most of the experiments, although there appeared to be an affinity between optimisation technique and control method combinations, with *PSO* giving the best results for the RNN. *SRQEA* did not dominate in any of the tests but it did outperform *PSO* and *SPSO-2011* variants in some of the quadruped walking experiments. The objective of expanding upon the number of optimisation algorithms applied to locomotion has been met with this research, but it is too early to tell if any algorithm should be preferred. All of them can be configured or parameterised differently, presenting many configurations worth testing.

Although it has been demonstrated that a vertical jump skill can be evolved in a more detailed model, more work needs to be done to successfully evolve a range of skills in complex models. This may be a case of better physically modelling, but improved control methods and optimisation techniques will probably be required as well.

6.2 Future directions

Substantial problems were encountered with the physical modelling during experiments. Not until the joint actuators were altered, could results be produced for jumping and walking that did not get stuck in evolving odd jumping or other ballistic patterns. Furthermore, the rigid body simulations often produce point contacts for the feet upon the floor. In normal usage, this will not prevent rotations. In the real world, feet tend to form area contacts which will resist rotations relative to the floor. It is therefore strongly recommend that better physical designs are established for experimentation, focussing on joint and actuator modelling so systems have a range of

capabilities, but do not produce excessively ballistic movement, and improved feet models so proper friction dynamics can be modelled.

The original inspiration for this thesis was the optimisation of sport skills in realistic models of human movement. As the techniques are also applicable to robot models, which in principal could feature more complicated designs in the future, a continued drive for this work should be the evolution of gaits in increasingly complex models. This can be done with the development of more complicated models, or by adopting pre-existing complex models. Publically available biomechanical models, of varying complexity, could be used to test methods and hopefully produce results relevant to humans, and the resulting methods should be powerful enough to apply to non-human or mechanical models.

The literature search highlighted a lack of comparative studies, and future work should build upon this thesis by continuing comparison between optimisation techniques and control methods. Only one neural network design was tested here and it had low success rates. It would be useful to test others - for example, reservoir networks may be worth study and comparison with other techniques, as they provide both a flexible structure and should be easier to optimise as only the output weights are evolved.

Although comparison of different optimisation techniques was presented in this thesis, optimisation algorithms used either traditional default parameter values, or values taken for other publications. Although possibly requiring a great deal of simulation time, it would be important to analyse the effect of varying parameters, introducing schemes to control diversity, or combining techniques into hybrid methods.

6.3 List of publications

Journal

J. Wright and I. Jordanov, ‘Convergence Properties of Quantum Evolutionary Algorithms on High Dimension Problems’, Neurocomputing, under submission.

J. Wright and I. Jordanov, ‘Quantum Inspired Evolutionary Algorithms’ Performance on Challenging Global Optimization Problems’, ICAE, under submission.

J. Wright and I. Jordanov, ‘Intelligent Approaches in Locomotion - A Review’, J. Intell. Robot. Syst., vol. 80, no. 2, pp. 255–277, Oct. 2014.

Conference

J. Wright and I. Jordanov, ‘Quantum Evolutionary Methods for Real Value Problems’, in Hybrid Artificial Intelligent Systems, Springer, 2015, pp. 282–293.

J. Wright and I. Jordanov, ‘Intelligent computational optimisation of sport skills’, Math Sport 2013, pp. 383-390

J. Wright and I. Jordanov, ‘Intelligent approaches in locomotion’, in Neural Networks (IJCNN), The 2012 International Joint Conference on, 2012, pp. 1–8.

7 References

[1] M. Kociecki and H. Adeli, 'Two-phase genetic algorithm for size optimization of free-form steel space-frame roof structures', *J. Constr. Steel Res.*, vol. 90, pp. 283–296, Nov. 2013.

[2] M. Kociecki and H. Adeli, 'Two-phase genetic algorithm for topology optimization of free-form steel space-frame roof structures with complex curvatures', *Eng. Appl. Artif. Intell.*, vol. 32, pp. 218–227, Jun. 2014.

[3] T. Chabuk, J. Reggia, J. Lohn, and D. Linden, 'Causally-guided evolutionary optimization and its application to antenna array design', *Integr. Comput.-Aided Eng.*, vol. 19, no. 2, pp. 111–124, Jan. 2012.

[4] M. R. AlRashidi and M. E. El-Hawary, 'A Survey of Particle Swarm Optimization Applications in Electric Power Systems', *IEEE Trans. Evol. Comput.*, vol. 13, no. 4, pp. 913–918, Aug. 2009.

[5] H. Tao, J. M. Zain, M. M. Ahmed, A. N. Abdalla, and W. Jing, 'A wavelet-based particle swarm optimization algorithm for digital image watermarking', *Integr. Comput.-Aided Eng.*, vol. 19, no. 1, pp. 81–91, Jan. 2012.

[6] W.-Y. Hsu, 'Application of quantum-behaved particle swarm optimization to motor imagery EEG classification', *Int. J. Neural Syst.*, vol. 23, no. 6, p. 1350026, Jul. 2013.

[7] S. Das and P. Suganthan, 'Problem definitions and evaluation criteria for CEC 2011 competition on testing evolutionary algorithms on real world optimization problems', *Jadavpur Univ. Nanyang Technol. Univ. Kolkata*, 2010.

[8] X. Yao, Y. Liu, and G. Lin, 'Evolutionary programming made faster', *IEEE Trans. Evol. Comput.*, vol. 3, no. 2, pp. 82–102, Jul. 1999.

[9] B. Y. Q. J. J.Liang, 'Problem Definitions and Evaluation Criteria for the CEC 2013 Special Session on Real-Parameter Optimization', *Tech. Rep. 201212 Comput. Intell. Lab. Zhengzhou Univ. Zhengzhou China*, 2013.

[10] E. J. Sprigings and S. J. Mackenzie, 'Examining the delayed release in the golf swing using computer simulation', *Sports Eng.*, vol. 5, no. 1, pp. 23–32, 2002.

[11] P. S. Glazier and K. Davids, 'Constraints on the Complete Optimization of Human Motion', *Sports Med.*, vol. 39, pp. 15–28, 2009.

[12] J. Hore and S. Watts, 'Timing Finger Opening in Overarm Throwing Based on a Spatial Representation of Hand Path', *J. Neurophysiol.*, vol. 93, pp. 3189–3199, Jun. 2005.

[13] S. Leigh, H. Liu, M. Hubbard, and B. Yu, 'Individualized optimal release angles in discus throwing', *J. Biomech.*, vol. 43, no. 3, pp. 540–545, Feb. 2010.

[14] M. Damsgaard, J. Rasmussen, S. T. Christensen, E. Surma, and M. de Zee, 'Analysis of musculoskeletal systems in the AnyBody Modeling System', *Simul. Model. Pract. Theory*, vol. 14, pp. 1100–1111, 2006.

[15] E. Arnold, S. Ward, R. Lieber, and S. Delp, 'A Model of the Lower Limb for Analysis of Human Movement', *Ann. Biomed. Eng.*, vol. 38, pp. 269–279, Feb. 2010.

[16] F. Asano and Z.-W. Luo, 'Energy-Efficient and High-Speed Dynamic Biped Locomotion Based on Principle of Parametric Excitation', *Robot. IEEE Trans. On*, vol. 24, pp. 1289–1301, Dec. 2008.

[17] Y.-D. Kim, B.-J. Lee, J.-H. Ryu, and J.-H. Kim, 'Landing Force Control for Humanoid Robot by Time-Domain Passivity Approach', *Robot. IEEE Trans. On*, vol. 23, pp. 1294–1301, Dec. 2007.

[18] B.-J. Lee, D. Stonier, Y.-D. Kim, J.-K. Yoo, and J.-H. Kim, 'Modifiable Walking Pattern of a Humanoid Robot by Using Allowable ZMP Variation', *Robot. IEEE Trans. On*, vol. 24, pp. 917–925, Aug. 2008.

[19] S. Czarnetzki, S. Kerner, and O. Urbann, 'Observer-based dynamic walking control for biped robots', *Robot. Auton. Syst.*, vol. 57, pp. 839–845, 2009.

[20] J. Xiao, J. Su, Y. Cheng, F. Wang, and X. Xu, 'Research on gait planning of artificial leg based on central pattern generator', presented at the Control and Decision Conference, 2008. CCDC 2008. Chinese, 2008, pp. 2147–2151.
[21] G. C. Nandi, A. J. Ijspeert, P. Chakraborty, and A. Nandi, 'Development of Adaptive Modular Active Leg (AMAL) using bipedal robotics technology', *Robot. Auton. Syst.*, vol. 57, pp. 603–616, 2009.
[22] V. B. Zordan, A. Majkowska, B. Chiu, and M. Fast, 'Dynamic response for motion capture animation'. Jul-2005.
[23] Y. Lee, K. Lee, S.-S. Kwon, J. Jeong, C. O'Sullivan, M. S. Park, and J. Lee, 'Push-recovery Stability of Biped Locomotion', *ACM Trans Graph*, vol. 34, no. 6, p. 180:1–180:9, Oct. 2015.
[24] A. J. Ijspeert, J. Hallam, and D. Willshaw, 'Evolving swimming controllers for a simulated lamprey with inspiration from neurobiology', *Adapt. Behav.*, vol. 7, p. 151, 1999.
[25] T. Ha and C.-H. Choi, 'An effective trajectory generation method for bipedal walking', *Robot. Auton. Syst.*, vol. 55, pp. 795–810, 2007.
[26] H. Hirukawa, F. Kanehiro, K. Kaneko, S. Kajita, K. Fujiwara, Y. Kawai, F. Tomita, S. Hirai, K. Tanie, T. Isozumi, K. Akachi, T. Kawasaki, S. Ota, K. Yokoyama, H. Handa, Y. Fukase, J. ichiro Maeda, Y. Nakamura, S. Tachi, and H. Inoue, 'Humanoid robotics platforms developed in HRP', *Robot. Auton. Syst.*, vol. 48, pp. 165–175, 2004.
[27] J. Wright and I. Jordanov, 'Intelligent Approaches in Locomotion - A Review', *J. Intell. Robot. Syst.*, vol. 80, no. 2, pp. 255–277, Oct. 2014.
[28] M. Vukobratovic and B. Borovac, 'Zero-moment point-thirty five years of its life', *Int. J. Humanoid Robot.*, vol. 1, pp. 157--173, 2004.
[29] T. Sugihara, Y. Nakamura, and H. Inoue, 'Real-time humanoid motion generation through ZMP manipulation based on inverted pendulum control', vol. 2, pp. 1404–1409 vol.2, 2002.
[30] Y. Choi, D. Kim, Y. Oh, and B.-J. You, 'Posture/Walking Control for Humanoid Robot Based on Kinematic Resolution of CoM Jacobian With Embedded Motion', *Robot. IEEE Trans. On*, vol. 23, pp. 1285–1293, Dec. 2007.
[31] F. Kanehiro, H. Hirukawa, and S. Kajita, 'Openhrp: Open architecture humanoid robotics platform', *Int. J. Robot. Res.*, vol. 23, p. 155, 2004.
[32] J.-K. Yoo, B.-J. Lee, and J.-H. Kim, 'Recent progress and development of the humanoid robot HanSaRam', *Robot. Auton. Syst.*, vol. 57, pp. 973–981, 2009.
[33] J. H. Park, 'Fuzzy-logic zero-moment-point trajectory generation for reduced trunk motions of biped robots', *Fuzzy Sets Syst.*, vol. 134, pp. 189–203, 2003.
[34] T. Furuta, T. Tawara, Y. Okumura, M. Shimizu, and K. Tomiyama, 'Design and construction of a series of compact humanoid robots and development of biped walk control strategies', *Robot. Auton. Syst.*, vol. 37, pp. 81–100, 2001.
[35] A. Goswami, 'Postural stability of biped robots and the foot-rotation indicator (FRI) point', *Int. J. Robot. Res.*, vol. 18, p. 523, 1999.
[36] M. B. Popovic, A. Goswami, and H. Herr, 'Ground reference points in legged locomotion: Definitions, biological trajectories and control implications', *Int. J. Robot. Res.*, vol. 24, p. 1013, 2005.
[37] S. Kagami, M. Mochimaru, Y. Ehara, N. Miyata, K. Nishiwaki, T. Kanade, and H. Inoue, 'Measurement and comparison of humanoid H7 walking with human being', *Robot. Auton. Syst.*, vol. 48, pp. 177–187, 2004.
[38] H. M. Herr and T. A. McMahon, 'A trotting horse model', *Int. J. Robot. Res.*, vol. 19, p. 566, 2000.
[39] H. M. Herr and T. A. McMahon, 'A galloping horse model', *Int. J. Robot. Res.*, vol. 20, p. 26, 2001.

[40] A. Formal'sky, C. Chevellereau, and B. Perrin, 'On ballistic walking locomotion of a quadruped', *Int. J. Robot. Res.*, vol. 19, pp. 743--61, 2000.
[41] I. Poulakakis, J. A. Smith, and M. Buehler, 'Modeling and Experiments of Untethered Quadrupedal Running with a Bounding Gait: The Scout II Robot', *Int. J. Robot. Res.*, vol. 24, p. 256, 2005.
[42] T.-T. Lee, C.-M. Liao, and T. K. Chen, 'On the stability properties of hexapod tripod gait', *Robot. Autom. IEEE J. Of*, vol. 4, pp. 427–434, Aug. 1988.
[43] D. J. Braun and M. Goldfarb, 'A Control Approach for Actuated Dynamic Walking in Biped Robots', *Robot. IEEE Trans. On*, vol. 25, pp. 1292–1303, Dec. 2009.
[44] S. Grillner, 'Locomotion in vertebrates: central mechanisms and reflex interaction', *Physiol. Rev.*, vol. 55, p. 247, 1975.
[45] E. Marder and D. Bucher, 'Central pattern generators and the control of rhythmic movements', *Curr. Biol.*, vol. 11, pp. R986–R996, 2001.
[46] J. Duysens and H. W. A. A. Van de Crommert, 'Neural control of locomotion; Part 1: The central pattern generator from cats to humans', *Gait Posture*, vol. 7, pp. 131--141, 1998.
[47] M. MacKay-Lyons, 'Central pattern generation of locomotion: a review of the evidence', *Phys. Ther.*, vol. 82, p. 69, 2002.
[48] N. I. Syed, A. G. Bulloch, and K. Lukowiak, 'In vitro reconstruction of the respiratory central pattern generator of the mollusk Lymnaea', *Science*, vol. 250, p. 282, 1990.
[49] K. Zhu, D. Zhang, and L. Lan, 'On Central Pattern Generator of Biological Motor System', presented at the Control, Automation, Robotics and Vision, 2006. ICARCV '06. 9th International Conference on, 2006, pp. 1–5.
[50] Q. Wu, C. Liu, J. Zhang, and Q. Chen, 'Survey of locomotion control of legged robots inspired by biological concept', *Sci. China Ser. F Inf. Sci.*, vol. 52, pp. 1715–1729, 2009.
[51] A. J. Ijspeert, 'Central pattern generators for locomotion control in animals and robots: a review', *Neural Netw.*, vol. 21, pp. 642–653, 2008.
[52] K. Matsuoka, 'Sustained oscillations generated by mutually inhibiting neurons with adaptation', *Biol. Cybern.*, vol. 52, pp. 367--376, 1985.
[53] Y. Fukuoka, H. Kimura, and A. H. Cohen, 'Adaptive dynamic walking of a quadruped robot on irregular terrain based on biological concepts', *Int. J. Robot. Res.*, vol. 22, p. 187, 2003.
[54] K. Feng, C.-M. Chew, G.-S. Hong, and T. Zielinska, 'Bipedal locomotion control using a four-compartmental central pattern generator', presented at the Mechatronics and Automation, 2005 IEEE International Conference, 2005, vol. 3, p. 1515–1520 Vol. 3.
[55] T. Komatsu and M. Usui, 'Dynamic walking and running of a bipedal robot using hybrid central pattern generator method', presented at the Mechatronics and Automation, 2005 IEEE International Conference, 2005, vol. 2, p. 987–992 Vol. 2.
[56] G. Endo, J. Morimoto, T. Matsubara, J. Nakanishi, and G. Cheng, 'Learning CPG-based Biped Locomotion with a Policy Gradient Method: Application to a Humanoid Robot', *Int. J. Robot. Res.*, vol. 27, pp. 213--228, 2008.
[57] K. Asa, K. Ishimura, and M. Wada, 'Behavior transition between biped and quadruped walking by using bifurcation', *Robot. Auton. Syst.*, vol. 57, pp. 155–160, 2009.
[58] H. Takemura, M. Deguchi, J. Ueda, Y. Matsumoto, and T. Ogasawara, 'Slip-adaptive walk of quadruped robot', *Robot. Auton. Syst.*, vol. 53, pp. 124–141, 2005.
[59] J. Shan, C. Junshi, and C. Jiapin, 'Design of central pattern generator for humanoid robot walking based on multi-objective GA', presented at the Intelligent Robots and Systems, 2000. (IROS 2000). Proceedings. 2000 IEEE/RSJ International Conference on, 2000, vol. 3, pp. 1930–1935 vol.3.
[60] J.-J. Kim and J.-J. Lee, 'Gait adaptation method of biped robot for various terrains using central pattern generator (CPG) and learning mechanism', presented at the Control,

Automation and Systems, 2007. ICCAS ’07. International Conference on, 2007, pp. 10–14.
[61] K. Wolff, J. Pettersson, A. Heralic, and M. Wahde, ‘Structural Evolution of Central Pattern Generators for Bipedal Walking in 3D Simulation’, presented at the Systems, Man and Cybernetics, 2006. SMC ’06. IEEE International Conference on, 2006, vol. 1, pp. 227–234.
[62] H. Inada and K. Ishii, ‘Behavior generation of bipedal robot using central pattern generator(CPG) (1st report: CPG parameters searching method by genetic algorithm)’, presented at the Intelligent Robots and Systems, 2003. (IROS 2003). Proceedings. 2003 IEEE/RSJ International Conference on, 2003, vol. 3, pp. 2179–2184 vol.3.
[63] Q. Lu, J. Tian, Q. Lu, and J. Tian, ‘Research on Walking Gait of Biped Robot Based on a Modified CPG Model, Research on Walking Gait of Biped Robot Based on a Modified CPG Model’, *Math. Probl. Eng. Math. Probl. Eng.*, vol. 2015, 2015, p. e793208, May 2015.
[64] C. Pribe, S. Grossberg, and M. A. Cohen, ‘Neural control of interlimb oscillations’, *Biol. Cybern.*, vol. 77, pp. 141–152, 1997.
[65] D. Micci-Barreca and H. Ogmen, ‘A central pattern generator for insect gait production’, presented at the From Perception to Action Conference, 1994., Proceedings, 1994, pp. 348–351.
[66] D. Lachat, A. Crespi, and A. J. Ijspeert, ‘BoxyBot: a swimming and crawling fish robot controlled by a central pattern generator’, presented at the Biomedical Robotics and Biomechatronics, 2006. BioRob 2006. The First IEEE/RAS-EMBS International Conference on, 2006, pp. 643–648.
[67] K. A. McIsaac and J. P. Ostrowski, ‘Experimental verification of open-loop control for an underwater eel-like robot’, *Int. J. Robot. Res.*, vol. 21, p. 849, 2002.
[68] V. Mehta, S. Brennan, and F. Gandhi, ‘Experimentally Verified Optimal Serpentine Gait and Hyperredundancy of a Rigid-Link Snake Robot’, *Robot. IEEE Trans. On*, vol. 24, pp. 348–360, Apr. 2008.
[69] K. A. McIsaac and J. P. Ostrowski, ‘Motion planning for anguilliform locomotion’, *Robot. Autom. IEEE Trans. On*, vol. 19, pp. 637–652, Aug. 2003.
[70] J. Morimoto, G. Endo, J. Nakanishi, and G. Cheng, ‘A Biologically Inspired Biped Locomotion Strategy for Humanoid Robots: Modulation of Sinusoidal Patterns by a Coupled Oscillator Model’, *Robot. IEEE Trans. On*, vol. 24, pp. 185–191, Feb. 2008.
[71] K. Watanabe, A. Tajima, and K. Izumi, ‘Locomotion pattern generation of semi-looper type robots using central pattern generators based on van der Pol oscillators’, presented at the Industrial Informatics, 2008. INDIN 2008. 6th IEEE International Conference on, 2008, pp. 377–382.
[72] J. S. Bay and H. Hemami, ‘Modeling of a Neural Pattern Generator with Coupled nonlinear Oscillators’, *Biomed. Eng. IEEE Trans. On*, vol. BME-34, pp. 297–306, Apr. 1987.
[73] C. Liu, Q. Chen, and J. Zhang, ‘Coupled Van Der Pol oscillators utilised as Central pattern generators for quadruped locomotion’, presented at the Control and Decision Conference, 2009. CCDC ’09. Chinese, 2009, pp. 3677–3682.
[74] R. Hliot and B. Espiau, ‘Online generation of cyclic leg trajectories synchronized with sensor measurement’, *Robot. Auton. Syst.*, vol. 56, pp. 410–421, 2008.
[75] S. Akio and Y. Masaki, ‘Design of a novel central pattern generator and the hebbian motion learning’, presented at the Control Applications, (CCA) \& Intelligent Control, (ISIC), 2009 IEEE, 2009, pp. 1655–1660.
[76] W. Xiao and W. Wang, ‘Hopf oscillator-based gait transition for a quadruped robot’, in *2014 IEEE International Conference on Robotics and Biomimetics (ROBIO)*, 2014, pp. 2074–2079.
[77] L. Righetti and A. J. Ijspeert, ‘Programmable central pattern generators: an application to biped locomotion control’, presented at the Robotics and Automation, 2006. ICRA 2006. Proceedings 2006 IEEE International Conference on, 2006, pp. 1585–1590.

[78] S. Rutishauser, A. Sprowitz, L. Righetti, and A. J. Ijspeert, 'Passive compliant quadruped robot using Central Pattern Generators for locomotion control', presented at the Biomedical Robotics and Biomechatronics, 2008. BioRob 2008. 2nd IEEE RAS \& EMBS International Conference on, 2008, pp. 710–715.
[79] G. CAPI, Y. NASU, and L. BAROLLI, 'Application of Genetic Algorithms for biped robot gait synthesis optimization during walking and going up-stairs', *Adv. Robot.*, vol. 15, pp. 675--694, 2001.
[80] G. Capi, Y. Nasu, L. Barolli, and K. Mitobe, 'Real time gait generation for autonomous humanoid robots: A case study for walking', *Robot. Auton. Syst.*, vol. 42, pp. 107–116, 2003.
[81] W. Ilg and K. Berns, 'A learning architecture based on reinforcement learning for adaptive control of the walking machine LAURON', *Robot. Auton. Syst.*, vol. 15, pp. 321–334, 1995.
[82] H. Benbrahim and J. A. Franklin, 'Biped dynamic walking using reinforcement learning', *Robot. Auton. Syst.*, vol. 22, pp. 283–302, 1997.
[83] J. S. Albus, 'A New Approach to Manipulator Control: The Cerebellar Model Articulation Controller (CMAC)', *J. Dyn. Syst. Meas. Control*, vol. 97, pp. 220–227, 1975.
[84] S. Srinivasan, R. E. Gander, and H. C. Wood, 'A movement pattern generator model using artificial neural networks', *Biomed. Eng. IEEE Trans. On*, vol. 39, pp. 716–722, Jul. 1992.
[85] K. Berns, R. Dillmann, and S. Piekenbrock, 'Neural networks for the control of a six-legged walking machine', *Robot. Auton. Syst.*, vol. 14, pp. 233–244, 1995.
[86] T. Reil and P. Husbands, 'Evolution of central pattern generators for bipedal walking in a real-time physics environment', *Evol. Comput. IEEE Trans. On*, vol. 6, pp. 159–168, Apr. 2002.
[87] J. C. Gallagher, R. D. Beer, K. S. Espenschied, and R. D. Quinn, 'Application of evolved locomotion controllers to a hexapod robot', *Robot. Auton. Syst.*, vol. 19, pp. 95–103, 1996.
[88] R. D. Beer and J. C. Gallagher, 'Evolving dynamical neural networks for adaptive behavior', *Adapt. Behav.*, vol. 1, pp. 91--122, 1992.
[89] F. Wyffels and B. Schrauwen, 'Design of a Central Pattern Generator Using Reservoir Computing for Learning Human Motion', presented at the Advanced Technologies for Enhanced Quality of Life, 2009. AT-EQUAL '09., 2009, pp. 118–122.
[90] T. D. Barfoot, E. J. P. Earon, and G. M. T. D'Eleuterio, 'Experiments in learning distributed control for a hexapod robot', *Robot. Auton. Syst.*, vol. 54, pp. 864–872, 2006.
[91] P. K. Pal and D. C. Kar, 'Gait optimization through search', *Int. J. Robot. Res.*, vol. 19, p. 394, 2000.
[92] P. R. Vundavilli and D. K. Pratihar, 'Dynamically balanced optimal gaits of a ditch-crossing biped robot', *Robot. Auton. Syst.*, vol. 58, pp. 349–361, 2010.
[93] D. K. Pratihar, K. Deb, and A. Ghosh, 'Optimal path and gait generations simultaneously of a six-legged robot using a GA-fuzzy approach', *Robot. Auton. Syst.*, vol. 41, pp. 1–20, 2002.
[94] R. K. Jha, B. Singh, and D. K. Pratihar, 'On-line stable gait generation of a two-legged robot using a genetic-fuzzy system', *Robot. Auton. Syst.*, vol. 53, pp. 15–35, 2005.
[95] C. Zhou and D. Ruan, 'Integration of linguistic and numerical information for biped control', *Robot. Auton. Syst.*, vol. 28, pp. 53–70, 1999.
[96] C. Zhou and Q. Meng, 'Dynamic balance of a biped robot using fuzzy reinforcement learning agents', *Fuzzy Sets Syst.*, vol. 134, pp. 169–187, 2003.
[97] C. Zhou, 'Robot learning with GA-based fuzzy reinforcement learning agents', *Inf. Sci.*, vol. 145, pp. 45–68, 2002.
[98] T. Inamura, I. Toshima, H. Tanie, and Y. Nakamura, 'Embodied symbol emergence based on mimesis theory', *Int. J. Robot. Res.*, vol. 23, p. 363, 2004.

[99] D. Lee and Y. Nakamura, 'Mimesis Model from Partial Observations for a Humanoid Robot', *Int. J. Robot. Res.*, vol. 29, p. 60, 2010.
[100] D. Kulic and Y. Nakamura, 'Incremental Learning and Memory Consolidation of Whole Body Human Motion Primitives', *Adapt. Behav.*, vol. 17, p. 484, 2009.
[101] D. Kulic, W. Takano, and Y. Nakamura, 'Incremental Learning, Clustering and Hierarchy Formation of Whole Body Motion Patterns using Adaptive Hidden Markov Chains', *Int. J. Robot. Res.*, vol. 27, pp. 761--784, 2008.
[102] L. E. Baum, T. Petrie, G. Soules, and N. Weiss, 'A Maximization Technique Occurring in the Statistical Analysis of Probabilistic Functions of Markov Chains', *Ann. Math. Stat.*, vol. 41, pp. 164–171, Feb. 1970.
[103] A. Viterbi, 'Error bounds for convolutional codes and an asymptotically optimum decoding algorithm', *Inf. Theory IEEE Trans. On*, vol. 13, pp. 260–269, Apr. 1967.
[104] A. Russell, G. Orchard, and R. Etienne-Cummings, 'Configuring of Spiking Central Pattern Generator Networks for Bipedal Walking Using Genetic Algorthms', presented at the Circuits and Systems, 2007. ISCAS 2007. IEEE International Symposium on, 2007, pp. 1525–1528.
[105] E. R. Westervelt, J. W. Grizzle, and D. E. Koditschek, 'Hybrid zero dynamics of planar biped walkers', *Autom. Control IEEE Trans. On*, vol. 48, pp. 42–56, Jan. 2003.
[106] E. R. Westervelt, G. Buche, and J. W. Grizzle, 'Experimental validation of a framework for the design of controllers that induce stable walking in planar bipeds', *Int. J. Robot. Res.*, vol. 23, p. 559, 2004.
[107] S. Miossec and Y. Aoustin, 'A Simplified Stability Study for a Biped Walk with Underactuated and Overactuated Phases', *Int. J. Robot. Res.*, vol. 24, p. 551, 2005.
[108] G. Capi, S. Kaneko, K. Mitobe, L. Barolli, and Y. Nasu, 'Optimal trajectory generation for a prismatic joint biped robot using genetic algorithms', *Robot. Auton. Syst.*, vol. 38, pp. 119–128, 2002.
[109] G. Bessonnet, P. Seguin, and P. Sardain, 'A Parametric Optimization Approach to Walking Pattern Synthesis', *Int. J. Robot. Res.*, vol. 24, p. 536, 2005.
[110] C. Chevallereau, J. W. Grizzle, and C.-L. Shih, 'Asymptotically Stable Walking of a Five-Link Underactuated 3-D Bipedal Robot', *Robot. IEEE Trans. On*, vol. 25, pp. 37–50, Feb. 2009.
[111] T. Hemker, M. Stelzer, O. von Stryk, and H. Sakamoto, 'Efficient Walking Speed Optimization of a Humanoid Robot', *Int. J. Robot. Res.*, vol. 28, pp. 303–314, 2009.
[112] T. Geng, B. Porr, and rg\"o W\"o, 'Fast Biped Walking with a Sensor-driven Neuronal Controller and Real-time Online Learning', *Int. J. Robot. Res.*, vol. 25, p. 259, 2006.
[113] G. S. Hornby, S. Takamura, T. Yamamoto, and M. Fujita, 'Autonomous evolution of dynamic gaits with two quadruped robots', *Robot. IEEE Trans. On*, vol. 21, pp. 402–410, Jun. 2005.
[114] M. Sznaier and M. J. Damborg, 'An adaptive controller for a one-legged mobile robot', *Robot. Autom. IEEE Trans. On*, vol. 5, pp. 253–259, Apr. 1989.
[115] H. Yuasa and M. Ito, 'A Theory on Autonomous Distributed Systems with Application to a Gait Pattern Generator of Quadruped', presented at the American Control Conference, 1991, 1991, pp. 2268–2273.
[116] L. Righetti and A. J. Ijspeert, 'Pattern generators with sensory feedback for the control of quadruped locomotion', presented at the Robotics and Automation, 2008. ICRA 2008. IEEE International Conference on, 2008, pp. 819–824.
[117] P. Arena, L. Fortuna, M. Frasca, and L. Patane, 'CNN based central pattern generators with sensory feedback', presented at the Cellular Neural Networks and Their Applications, 2002. (CNNA 2002). Proceedings of the 2002 7th IEEE International Workshop on, 2002, pp. 275–282.

[118] W. Zhao, J. Yu, Y. Fang, and L. Wang, 'Development of Multi-mode Biomimetic Robotic Fish Based on Central Pattern Generator', presented at the Intelligent Robots and Systems, 2006 IEEE/RSJ International Conference on, 2006, pp. 3891–3896.

[119] D. Zhang, D. Hu, L. Shen, and H. Xie, 'Design of a Central Pattern Generator for Bionic-robot Joint with Angular Frequency Modulation', presented at the Robotics and Biomimetics, 2006. ROBIO '06. IEEE International Conference on, 2006, pp. 1664–1669.

[120] I. Tanev, T. Ray, and A. Buller, 'Automated Evolutionary Design, Robustness, and Adaptation of Sidewinding Locomotion of a Simulated Snake-Like Robot', *Robot. IEEE Trans. On*, vol. 21, pp. 632–645, Aug. 2005.

[121] Y. Geva and A. Shapiro, 'A combined potential function and graph search approach for free gait generation of quadruped robots', presented at the Robotics and Automation (ICRA), 2012 IEEE International Conference on, 2012, pp. 5371–5376.

[122] M. Reyes-Sierra and C. C. Coello, 'Multi-objective particle swarm optimizers: A survey of the state-of-the-art', *Int. J. Comput. Intell. Res.*, vol. 2, no. 3, pp. 287–308, 2006.

[123] M. Zambrano-Bigiarini, M. Clerc, and R. Rojas, 'Standard Particle Swarm Optimisation 2011 at CEC-2013: A baseline for future PSO improvements', in *2013 IEEE Congress on Evolutionary Computation (CEC)*, 2013, pp. 2337–2344.

[124] M. Srinivas and L. M. Patnaik, 'Genetic algorithms: a survey', *Computer*, vol. 27, no. 6, pp. 17–26, Jun. 1994.

[125] S. M. Elsayed, R. A. Sarker, and D. L. Essam, 'A genetic algorithm for solving the CEC'2013 competition problems on real-parameter optimization', in *2013 IEEE Congress on Evolutionary Computation (CEC)*, 2013, pp. 356–360.

[126] S. Das and P. N. Suganthan, 'Differential Evolution: A Survey of the State-of-the-Art', *IEEE Trans. Evol. Comput.*, vol. 15, no. 1, pp. 4–31, Feb. 2011.

[127] J. Tvrdik and R. Polakova, 'Competitive differential evolution applied to CEC 2013 problems', in *2013 IEEE Congress on Evolutionary Computation (CEC)*, 2013, pp. 1651–1657.

[128] D. Goldberg and J. Holland, 'Genetic Algorithms and Machine Learning', *Mach. Learn.*, vol. 3, pp. 95–99, Oct. 1988.

[129] R. Eberhart and J. Kennedy, 'A new optimizer using particle swarm theory', presented at the Micro Machine and Human Science, 1995. MHS '95., Proceedings of the Sixth International Symposium on, 1995, pp. 39–43.

[130] R. C. Eberhart and Y. Shi, 'Comparing inertia weights and constriction factors in particle swarm optimization', in *Proceedings of the 2000 Congress on Evolutionary Computation, 2000*, 2000, vol. 1, pp. 84–88 vol.1.

[131] S. Baluja, 'Population-based incremental learning. a method for integrating genetic search based function optimization and competitive learning', DTIC Document, 1994.

[132] M. D. Platel, S. Schliebs, and N. Kasabov, 'Quantum-inspired evolutionary algorithm: a multimodel EDA', *Evol. Comput. IEEE Trans. On*, vol. 13, pp. 1218–1232, 2009.

[133] Y. Guan, E. S. Neo, K. Yokoi, and K. Tanie, 'Stepping over obstacles with humanoid robots', *Robot. IEEE Trans. On*, vol. 22, pp. 958–973, Oct. 2006.

[134] M. K. Chakouch, P. Pouletaut, F. Charleux, and S. F. Bensamoun, 'Viscoelastic shear properties of in vivo thigh muscles measured by MR elastography', *J. Magn. Reson. Imaging*, vol. 43, no. 6, pp. 1423–1433, Jun. 2016.

[135] K.-H. Han and J.-H. Kim, 'Quantum-inspired evolutionary algorithm for a class of combinatorial optimization', *Evol. Comput. IEEE Trans. On*, vol. 6, pp. 580–593, 2002.

[136] G. K. Venayagamoorthy and G. Singhal, 'Quantum-Inspired Evolutionary Algorithms and Binary Particle Swarm Optimization for Training MLP and SRN Neural Networks', *J. Comput. Theor. Nanosci.*, vol. 2, no. 4, pp. 561–568, Dec. 2005.

[137] J. Liu, H. Wang, Y. Sun, C. Fu, and J. Guo, 'Real-Coded Quantum-Inspired Genetic Algorithm-Based BP Neural Network Algorithm', *Math. Probl. Eng.*, vol. 2015, p. e571295, Jan. 2015.

[138] H. Xing, Y. Ji, L. Bai, X. Liu, Z. Qu, and X. Wang, 'An adaptive-evolution-based quantum-inspired evolutionary algorithm for QoS multicasting in IP/DWDM networks', *Comput. Commun.*, vol. 32, no. 6, pp. 1086–1094, Apr. 2009.

[139] G. R. Harik, F. G. Lobo, and D. E. Goldberg, 'The compact genetic algorithm', *IEEE Trans. Evol. Comput.*, vol. 3, no. 4, pp. 287–297, Nov. 1999.

[140] Q. Zhang, 'On stability of fixed points of limit models of univariate marginal distribution algorithm and factorized distribution algorithm', *IEEE Trans. Evol. Comput.*, vol. 8, no. 1, pp. 80–93, Feb. 2004.

[141] K.-H. Han and J.-H. Kim, 'Quantum-inspired evolutionary algorithms with a new termination criterion, Hε gate, and two-phase scheme', *IEEE Trans. Evol. Comput.*, vol. 8, no. 2, pp. 156–169, Apr. 2004.

[142] G. Zhang and H. Rong, 'Real-Observation Quantum-Inspired Evolutionary Algorithm for a Class of Numerical Optimization Problems', in *Computational Science – ICCS 2007*, Y. Shi, G. D. van Albada, J. Dongarra, and P. M. A. Sloot, Eds. Springer Berlin Heidelberg, 2007, pp. 989–996.

[143] Q. Chaoyong, L. Yongjuan, and Z. Jianguo, 'A real-coded quantum-inspired evolutionary algorithm for global numerical optimization', presented at the Cybernetics and Intelligent Systems, 2008 IEEE Conference on, 2008, pp. 1160–1164.

[144] Z. Tu and Y. Lu, 'Corrections to "A Robust Stochastic Genetic Algorithm (StGA) for Global Numerical Optimization"', *Evol. Comput. IEEE Trans. On*, vol. 12, pp. 781–781, 2008.

[145] G. S. Babu, D. B. Das, and C. Patvardhan, 'Real-parameter quantum evolutionary algorithm for economic load dispatch', *Gener. Transm. Distrib. IET*, vol. 2, pp. 22–31, 2008.

[146] M. A. Hossain, M. K. Hossain, and M. M. A. Hashem, 'A Generalized Hybrid Real-Coded Quantum Evolutionary Algorithm Based on Particle Swarm Theory with Arithmetic Crossover', *Int. J. Comput. Sci. Inf. Technol.*, vol. 2, no. 4, pp. 172–187, Aug. 2010.

[147] J. Xiao, J. Xu, Z. Chen, K. Zhang, and L. Pan, 'A hybrid quantum chaotic swarm evolutionary algorithm for DNA encoding', *Comput. Math. Appl.*, vol. 57, no. 11–12, pp. 1949–1958, Jun. 2009.

[148] Z. Tu and Y. Lu, 'Corrections to "A Robust Stochastic Genetic Algorithm (StGA) for Global Numerical Optimization"', *Evol. Comput. IEEE Trans. On*, vol. 12, pp. 781–781, 2008.

[149] G. Zhang, 'Quantum-inspired evolutionary algorithms: a survey and empirical study', *J. Heuristics*, vol. 17, pp. 303–351, 2011.

[150] R. Zhang and H. Gao, 'Real-coded Quantum Evolutionary Algorithm for Complex Functions with High-dimension', in *International Conference on Mechatronics and Automation, 2007. ICMA 2007*, 2007, pp. 2974–2979.

[151] J. Cheng, G. Zhang, F. Caraffini, and F. Neri, 'Multicriteria adaptive differential evolution for global numerical optimization', *Integr. Comput.-Aided Eng.*, vol. 22, no. 2, pp. 103–107, Apr. 2015.

[152] M. G. Epitropakis, D. K. Tasoulis, N. G. Pavlidis, V. P. Plagianakos, and M. N. Vrahatis, 'Enhancing Differential Evolution Utilizing Proximity-Based Mutation Operators', *IEEE Trans. Evol. Comput.*, vol. 15, no. 1, pp. 99–119, Feb. 2011.

[153] R. Tanabe and A. Fukunaga, 'Evaluating the performance of SHADE on CEC 2013 benchmark problems', in *2013 IEEE Congress on Evolutionary Computation (CEC)*, 2013, pp. 1952–1959.

[154] A. Zamuda, J. Brest, and E. Mezura-Montes, 'Structured Population Size Reduction Differential Evolution with Multiple Mutation Strategies on CEC 2013 real parameter optimization', in *2013 IEEE Congress on Evolutionary Computation (CEC)*, 2013, pp. 1925–1931.

[155] J. L. Rueda and I. Erlich, 'Hybrid Mean-Variance Mapping Optimization for solving the IEEE-CEC 2013 competition problems', in *2013 IEEE Congress on Evolutionary Computation (CEC)*, 2013, pp. 1664–1671.

[156] J. L. F. Martínez and E. G. Gonzalo, 'The Generalized PSO: A New Door to PSO Evolution', *J Artif Evol App*, vol. 2008, p. 5:1–5:15, Jan. 2008.

[157] V. K. Koumousis and C. P. Katsaras, 'A saw-tooth genetic algorithm combining the effects of variable population size and reinitialization to enhance performance', *IEEE Trans. Evol. Comput.*, vol. 10, no. 1, pp. 19–28, Feb. 2006.

[158] H.-B. Duan, C.-F. Xu, and Z.-H. Xing, 'A hybrid artificial bee colony optimization and quantum evolutionary algorithm for continuous optimization problems', *Int. J. Neural Syst.*, vol. 20, no. 1, pp. 39–50, Feb. 2010.

[159] H. K. Singh and T. Ray, 'Performance of a hybrid EA-DE-memetic algorithm on CEC 2011 real world optimization problems', in *2011 IEEE Congress on Evolutionary Computation (CEC)*, 2011, pp. 1322–1326.

[160] M. Asafuddoula, T. Ray, and R. Sarker, 'An adaptive differential evolution algorithm and its performance on real world optimization problems', in *2011 IEEE Congress on Evolutionary Computation (CEC)*, 2011, pp. 1057–1062.

[161] N. Mladenović, J. Petrović, V. Kovačević-Vujčić, and M. Čangalović, 'Solving spread spectrum radar polyphase code design problem by tabu search and variable neighbourhood search', *Eur. J. Oper. Res.*, vol. 151, no. 2, pp. 389–399, Dec. 2003.

[162] T. Finni, P. V. Komi, and V. Lepola, 'In vivo human triceps surae and quadriceps femoris muscle function in a squat jump and counter movement jump', *Eur. J. Appl. Physiol.*, vol. 83, pp. 416–426, Nov. 2000.

[163] P. V. Komi, 'Stretch-shortening cycle: a powerful model to study normal and fatigued muscle', *J. Biomech.*, vol. 33, no. 10, pp. 1197–1206, Oct. 2000.

[164] A. Kiam Heong, G. Chong, and L. Yun, 'PID control system analysis, design, and technology', *Control Syst. Technol. IEEE Trans. On*, vol. 13, pp. 559–576, 2005.

[165] D. Tan and Z. Chen, 'On a general formula of fourth order Runge–Kutta method', *J Math Sci Math Educ*, vol. 7, pp. 1–10, 2012.

Appendix A Additional QIEA Results

This appendix presents additional data produced in the QIEA development (chapter 4), providing results for 10 and 30 dimension versions fo the fitness functions.

Table 26: Summary statistics for the 13 traditional test functions with 10 dimensions. Bold are best.

Traditional test functions 10 Dimensions	***bQIEA***						***rQIEA***					
	Classic			***HSB***			***RCQIEA***			***SRQEA***		
Function	**Min**	**Mean**	**Std dev**	**Min**	**Mean**	**Std dev**	**Min**	**Mean**	**Std dev**	**Min**	**Mean**	**Std dev**
01 Sphere	3.08E+00	4.70E+01	3.55E+01	2.33E+00	1.48E+01	9.74E+00	1.94E-05	1.02E-04	6.11E-05	**0.00E+00**	**0.00E+00**	**0.00E+00**
02 Schwefel 222	1.97E+00	8.83E+00	3.62E+00	1.36E+00	5.79E+00	2.15E+00	6.98E-03	2.05E-02	5.46E-03	**0.00E+00**	**0.00E+00**	**0.00E+00**
03 Schwefel 12	7.70E+01	4.45E+02	3.12E+02	9.39E+00	1.16E+02	7.02E+01	2.79E-04	2.71E-03	2.74E-03	**0.00E+00**	**0.00E+00**	**0.00E+00**
04 Schwefel 221	3.78E+00	1.03E+01	2.83E+00	2.75E+00	6.70E+00	2.22E+00	1.51E-02	3.26E-02	1.09E-02	**1.14E-05**	**4.67E-05**	**2.43E-05**
05 Rosenbrock	7.39E+02	2.10E+05	4.98E+05	6.51E+02	6.52E+03	5.00E+03	3.25E-01	2.19E+01	3.26E+01	**3.61E-03**	**1.98E+01**	**2.72E+01**
06 Step	6.00E+00	6.73E+01	5.94E+01	1.00E+00	2.07E+01	1.51E+01	**0.00E+00**	**0.00E+00**	**0.00E+00**	**0.00E+00**	**0.00E+00**	**0.00E+00**
07 Quartic	1.40E+02	2.73E+04	3.37E+04	2.99E+01	1.77E+03	3.47E+03	**0.00E+00**	**0.00E+00**	**0.00E+00**	**0.00E+00**	**0.00E+00**	**0.00E+00**
08 Schwefel 226	7.33E-01	6.27E+00	5.60E+00	1.43E-01	1.94E+00	1.55E+00	2.76E-06	1.26E-05	8.59E-06	**0.00E+00**	**0.00E+00**	**0.00E+00**
09 Basic Rastrigin	5.13E+00	9.51E+00	1.90E+00	3.68E+00	7.22E+00	1.69E+00	7.96E-06	6.38E-05	4.83E-05	**0.00E+00**	**0.00E+00**	**0.00E+00**
10 Basic Ackley	8.44E+00	1.77E+01	1.59E+00	5.17E+00	1.49E+01	3.51E+00	3.75E-03	1.35E-02	4.59E-03	**0.00E+00**	**0.00E+00**	**0.00E+00**
11 Basic Griewank	4.71E-01	1.46E+00	5.70E-01	4.23E-01	9.03E-01	2.42E-01	3.11E-04	3.87E-02	2.27E-02	**0.00E+00**	**1.09E-02**	**1.02E-02**
12 Penalised 1	2.09E-01	2.60E+02	7.28E+02	3.97E-01	1.26E+01	6.53E+00	**0.00E+00**	**0.00E+00**	**0.00E+00**	**0.00E+00**	**0.00E+00**	**0.00E+00**
13 Penalised 2	1.99E+00	5.38E+04	8.75E+04	3.65E-01	2.58E+03	1.83E+04	**0.00E+00**	**0.00E+00**	**0.00E+00**	**0.00E+00**	**0.00E+00**	**0.00E+00**

Table 27: Summary statistics for the 13 traditional test functions with 30 dimensions. Bold are best.

Traditional test functions 30 Dimensions	***bQIEA***						***rQIEA***					
	Classic			***HSB***			***RCQIEA***			***SRQEA***		
Function	**Min**	**Mean**	**Std dev**	**Min**	**Mean**	**Std dev**	**Min**	**Mean**	**Std dev**	**Min**	**Mean**	**Std dev**
01 Sphere	4.03E+02	1.27E+03	3.82E+02	2.35E+02	5.33E+02	1.52E+02	9.05E-05	3.38E-04	1.48E-04	**0.00E+00**	**0.00E+00**	**0.00E+00**
02 Schwefel 222	8.18E+01	1.08E+02	1.21E+01	3.14E+01	5.54E+01	1.03E+01	4.00E-02	6.24E-02	1.12E-02	**0.00E+00**	**0.00E+00**	**0.00E+00**
03 Schwefel 12	9.50E+04	2.05E+05	5.15E+04	3.02E+04	6.42E+04	2.08E+04	2.09E-02	7.79E-02	4.29E-02	**0.00E+00**	**0.00E+00**	**0.00E+00**
04 Schwefel 221	2.22E+01	3.18E+01	3.80E+00	1.94E+01	2.44E+01	2.09E+00	8.27E-02	1.43E-01	3.05E-02	**3.51E-03**	**6.16E-03**	**1.56E-03**
05 Rosenbrock	2.02E+07	4.04E+07	1.40E+07	3.03E+05	6.65E+06	5.30E+06	1.65E+01	1.16E+02	**5.33E+01**	**1.04E-02**	**8.86E+01**	1.80E+02
06 Step	5.25E+02	1.34E+03	3.65E+02	1.93E+02	5.91E+02	1.95E+02	**0.00E+00**	**0.00E+00**	**0.00E+00**	**0.00E+00**	**0.00E+00**	**0.00E+00**
07 Quartic	1.50E+06	5.15E+06	2.09E+06	1.85E+05	9.82E+05	6.14E+05	**0.00E+00**	**0.00E+00**	**0.00E+00**	**0.00E+00**	**0.00E+00**	**0.00E+00**
08 Schwefel 226	7.50E+01	1.49E+02	3.64E+01	1.11E+01	5.90E+01	2.10E+01	1.15E-05	4.03E-05	1.46E-05	**0.00E+00**	**0.00E+00**	**0.00E+00**
09 Basic Rastrigin	4.61E+01	5.86E+01	5.32E+00	3.26E+01	4.15E+01	4.57E+00	8.00E-05	1.65E-04	6.63E-05	**0.00E+00**	**0.00E+00**	**0.00E+00**
10 Basic Ackley	1.99E+01	2.00E+01	2.15E-02	1.84E+01	1.95E+01	3.28E-01	8.80E-03	**1.61E-02**	**4.55E-03**	**0.00E+00**	9.20E-01	4.01E+00
11 Basic Griewank	5.97E+00	1.33E+01	3.07E+00	1.79E+00	5.68E+00	1.42E+00	2.32E-04	**1.30E-02**	**1.39E-02**	**0.00E+00**	2.06E-02	2.25E-02
12 Penalised 1	8.03E+04	2.03E+06	1.47E+06	8.37E+01	1.51E+05	3.28E+05	**0.00E+00**	**0.00E+00**	**0.00E+00**	**0.00E+00**	**0.00E+00**	**0.00E+00**
13 Penalised 2	2.55E+05	1.35E+07	6.56E+06	7.33E+03	1.75E+06	1.72E+06	**0.00E+00**	**0.00E+00**	**0.00E+00**	**0.00E+00**	**0.00E+00**	**0.00E+00**

Table 28: Summary statistics for the 20 CEC-2013 test functions with 10 dimensions. Bold are best.

CEC-2013 test functions 10 Dimensions	*bQIEA*						*rQIEA*					
	Classic			*HSB*			*RCQIEA*			*SRQEA*		
Function	**Min**	**Mean**	**Std dev**	**Min**	**Mean**	**Std dev**	**Min**	**Mean**	**Std dev**	**Min**	**Mean**	**Std dev**
14 Sphere [duplicated]	3.08E+00	4.70E+01	3.55E+01	2.33E+00	1.48E+01	9.74E+00	1.94E-05	1.02E-04	6.11E-05	**0.00E+00**	**0.00E+00**	**0.00E+00**
15 R HC elliptic	3.87E+05	2.17E+06	1.12E+06	2.21E+05	1.38E+06	8.52E+05	3.97E+05	2.13E+06	1.24E+06	**4.61E+04**	**5.22E+05**	**4.18E+05**
16 Rotated bent cigar	6.70E+06	1.72E+08	1.09E+08	1.47E+06	4.95E+07	3.11E+07	1.39E+03	5.27E+03	**1.80E+03**	**4.59E+01**	**3.48E+03**	2.00E+03
17 Rotated discus	6.79E+03	1.44E+04	**3.82E+03**	**4.34E+03**	**1.24E+04**	4.51E+03	9.97E+03	3.14E+04	1.04E+04	7.17E+03	3.09E+04	1.19E+04
18 Different powers	4.15E+00	3.39E+01	1.38E+01	2.86E+00	1.06E+01	6.65E+00	1.96E-04	2.13E-03	1.72E-03	**0.00E+00**	**0.00E+00**	**0.00E+00**
19 Rotated Rosenbrock	8.12E+00	1.55E+01	4.34E+00	1.89E+00	1.00E+01	**4.00E+00**	2.16E-03	4.69E+00	4.66E+00	**5.59E-04**	**2.76E+00**	4.01E+00
20 Rotated Schaffers F7	2.36E+01	4.17E+01	9.59E+00	**1.77E+01**	**3.40E+01**	**6.81E+00**	2.13E+01	7.13E+01	3.23E+01	3.31E+01	8.88E+01	3.64E+01
21 Rotated Ackley	2.02E+01	2.04E+01	8.11E-02	2.01E+01	2.03E+01	8.29E-02	2.02E+01	2.04E+01	**6.97E-02**	**2.01E+01**	**2.03E+01**	8.04E-02
22 Rotated Weierstrass	3.32E+00	5.05E+00	6.45E-01	**2.68E+00**	**4.20E+00**	**5.51E-01**	3.00E+00	6.02E+00	1.43E+00	3.05E+00	7.26E+00	1.38E+00
23 Rotated Griewank	4.82E+00	2.85E+01	1.21E+01	3.74E+00	1.96E+01	7.69E+00	**4.59E-01**	**1.52E+00**	**6.96E-01**	7.03E-01	2.87E+00	1.67E+00
24 Rastrigin	3.03E+00	1.07E+01	2.58E+00	3.97E+00	7.95E+00	1.88E+00	3.11E-05	2.09E-04	1.39E-04	**0.00E+00**	**0.00E+00**	**0.00E+00**
25 Rotated Rastrigin	1.08E+01	2.61E+01	6.00E+00	**8.19E+00**	**2.18E+01**	**5.71E+00**	1.80E+01	3.55E+01	1.30E+01	1.49E+01	4.48E+01	1.62E+01
26 NC rotated Rastrigin	2.15E+01	3.71E+01	7.88E+00	1.66E+01	**3.02E+01**	**7.59E+00**	1.85E+01	4.67E+01	1.47E+01	**1.64E+01**	5.21E+01	1.46E+01
27 Schwefel 7	2.70E+01	7.86E+01	2.99E+01	1.19E+01	5.77E+01	1.75E+01	1.03E-03	**7.30E-03**	**6.87E-03**	**0.00E+00**	6.66E-02	1.79E-01
28 Rotated Schwefel 7	3.34E+02	5.88E+02	1.40E+02	2.21E+02	**4.90E+02**	**1.28E+02**	**1.32E+02**	6.86E+02	2.20E+02	4.01E+02	8.14E+02	2.12E+02
29 Rotated Katsuura	2.15E-01	4.70E-01	1.22E-01	**1.50E-01**	**4.13E-01**	**1.07E-01**	1.94E-01	4.88E-01	1.68E-01	2.20E-01	5.95E-01	2.11E-01
30 Lunacek bi-Rastrigin	1.12E+01	2.12E+01	5.70E+00	6.79E+00	1.39E+01	3.90E+00	2.95E-03	1.99E-02	1.65E-02	**0.00E+00**	**0.00E+00**	**0.00E+00**
31 R Lunacek bi-Rastrigin	2.56E+01	5.27E+01	9.40E+00	1.76E+01	4.34E+01	**8.59E+00**	**1.74E+01**	**3.72E+01**	9.29E+00	2.17E+01	4.27E+01	1.02E+01
32 RE Griewank Rosen.	**4.35E-01**	2.40E+00	9.78E-01	9.33E-01	**1.91E+00**	**4.47E-01**	8.77E-01	3.85E+00	1.88E+00	8.65E-01	5.92E+00	3.54E+00
33 RE Schaffers F6	1.33E+00	1.92E+00	2.90E-01	1.09E+00	**1.71E+00**	**2.79E-01**	1.24E+00	2.22E+00	4.46E-01	**1.00E+00**	2.30E+00	5.94E-01

Table 29: Summary statistics for the 20 CEC-2013 test functions with 30 dimensions. Bold are best.

CEC-2013 test functions 30 Dimensions	*bQIEA*						*rQIEA*					
	Classic			*HSB*			*RCQIEA*			*SRQEA*		
Function	**Min**	**Mean**	**Std dev**	**Min**	**Mean**	**Std dev**	**Min**	**Mean**	**Std dev**	**Min**	**Mean**	**Std dev**
14 Sphere [duplicated]	4.03E+02	1.27E+03	3.82E+02	2.35E+02	5.33E+02	1.52E+02	9.05E-05	3.38E-04	1.48E-04	**0.00E+00**	**0.00E+00**	**0.00E+00**
15 R HC elliptic	2.52E+07	4.90E+07	1.13E+07	1.26E+07	3.14E+07	7.10E+06	2.72E+06	7.04E+06	2.23E+06	**8.12E+05**	**1.78E+06**	**6.45E+05**
16 Rotated bent cigar	1.19E+10	1.94E+10	4.16E+09	3.41E+09	7.47E+09	1.63E+09	2.40E+02	5.01E+03	8.83E+03	**1.27E-02**	**8.81E+01**	**2.63E+02**
17 Rotated discus	**4.82E+04**	8.84E+04	1.37E+04	4.98E+04	**7.60E+04**	**1.10E+04**	8.14E+04	1.20E+05	2.05E+04	7.49E+04	1.19E+05	2.06E+04
18 Different powers	2.79E+02	6.66E+02	2.01E+02	8.30E+01	1.97E+02	6.44E+01	9.81E-04	3.19E-03	2.21E-03	**0.00E+00**	**0.00E+00**	**0.00E+00**
19 Rotated Rosenbrock	1.42E+02	1.80E+02	1.58E+01	7.42E+01	1.17E+02	1.52E+01	1.98E-01	2.17E+01	1.24E+01	**6.54E-02**	**1.43E+01**	**7.99E+00**
20 Rotated Schaffers F7	1.09E+02	1.49E+02	1.98E+01	**1.07E+02**	**1.31E+02**	**1.22E+01**	1.12E+02	1.79E+02	4.02E+01	1.32E+02	1.79E+02	3.00E+01
21 Rotated Ackley	2.08E+01	2.10E+01	6.04E-02	2.09E+01	2.10E+01	**4.68E-02**	2.07E+01	2.09E+01	5.74E-02	**2.07E+01**	**2.09E+01**	7.01E-02
22 Rotated Weierstrass	2.44E+01	2.85E+01	**1.35E+00**	**1.83E+01**	**2.57E+01**	1.69E+00	2.57E+01	3.27E+01	3.03E+00	2.87E+01	3.50E+01	2.42E+00
23 Rotated Griewank	3.24E+02	5.77E+02	1.27E+02	1.36E+02	3.15E+02	8.79E+01	1.33E+00	1.90E+00	2.94E-01	**4.19E-02**	**2.11E-01**	**7.20E-02**
24 Rastrigin	6.82E+01	8.84E+01	9.08E+00	4.13E+01	5.25E+01	5.75E+00	2.12E-04	5.58E-04	2.85E-04	**0.00E+00**	**0.00E+00**	**0.00E+00**
25 Rotated Rastrigin	1.82E+02	2.27E+02	2.26E+01	1.31E+02	**1.74E+02**	**2.01E+01**	**1.20E+02**	2.55E+02	7.84E+01	1.50E+02	2.73E+02	6.74E+01
26 NC rotated Rastrigin	2.26E+02	2.83E+02	2.41E+01	1.78E+02	**2.38E+02**	**2.37E+01**	**1.35E+02**	2.76E+02	5.10E+01	2.04E+02	2.97E+02	5.15E+01
27 Schwefel 7	3.89E+02	8.37E+02	1.47E+02	1.71E+02	3.41E+02	8.30E+01	**8.81E-03**	**4.12E-02**	**2.33E-02**	2.72E-02	4.40E-01	2.56E-01
28 Rotated Schwefel 7	2.81E+03	3.79E+03	**3.48E+02**	**2.17E+03**	**3.18E+03**	3.58E+02	2.17E+03	3.34E+03	5.00E+02	2.48E+03	3.46E+03	4.66E+02
29 Rotated Katsuura	6.84E-01	1.04E+00	1.63E-01	6.96E-01	**9.40E-01**	**1.35E-01**	**4.48E-01**	1.10E+00	3.28E-01	6.13E-01	1.25E+00	3.28E-01
30 Lunacek bi-Rastrigin	9.90E+01	1.58E+02	2.09E+01	5.95E+01	9.06E+01	1.32E+01	1.57E-02	5.53E-02	2.60E-02	**0.00E+00**	**4.77E-04**	**2.30E-03**
31 R Lunacek bi-Rastrigin	3.07E+02	4.11E+02	3.63E+01	2.36E+02	3.11E+02	**3.35E+01**	**1.15E+02**	**2.19E+02**	4.49E+01	1.50E+02	2.45E+02	5.13E+01
32 RE Griewank Rosen.	2.01E+01	1.29E+02	6.90E+01	**1.39E+01**	**5.15E+01**	1.81E+01	3.26E+01	5.78E+01	**1.79E+01**	4.89E+01	1.30E+02	4.51E+01
33 RE Schaffers F6	9.90E+00	1.12E+01	**5.69E-01**	**8.07E+00**	**9.65E+00**	6.20E-01	9.44E+00	1.33E+01	1.59E+00	9.16E+00	1.38E+01	1.36E+00

Table 30: SRQEA compared to SPSO-2011 and a GA algorithm for the CEC-2013 functions with 10 dimensions. Bold are best.

10 Dimensions	***SRQEA***			**SPSO-2011** [123]			**GA** [125]			
Function	**Min**	**Mean**	**Std dev**	**Min**	**Median**	**Std dev**	**Min**	**Median**	**Mean**	**Std dev**
14 Sphere [duplicated]	**0.00E+00**	**0.00E+00**	**0.00E+00**	**0.00E+00**	**0.00E+00**	**0.00E+00**	**0.00E+00**	**0.00E+00**	**0.00E+00**	**0.00E+00**
15 R HC elliptic	4.61E+04	5.22E+05	4.18E+05	2.09E+03	3.63E+04	7.36E+04	**0.00E+00**	**0.00E+00**	**0.00E+00**	**0.00E+00**
16 Rotated bent cigar	4.59E+01	3.48E+03	2.00E+03	**0.00E+00**	2.68E+05	1.66E+07	**0.00E+00**	**0.00E+00**	**0.00E+00**	**0.00E+00**
17 Rotated discus	7.17E+03	3.09E+04	1.19E+04	1.35E+03	8.87E+03	4.56E+03	**0.00E+00**	**0.00E+00**	**0.00E+00**	**0.00E+00**
18 Different powers	**0.00E+00**	**0.00E+00**	**0.00E+00**	**0.00E+00**	**0.00E+00**	3.14E-05	**0.00E+00**	**0.00E+00**	**0.00E+00**	**0.00E+00**
19 Rotated Rosenbrock	5.59E-04	2.76E+00	4.01E+00	**0.00E+00**	9.80E+00	4.97E+00	**0.00E+00**	**0.00E+00**	**0.00E+00**	**0.00E+00**
20 Rotated Schaffers F7	3.31E+01	8.88E+01	3.64E+01	2.60E+00	2.11E+01	1.33E+01	**2.94E-06**	**1.42E-03**	**4.44E-02**	**2.10E-01**
21 Rotated Ackley	**2.01E+01**	**2.03E+01**	8.04E-02	2.02E+01	**2.03E+01**	**6.72E-02**	2.02E+01	2.04E+01	2.04E+01	8.64E-02
22 Rotated Weierstrass	3.05E+00	7.26E+00	**1.38E+00**	1.30E+00	4.80E+00	1.50E+00	**0.00E+00**	**2.60E+00**	**3.43E+00**	2.90E+00
23 Rotated Griewank	7.03E-01	2.87E+00	1.67E+00	1.00E-01	3.00E-01	2.71E-01	**0.00E+00**	**3.69E-02**	**4.03E-02**	**2.82E-02**
24 Rastrigin	**0.00E+00**	**0.00E+00**	**0.00E+00**	3.00E+00	1.09E+01	5.66E+00	**0.00E+00**	**0.00E+00**	2.73E-01	4.91E-01
25 Rotated Rastrigin	1.49E+01	4.48E+01	1.62E+01	3.00E+00	1.39E+01	6.56E+00	**1.99E+00**	**5.97E+00**	**6.36E+00**	**2.29E+00**
26 NC rotated Rastrigin	1.64E+01	5.21E+01	1.46E+01	5.40E+00	2.08E+01	9.82E+00	**1.99E+00**	**8.52E+00**	**1.01E+01**	**6.30E+00**
27 Schwefel 7	**0.00E+00**	**6.66E-02**	**1.79E-01**	3.23E+02	8.34E+02	2.34E+02	1.87E-01	**1.86E+01**	2.74E+01	2.75E+01
28 Rotated Schwefel 7	4.01E+02	**8.14E+02**	**2.12E+02**	3.37E+02	**7.74E+02**	2.51E+02	**2.53E+02**	8.51E+02	8.31E+02	2.58E+02
29 Rotated Katsuura	2.20E-01	**5.95E-01**	**2.11E-01**	2.00E-01	**5.00E-01**	2.46E-01	**5.18E-02**	1.34E+00	1.28E+00	3.26E-01
30 Lunacek bi-Rastrigin	**0.00E+00**	**0.00E+00**	**0.00E+00**	1.04E+01	1.89E+01	5.87E+00	1.02E+01	**1.11E+01**	1.12E+01	7.73E-01
31 R Lunacek bi-Rastrigin	2.17E+01	4.27E+01	1.02E+01	1.25E+01	1.78E+01	**4.53E+00**	**1.22E+01**	**1.74E+01**	**1.86E+01**	5.22E+00
32 RE Griewank Rosen.	8.65E-01	5.92E+00	3.54E+00	3.00E-01	9.00E-01	3.89E-01	**2.45E-01**	**5.11E-01**	**5.32E-01**	**1.48E-01**
33 RE Schaffers F6	**1.00E+00**	**2.30E+00**	5.94E-01	2.00E+00	3.40E+00	**4.19E-01**	1.70E+00	**3.21E+00**	3.21E+00	5.05E-01

Table 31: SRQEA compared to SPSO-2011 and a GA algorithm for the CEC-2013 functions with 30 dimensions. Bold are best.

30 Dimensions	***SRQEA***			**SPSO-2011** [123]			**GA** [125]			
Function	**Min**	**Mean**	**Std dev**	**Min**	**Median**	**Std dev**	**Min**	**Median**	**Mean**	**Std dev**
14 Sphere [duplicated]	**0.00E+00**	**0.00E+00**	**0.00E+00**	**0.00E+00**	**0.00E+00**	1.88E-13	**0.00E+00**	**0.00E+00**	**0.00E+00**	**0.00E+00**
15 R HC elliptic	8.12E+05	1.78E+06	6.45E+05	6.92E+04	3.09E+05	1.67E+05	**2.26E+04**	**1.10E+05**	**1.55E+05**	**1.37E+05**
16 Rotated bent cigar	**1.27E-02**	**8.81E+01**	**2.63E+02**	1.17E+06	1.19E+08	5.24E+08	5.62E+02	**8.41E+06**	3.28E+07	7.55E+07
17 Rotated discus	7.49E+04	1.19E+05	2.06E+04	2.73E+04	3.91E+04	6.70E+03	**2.18E-02**	**2.77E-01**	**9.08E-01**	**1.26E+00**
18 Different powers	**0.00E+00**	**0.00E+00**	**0.00E+00**	**0.00E+00**	**0.00E+00**	4.91E-05	**0.00E+00**	**0.00E+00**	**0.00E+00**	**0.00E+00**
19 Rotated Rosenbrock	**6.54E-02**	**1.43E+01**	7.99E+00	2.00E-01	2.83E+01	2.83E+01	1.96E+00	**1.97E+01**	2.04E+01	**7.92E+00**
20 Rotated Schaffers F7	1.32E+02	1.79E+02	3.00E+01	5.02E+01	8.69E+01	**2.11E+01**	**4.70E+00**	**4.04E+01**	**4.58E+01**	2.97E+01
21 Rotated Ackley	2.07E+01	**2.09E+01**	7.01E-02	**2.07E+01**	**2.09E+01**	5.89E-02	2.08E+01	2.10E+01	2.10E+01	**5.34E-02**
22 Rotated Weierstrass	2.87E+01	**3.50E+01**	**2.42E+00**	2.11E+01	**2.84E+01**	4.43E+00	**1.98E+01**	4.03E+01	3.70E+01	6.44E+00
23 Rotated Griewank	4.19E-02	2.11E-01	7.20E-02	1.00E-01	3.00E-01	1.48E-01	**7.40E-03**	**7.39E-02**	**8.35E-02**	**4.66E-02**
24 Rastrigin	**0.00E+00**	**0.00E+00**	**0.00E+00**	5.07E+01	1.08E+02	2.74E+01	5.97E+00	**1.99E+01**	2.13E+01	1.07E+01
25 Rotated Rastrigin	1.50E+02	2.73E+02	6.74E+01	4.88E+01	9.45E+01	3.54E+01	**1.99E+01**	**3.68E+01**	**3.77E+01**	**9.55E+00**
26 NC rotated Rastrigin	2.04E+02	2.97E+02	5.15E+01	1.12E+02	1.98E+02	3.86E+01	**3.68E+01**	**8.02E+01**	**8.10E+01**	**1.95E+01**
27 Schwefel 7	**2.72E-02**	**4.40E-01**	**2.56E-01**	2.96E+03	4.02E+03	6.19E+02	5.90E+01	**9.66E+02**	1.01E+03	4.74E+02
28 Rotated Schwefel 7	2.48E+03	**3.46E+03**	**4.66E+02**	**1.94E+03**	**3.80E+03**	6.94E+02	2.88E+03	4.11E+03	4.10E+03	6.93E+02
29 Rotated Katsuura	6.13E-01	**1.25E+00**	**3.28E-01**	4.00E-01	**1.40E+00**	3.59E-01	**2.22E-01**	2.83E+00	2.72E+00	5.05E-01
30 Lunacek bi-Rastrigin	**0.00E+00**	**4.77E-04**	**2.30E-03**	7.27E+01	1.15E+02	2.02E+01	4.27E+01	**5.78E+01**	6.01E+01	1.10E+01
31 R Lunacek bi-Rastrigin	1.50E+02	2.45E+02	5.13E+01	7.68E+01	1.17E+02	2.46E+01	**5.08E+01**	**7.23E+01**	**7.45E+01**	**1.80E+01**
32 RE Griewank Rosen.	4.89E+01	1.30E+02	4.51E+01	2.80E+00	9.00E+00	4.42E+00	**1.68E+00**	**3.65E+00**	**4.14E+00**	**1.99E+00**
33 RE Schaffers F6	**9.16E+00**	1.38E+01	1.36E+00	1.05E+01	1.40E+01	1.11E+00	1.23E+01	**1.39E+01**	**1.37E+01**	**4.78E-01**

Appendix B Fitness functions

In this section, formulae are given for the fitness functions used in the QIEA analysis in chapter 4.

Traditional fitness functions [8]

D is the number of dimensions.

$$f_{01}(x)=\sum_{i=1}^{D}x_i^{\,2}$$

$$f_{02}(x)=\sum_{i=1}^{D}|x_i|+\prod_{i=1}^{D}|x_i|$$

$$f_{03}(x)=\sum_{i=1}^{D}\left(\sum_{j=1}^{i}x_i\right)^2$$

$$f_{04}(x)=\max_i\left(|x_i|\right)$$

$$f_{05}(x)=\sum_{i=1}^{D-1}\left[100\left(x_{i+1}-x_i^{\,2}\right)^2+\left(x_i-1\right)^2\right]$$

$$f_{06}(x)=\sum_{i=1}^{D}\left(\lfloor x_i+0.5\rfloor\right)^2$$

$$f_{07}(x)=\sum_{i=1}^{D}ix_i^{\,4}+\text{random}[0,1)$$

$$f_{08}(x)=-\sum_{i=1}^{D}\left(x_i\sin\left(\sqrt{|x_i|}\right)\right)$$

$$f_{09}(x)=\sum_{i=1}^{D}\left[x_i^{\,2}-10\cos\left(2\pi x_i\right)+10\right]$$

$$f_{10}(x)=-20\exp\left(-0.2\sqrt{\frac{1}{30}\sum_{i=1}^{D}x_i^{\,2}}\right)$$

$$-\exp\left(\frac{1}{30}\sum_{i=1}^{D}\cos\left(2\pi x_i\right)\right)+20+e$$

$$f_{11}(x)=\frac{1}{4000}\sum_{i=1}^{D}x_i^{\,2}-\prod_{i=1}^{D}\cos\left(\frac{x_i}{\sqrt{i}}\right)+1$$

$$f_{12}(x)=\frac{\pi}{D}\left(\begin{array}{l}10\sin^2(\pi y_1)\\+\sum_{i=1}^{D-1}\left[(y_i-1)^2\left(1+10\sin^2(\pi y_{i+1})\right)\right]\end{array}\right)$$

$$+\sum_{i=1}^{D}u(x_i,10,100,4)$$

and

$$f_{13}(x)=0.1\left(\begin{array}{l}\sin^2(\pi 3x_1)\\+\sum_{i=1}^{D-1}\left[(x_i-1)^2\left(1+\sin^2(3\pi x_{i+1})\right)\right]\\+(x_D-1)^2\left(1+\sin^2(2\pi x_D)\right)\end{array}\right)$$ where:

$$+\sum_{i=1}^{D}u(x_i,5,100,4)$$

$$u(x_i,a,k,m)=\begin{cases}k(x_i-a)^m, & x_i>a\\0, & -a\le x_i\le a\\k(-x_i-a)^m, & x_i<a\end{cases}$$

$$y_i=1+\frac{1}{4}(x_i+1)$$

CEC-2013 fitness functions [9]

f_n^* is a minimum height offset, **o** is an origin offset, $\mathbf{M_1}$ and $\mathbf{M_2}$ are rotation matrices, Λ^α is a diaganol matrix, T_{OSZ} and T_{asy} are non-linear transformation functions.

$$f_{14}(x)=\sum_{i=1}^{D}z_i^{\,2}+f_{14}^{\,*},\mathbf{z}=\mathbf{x}-\mathbf{o}$$

$$f_{15}(x)=\sum_{i=1}^{D}\left(10^6\right)^{\frac{i-1}{D-1}}z_i^{\,2}+f_{15}^{\,*},\mathbf{z}=T_{osz}\left(\mathbf{M}_1(\mathbf{x}-\mathbf{o})\right)$$

$$f_{16}(x)=z_1^{\,2}+10^6\sum_{i=2}^{D}z_i^{\,2}+f_{16}^{\,*},\mathbf{z}=\mathbf{M_2}T_{asy}^{0.5}\left(\mathbf{M}_1(\mathbf{x}-\mathbf{o})\right)$$

$$f_{17}(x)=10^6z_1^{\,2}+\sum_{i=2}^{D}z_i^{\,2}+f_{17}^{\,*},\mathbf{z}=T_{osz}\left(\mathbf{M}_1(\mathbf{x}-\mathbf{o})\right)$$

$$f_{18}(x)=\sqrt{\sum_{i=1}^{D}|z_i|^{2+4\frac{i-1}{D-1}}}+f_{18}^{\,*},\mathbf{z}=\mathbf{x}-\mathbf{o}$$

$$f_{19}(x)=\sum_{i=1}^{D-1}\left(100\left(z_i^2-z_{i+1}\right)^2+(z_i-1)^2\right)+f_{19}^{\,*},$$

$$\mathbf{z}=\mathbf{M}_1\left(\frac{2.048(\mathbf{x}-\mathbf{o})}{100}\right)+1$$

$$f_{20}(x)=\left(\frac{1}{D-1}\sum_{i=1}^{D-1}\left(\sqrt{z_i}\left(1+\sin^2\left(50z_i^{0.2}\right)\right)\right)\right)^2+f_{20}^{*},$$

$$z_i=\sqrt{y_i^2+y_{i+1}^2},\ \mathbf{y}=\Lambda^{10}\mathbf{M}_2T_{asy}^{0.5}\left(\mathbf{M}_1\left(\mathbf{x}-\mathbf{o}\right)\right)$$

$$f_{21}(x)=-20\exp\left(-0.2\sqrt{\frac{1}{D}\sum_{i=1}^{D}z_i^2}\right)$$

$$-\exp\left(\frac{1}{D}\sum_{i=1}^{D}\cos\left(2\pi z_i\right)\right)+20+e+f_{21}^{*},$$

$$\mathbf{z}=\Lambda^{10}\mathbf{M}_2T_{asy}^{0.5}\left(\mathbf{M}_1\left(\mathbf{x}-\mathbf{o}\right)\right)$$

$$f_{22}(x)=\sum_{i=1}^{D}\left(\sum_{k=0}^{20}\left[0.5^k\cos\left(2\pi\cdot3^k\left(z_i+0.5\right)\right)\right]\right)$$

$$-D\left(\sum_{k=0}^{20}\left[0.5^k\cos\left(2\pi\cdot3^k\cdot0.5\right)\right]\right)+f_{22}^{*},$$

$$\mathbf{z}=\Lambda^{10}\mathbf{M}_2T_{asy}^{0.5}\left(\mathbf{M}_1\left(\frac{0.5\left(\mathbf{x}-\mathbf{o}\right)}{100}\right)\right)$$

$$f_{23}(x)=\sum_{i=1}^{D}\frac{z_i^2}{4000}-\prod_{i=1}^{D}\cos\left(\frac{z_i}{\sqrt{i}}\right)+1+f_{23}^{*},$$

$$\mathbf{z}=\Lambda^{100}\mathbf{M}_1\left(\frac{600\left(\mathbf{x}-\mathbf{o}\right)}{100}\right)$$

$$f_{24}(x)=\sum_{i=1}^{D}\left(z_i^2-10\cos\left(2\pi z_i\right)+10\right)+f_{24}^{*},$$

$$\mathbf{z}=\Lambda^{10}T_{asy}^{0.2}\left(T_{osz}\left(\frac{5.12\left(\mathbf{x}-\mathbf{o}\right)}{100}\right)\right)$$

$$f_{25}(x)=\sum_{i=1}^{D}\left(z_i^2-10\cos\left(2\pi z_i\right)+10\right)+f_{25}^{*},$$

$$\mathbf{z}=\mathbf{M}_1\Lambda^{10}\mathbf{M}_2T_{asy}^{0.2}\left(T_{osz}\left(\mathbf{M}_1\left(\frac{5.12\left(\mathbf{x}-\mathbf{o}\right)}{100}\right)\right)\right)$$

$$f_{26}(x)=\sum_{i=1}^{D}\left(z_i^2-10\cos\left(2\pi z_i\right)+10\right)+f_{26}^{*},$$

$$\widehat{\mathbf{x}}=\mathbf{M}_1\left(\frac{5.12\left(\mathbf{x}-\mathbf{o}\right)}{100}\right),$$

$$y_i=\begin{cases}\widehat{x}_i, & \left|\widehat{x}_i\right|\le0.5\\ \text{round}\left(2\widehat{x}_i\right)/2, & \left|\widehat{x}_i\right|>0.5\end{cases},$$

$$\mathbf{z}=\mathbf{M}_1\Lambda^{10}\mathbf{M}_2T_{asy}^{0.2}\left(T_{osz}\left(y\right)\right)$$

$$f_{27}(x) = 418.9829D - \sum_{i=1}^{D} g(z_i) + f_{27}^{*},$$

$$\mathbf{z} = \Lambda^{10}\left(\frac{1000(\mathbf{x}-\mathbf{o})}{100}\right) + 4.209687462275036e+002,$$

$$g(z_i) = \begin{cases} z_i \sin\left(|z_i|^{1/2}\right), & |z_i| \le 500 \\ \left(500 - \operatorname{mod}(z_i, 500)\right) & \\ \cdot \sin\left(\sqrt{\left|500 - \operatorname{mod}(z_i, 500)\right|}\right) & \\ -\frac{(z_i - 500)^2}{10000D}, & z_i > 500 \\ \left(\operatorname{mod}(z_i, 500) - 500\right) & \\ \cdot \sin\left(\sqrt{\left|\operatorname{mod}(z_i, 500) - 500\right|}\right) & \\ -\frac{(z_i + 500)^2}{10000D}, & z_i < 500 \end{cases}$$

$$f_{28}(x) = 418.9829D - \sum_{i=1}^{D} g(z_i) + f_{28}^{*},$$

$$\mathbf{z} = \Lambda^{10}\mathbf{M}_1\left(\frac{1000(\mathbf{x}-\mathbf{o})}{100}\right)$$

$$+4.209687462275036e+002,$$

$$g(z_i) = \begin{cases} z_i \sin\left(|z_i|^{1/2}\right), & |z_i| \le 500 \\ \left(500 - \mathrm{mod}(z_i, 500)\right) & \\ \cdot \sin\left(\sqrt{\left|500 - \mathrm{mod}(z_i, 500)\right|}\right) & \\ -\dfrac{(z_i - 500)^2}{10000D}, & z_i > 500 \\ \left(\mathrm{mod}(z_i, 500) - 500\right) & \\ \cdot \sin\left(\sqrt{\left|\mathrm{mod}(z_i, 500) - 500\right|}\right) & \\ -\dfrac{(z_i + 500)^2}{10000D}, & z_i < 500 \end{cases}$$

$$f_{29}(x) = \frac{10}{D^2}\prod_{i=1}^{D}\left(1 + i\sum_{j=1}^{32}\frac{\left|2^j z_i - \text{round}\left(2^j z_i\right)\right|}{2^j}\right)^{\frac{10}{D^{1.2}}}$$

$$-\frac{10}{D^2} + f_{29}^{*},$$

$$\mathbf{z} = \mathbf{M}_2 \Lambda^{100}\left(\mathbf{M}_1\left(\frac{5(\mathbf{x}-\mathbf{o})}{100}\right)\right)$$

$$f_{30}(x) = \min\left(\sum_{i=1}^{D}\left(\hat{x}_i - \mu_0\right)^2, D + s\sum_{i=1}^{D}\left(\hat{x}_i - \mu_1\right)^2\right)$$

$$+10\left(D - \sum_{i=1}^{D}\cos\left(2\pi\hat{z}_i\right)\right) + f_{30}^{*},$$

$$\mu_0 = 2.5,\ \mu_1 = -\sqrt{\frac{\mu_0^2 - 1}{s}},\ 1 - \frac{1}{2\sqrt{D+20} - 8.2},$$

$$\mathbf{y} = \frac{10(\mathbf{x}-\mathbf{o})}{100},\ \hat{x}_i = 2\,\text{sign}\left(x_i\right) y_i + \mu_0,$$

$$\mathbf{z} = \Lambda^{100}\left(\hat{\mathbf{x}} - \mu_0\right)$$

$$f_{31}(x) = \min\left(\sum_{i=1}^{D}(\hat{x}_i - \mu_0)^2, D + s\sum_{i=1}^{D}(\hat{x}_i - \mu_1)^2\right)$$

$$+10\left(D - \sum_{i=1}^{D}\cos(2\pi\hat{z}_i)\right) + f_{31}^{*},$$

$$\mu_0 = 2.5,\ \mu_1 = -\sqrt{\frac{\mu_0^2 - 1}{s}},\ 1 - \frac{1}{2\sqrt{D+20} - 8.2},$$

$$\mathbf{y} = \frac{10(\mathbf{x} - \mathbf{o})}{100},\ \hat{x}_i = 2\,\mathrm{sign}(x_i)\,y_i + \mu_0,$$

$$\mathbf{z} = \mathbf{M_2}\Lambda^{100}(\mathbf{M_1}(\hat{\mathbf{x}} - \mu_0))$$

$$g_1(x) = \sum_{i=1}^{D}\frac{x_i^2}{4000} - \prod_{i=1}^{D}\cos\left(\frac{x_i}{\sqrt{i}}\right) + 1$$

$$g_2(x, y) = 100\left(x^2 - y\right)^2 + (x-1)^2$$

$$f_{32}(x) = \sum_{i=1}^{D} g_1\left(g_2\left(z_i, z_{\mathrm{mod}(i,D-1)+1}\right)\right) + f_{32}^{*},$$

$$\mathbf{z} = \mathbf{M_1}\left(\frac{5(\mathbf{x} - \mathbf{o})}{100}\right) + 1$$

$$g(x, y) = 0.5 + \frac{\sin^2\left(\sqrt{x^2 + y^2}\right) - 0.5}{\left(1 + 0.001\left(x^2 + y^2\right)\right)^2}$$

$$f_{33}(x) = \sum_{i=1}^{D} g_1\left(g_2\left(z_i, z_{\mathrm{mod}(i,D-1)+1}\right)\right) + f_{33}^{*},$$

$$\mathbf{z} = \mathbf{M_2}T_{asy}^{0.5}\mathbf{M_1}((\mathbf{x} - \mathbf{o}))$$

Appendix C Code Listings

While experimenting with optimising locomotion in models on the Newton Dynamics system, it was necessary to change the hinge joint code in order produce successful movement patterns. The original version is shown in C.1 and the successful modified version is in C.2.

C.1 Original hinge code

```
void CustomHingeActuator::SubmitConstraintsFreeDof(dFloat timestep, const
dMatrix& matrix0, const dMatrix& matrix1)
{
	if (m_flag) {
		dFloat jointangle = GetActuatorAngle();
		dFloat relAngle = jointangle - m_angle;
		NewtonUserJointAddAngularRow(m_joint, -relAngle,
&matrix0.m_front[0]);

		dFloat step = m_angularRate * timestep;
		if (dAbs(relAngle) > 2.0f * dAbs(step)) {
			dFloat desiredSpeed = -dSign(relAngle) * m_angularRate;
			dFloat currentSpeed = GetJointOmega();
			dFloat accel = (desiredSpeed - currentSpeed) / timestep;
			NewtonUserJointSetRowAcceleration(m_joint, accel);
		}
		NewtonUserJointSetRowMinimumFriction(m_joint, -m_maxForce);
		NewtonUserJointSetRowMaximumFriction(m_joint, m_maxForce);
		NewtonUserJointSetRowStiffness(m_joint, 1.0f);
	}
	else {
		CustomHinge::SubmitConstraintsFreeDof(timestep, matrix0,
matrix1);
	}

}
```

C.2 Modified hinge code

```
void CustomHingeActuator::SubmitConstraintsFreeDof(dFloat timestep, const
dMatrix& matrix0, const dMatrix& matrix1)
{
	if (m_flag) {
		dFloat jointangle = GetActuatorAngle();
		dFloat relAngle = jointangle - m_angle;
		NewtonUserJointAddAngularRow(m_joint, -relAngle,
&matrix0.m_front[0]);

		dFloat desiredSpeed = -dSign(relAngle) * 50.0;
		dFloat currentSpeed = GetJointOmega() * 40.0;
		dFloat accel = (desiredSpeed - currentSpeed);// / timestep;
		NewtonUserJointSetRowAcceleration(m_joint, accel);

		NewtonUserJointSetRowMinimumFriction(m_joint, -m_maxForce);
```

```
                NewtonUserJointSetRowMaximumFriction(m_joint, m_maxForce);
                NewtonUserJointSetRowStiffness(m_joint, 1.0f);
        }
        else {
                CustomHinge::SubmitConstraintsFreeDof(timestep, matrix0,
matrix1);
        }
}
```