Minimum Torque Gait Transitions in Quadruped Rovers Using Central Pattern Generators

by

Lukas Zhornyak

Supervisor: M. Reza Emami

April 2020

B.A.Sc. Thesis

Division of Engineering Science
UNIVERSITY OF TORONTO

Minimum Torque Gait Transitions in Quadruped Rovers Using Central Pattern Generators

by

Lukas Zhornyak

Supervisor: M. Reza Emami

April 2020

Abstract

A self-contained variable-speed gait generation architecture is developed with a focus on producing torque-optimal and stable gait transitions, based on a central pattern generator producing gait patterns interpreted by a recurrent neural network. This model is initialize to mimic the results of optimization at fixed speeds. Preliminary results based on the initialized model suggest sufficient capacity to achieve this goal but full optimization proves too time-consuming to conduct. Further developments of this model are proposed, both to generalize the results and to significantly reduce the training time by allow end-to-end optimization.

Acknowledgements

I would like to begin by thanking Professor M. Reza Emami for all his advice and support through the years, without which I do not believe I would be where I am today. I thank the members of the Division of Engineering Science at the University of Toronto for providing a space where I could explore a wide variety of topics and the support needed to do so. I also thank my friends and classmates who helped keep me sane and on track, even when assignments and exams seemed overwhelming. Finally, I would like to thank my parents and my sister for always being a bedrock of support.

Contents

List of Figures

List of Tables

1 Introduction

Quadruped animals use a variety of gaits, which are well documented and studied [1]. Depending on the relative ordering of footfalls, these gaits can be labelled as a walk, pace, trot, or gallop; these gait categories remain consistent across species, even though the duration of footfalls varies [2], [3]. The selection of the appropriate gait appears to be largely dependent on the desired speed of movement with smooth transitions typically appearing between gaits. A quadruped rover model based on the structure of a domestic cat was developed by the author in a previous work and was used to produce torque-optimal gaits at a small number of constant speeds [4]. This work adapts the prior work to develop a novel online trajectory generation system that is able to generate stable and optimal gaits at intermediary speeds, with specific focus placed on the transitions between gaits.

Cross-species analysis of gait transitions has found that the transition speed is approximately biomechanically equivalent even in species who exhibit very dissimilar gaiting [5] and that the energy cost per kilogram for gaits is preserved across species [6]. In studies examining gait transitions in cats, cats were observed employing two distinct types of gait transition: a gradual transition in all limbs and an abrupt transition in one limb and a gradual transition in the rest [7], [8]. These gradual transitions often span across multiple step cycles, forming stable intermediary gaits.

To understand how gait transitions occur, the control mechanism of animals transiting gaits during locomotion must be studied. There is enough evidence to suggest that rhythmic patterns in animals are generated centrally in the nervous system without the need for an external input [9], [10]. The collections of neurons responsible for these patterns are collectively known as a Central Pattern Generator (CPG). Various types of CPG models have been developed, such as connectionist models [11], vector maps [12] and systems of coupled oscillators [13][14].

Most research on gait generation in robotic applications has focused on controller design [15], dynamic analysis [16] or optimization [4], [17]; it is only relatively recently that work attempting reproduce the dynamics of gait generation using CPGs

has appeared [18]–[21]. CPGs are of interest here as they allow for autonomous and stable modulation of known trajectories to produce a self-stabilizing design without feedback. Particularly, they allow for the automatic selection and transition between different gaits in multi-legged rovers. However, the existing literature has relied heavily on empirically chosen parameters and gaitings, often directly switching between multiple parameter sets for the CPG. Current research suggests that each CPG is responsible for producing multiple motions, and that each motion is not associated with a unique CPG [10]. Furthermore, the focus to date has been on producing stable motion and analysing the ability of a CPG to modulate its output using various forms of feedback – not on optimizing the resulting movement. This work seeks to address these shortcomings by developing single CPG to generate the optimal movement at a desired speed by transitioning through multiple gaits continuously, without auxiliary inputs or modification.

2 Background

2.1 Rover Model

An illustration of a quadruped rover and associated variables is given in fig. 1. The associated geometric and physical properties are summarised in tables 1 and 2. To understand the difference between the basic categories of gaits, several terms are defined. The instant that a leg's toe touches the ground is referred to as its Anterior Extreme Position (AEP), while the instant when the toe takes-off is referred to as the Posterior Extreme Position. The leg movement from AEP to PEP is referred to as the swing phase, whereas the leg movement from PEP to AEP is referred to as the stance phase, as illustrated in fig. 2. The ratio of stride time to the total cycle time is known as the duty factor β. Using the front left as a reference, the relative phase offset ϕ of each other leg can be used to distinguish between pace, trot, and lateral sequence walks, as illustrated in fig. 3.

The desired motion for each leg can be defined in each phase by using its toe position in the sagittal plane, $x_3(t)$ and $z_3(t)$, as well as the orientation of the metacarpal/metatarsal link $\phi_3(t)$, each in the form of a Bezier curve of order m [22]:

$$\delta_m(t) = \frac{1}{(t_f)^m} \sum_{k=0}^{m} \frac{m!}{k!(m-k)!} t^k (t_f - t)^{m-k} \cdot a_k \tag{1}$$

Table 1: Denavit-Hartenberg (DH) parameters for the front and hind legs

Length L_B **(cm)**	**Width (cm)**	**Height (cm)**	**Mass** M_B **(kg)**
29.4	15.3	5.0	2.486

Table 2: DH and mass parameters for the front and hind legs

j	$a_j^{\mathrm{F}} = L_j^{\mathrm{F}}$ **(cm)**	α_j^{F}	d_j^{F} **(cm)**	m_j^{H} **(kg)**	$a_j^{\mathrm{H}} = L_j^{\mathrm{H}}$ **(cm)**	α_j^{H}	d_j^{H} **(cm)**	m_j^{H} **(kg)**
1	15.0	0	0	0.061	13.5	0	0	0.061
2	13.2	0	0	0.139	11.7	0	0	0.139
3	7.5	0	0	0.114	9.5	0	0	0.116

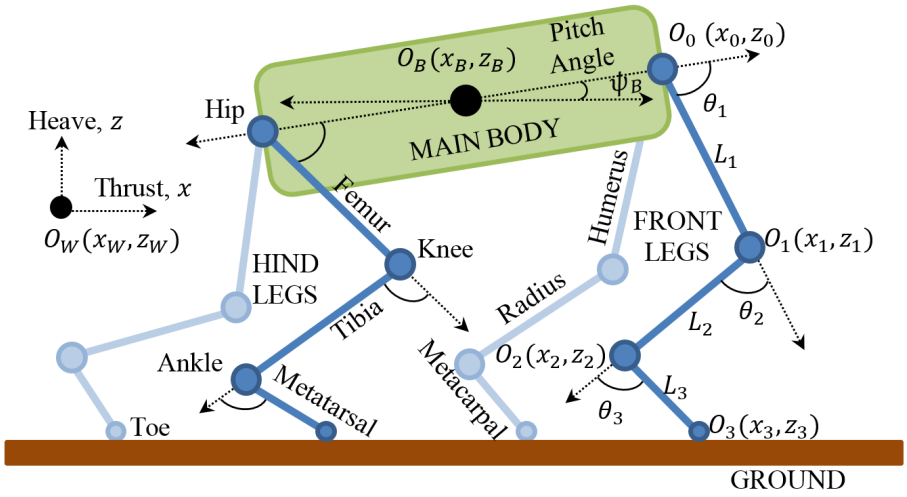


Figure 1: Rover model in the sagittal plane.

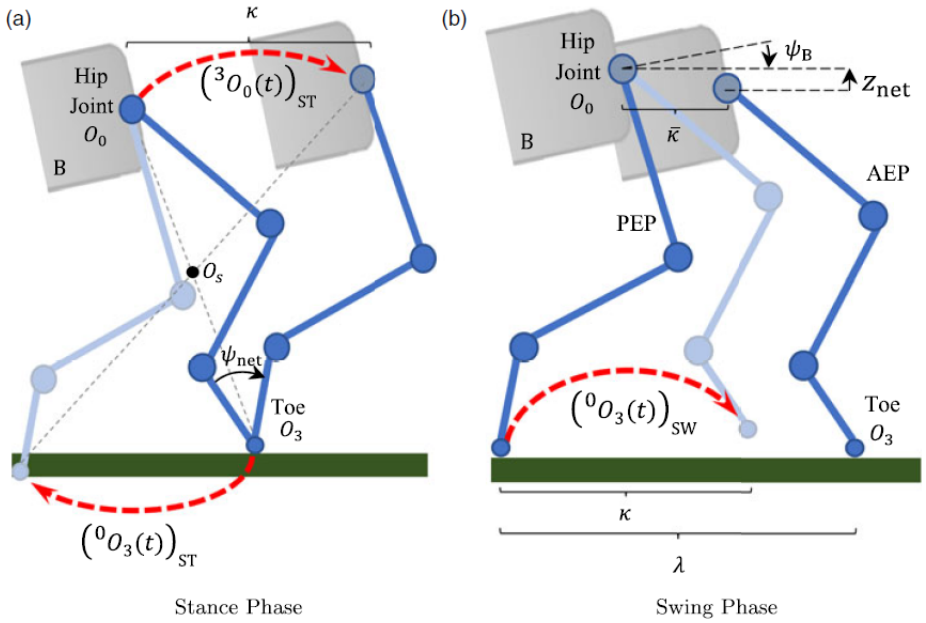


Figure 2: Front leg models showing the stance (a) and swing (b) phases.

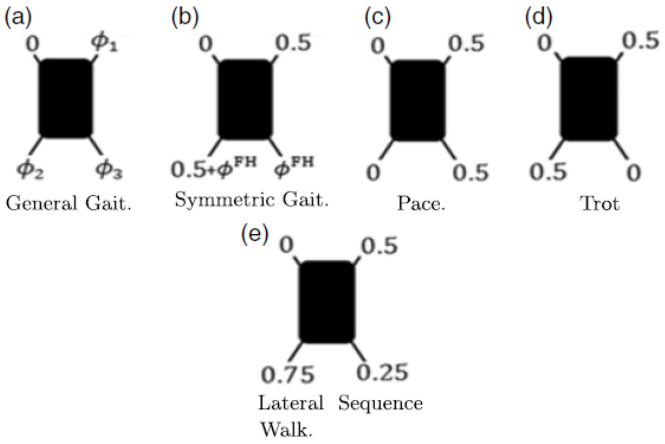


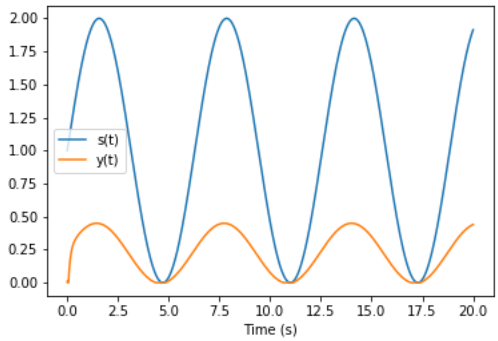
Figure 3: The relative phases of the legs for a (a) general gait, (b) symmetric gait, (c) pace, (d) trot and (e) lateral sequence walk. Note that 0 is the reference leg.

where t_f is the duration of the curve and a_k parametrizes the curve. With the appropriate choice of boundary conditions for the Bezier curves (i.e. initial and final position and velocity), the curves for both swing and stance phases form a single continuous curve. Through inverse kinematics, this correspondingly defines a continuous joint trajectory for each leg. Section 3.1 discuses the optimization of the variables that define these trajectories.

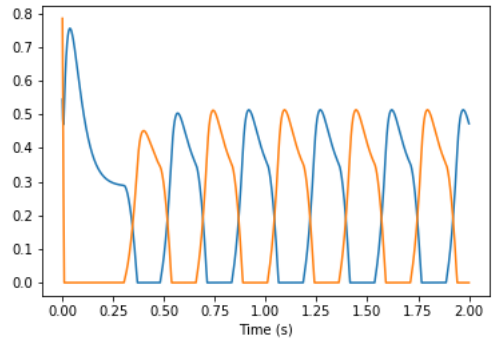
For a more thorough definition of the rover model, including frame and coordinate definitions as well as boundary conditions for the trajectories, refer to section 2 in the prior work on gait optimization [4].

2.2 Central Pattern Generators

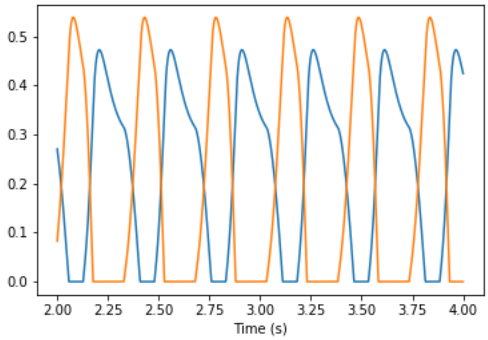
Central Pattern Generators (CPG), a form of neural circuit capable of producing rhythmic motor impulses without a rhythmic input, have been found in most vertebrate and are believed to form the basis for various repetitive motions, including walking [9], [10]. These circuits are capable of producing these patterns in the absence of sensory input, instead relying on sensory data to modulate the generated pattern. Almost all known examples of CPG observed are formed of sets of reciprocally inhibitory neurons. In these sets, a high potential on one neuron inhibits the firing of another. This other neuron will fire on rebound from the inhibition, producing the



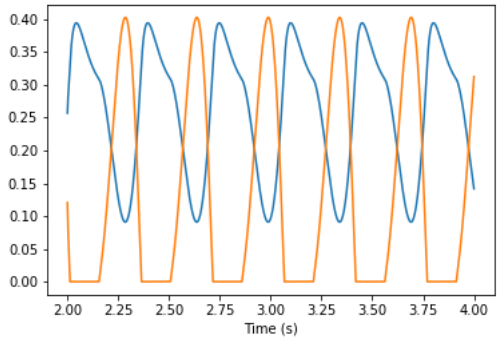
(a) Single Neuron



(b) Phase Ratio 0.5

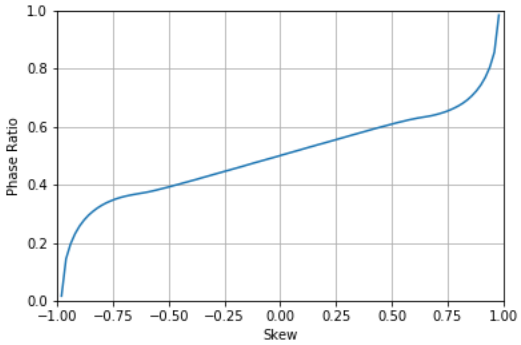


(c) Phase Ratio 0.6

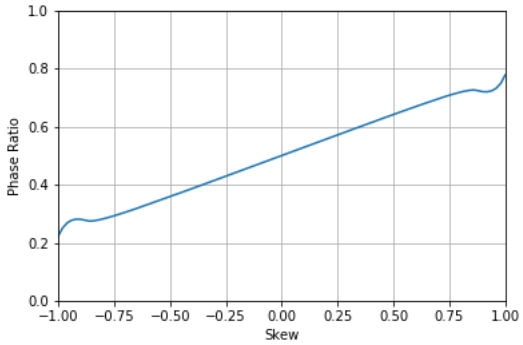


(d) Phase Ratio 0.65

Figure 4: Example patterns produced by Matsuoka oscillator. Figure 4a shows the response of a single oscillator to a sinusoidal input signal. Figures 4b to 4d display a half-centre oscillator self-stabilizing to a specified phase ratio from random initial conditions.



(a) $\tau = 0.24$



(b) $\tau = 1.00$

Figure 5: Phase ratio versus skew value in the $\mathbf{A}$ matrix for two values of τ. Note the linear region in both plots.

observed oscillation. In the simplest case of two neurons, the neurons fire alternately in a stable rhythm. This is known as a half-centre oscillator.

Various types of CPG models have been developed, such as connectionist models [11], vector maps [12] and systems of coupled oscillators [13][14]. One of the more commonly used models is the Matsuoka oscillator [14], described by the following equations:

$$T\frac{\mathrm{d}\mathbf{x}}{\mathrm{d}t} + \mathbf{x} = \mathbf{s} - \mathbf{A}\mathbf{y} - \mathbf{b}^T\mathbf{f} \tag{2}$$

$$\tau\frac{\mathrm{d}\mathbf{f}}{\mathrm{d}t} + \mathbf{f} = \mathbf{y} \tag{3}$$

$$y_i = \max(x_i, 0) \tag{4}$$

$$a_{ii} = 0 \tag{5}$$

Here, $\mathbf{x}$ describes the internal state of the neuron, $\mathbf{y}$ is the output, $\mathbf{s}$ is the input, and $\mathbf{f}$ is the fatigue of the neuron. Each of these variables are n–dimensional vectors, where n is the number of neurons modelled by the system. The remaining parameters are constants describing the behaviour of the network. T and τ are time constants inversely associated with the rate of growth and rate of decay, respectively, of the output, $\mathbf{A}$ is an $n \times n$ matrix with a zero diagonal that describes the strength of the connections between the set of neurons, and $\mathbf{b}$ is an n–dimensional vector describing the strength of the fatigue for each neuron.

A single neuron exhibits some high-pass features, visible in fig. 4a. At the same time, the magnitude of the steady-state neuron input is linearly proportional to magnitude of the input. By adding a second neuron and forming a half-centre oscillator, oscillatory motion is produced, shown in fig. 4.

For the half-centre oscillator, the value of T and τ control the frequency of the oscillation. Assuming a constant ratio τ/T is maintained, the period of oscillation becomes proportional to the value of T. In contrast, modifying this ratio affects the transition between the oscillators; a large τ/T produces faster transitions and vice versa.

Similarly, $\mathbf{A}$ controls the phase ratio between the two oscillators. Here, an oscillator is considered to be *in-phase* when its value is larger than the value of any other oscillator; the phase ratio γ is then defined as the proportion of a cycle where one oscillator is in-phase. By applying a skew δ to $\mathbf{A}$,

$$a_{12} = 1.5 - \delta, \quad a_{21} = 1.5 + \delta \tag{6}$$

the relationship observed in fig. 5 is obtained. In this way, the phase ratio of the oscillation can be controlled while maintaining an approximately constant oscillation period. Figure 5 also demonstrates that there exists an approximately linear relationship for small values of δ. This relationship collapses when one neuron is no longer able to reset to 0 at some point in the cycle, as demonstrated in figs. 4c and 4d. By increasing the value of τ, the linear region can be expanded.

2.3 Recurrent Neural Networks

A Recurrent Neural Network (RNN) is a generalization of traditional feed-forward neural network where the output of the network is in some way connected to the input of the network. This allows for arbitrary-length sequences of data to be processed and inferences to be drawn from them. Particularly, they allow for the conversion of a sequence in one domain to a sequence in another domain. This has applied for great results in the field of language translation [23], [24] and many other distinct fields [25], [26].

In all the examples provided above, the flow of information is tightly controlled between the input, hidden state, and output of the RNN. This is done through *gates*, learned functions that control when and how the hidden state is updated. One common configuration of gates is known as a Gated Recurrent Unit (GRU) [24], defined

by the following set of equations:

$$\mathbf{r}_t = \sigma(\mathbf{W}_{xr}\mathbf{x}_t + \mathbf{b}_{xr} + \mathbf{W}_{hr}\mathbf{h}_{t-1} + \mathbf{b}_{hr}) \tag{7}$$

$$\mathbf{z}_t = \sigma(\mathbf{W}_{xz}\mathbf{x}_t + \mathbf{b}_{xz} + \mathbf{W}_{hz}\mathbf{h}_{t-1} + \mathbf{b}_{hz}) \tag{8}$$

$$\tilde{\mathbf{h}}_t = \tanh\left(\mathbf{W}_{x\tilde{h}}\mathbf{x}_t + \mathbf{b}_{x\tilde{h}} + \mathbf{r}_t(\mathbf{W}_{h\tilde{h}}\mathbf{h}_{t-1} + \mathbf{b}_{h\tilde{h}})\right) \tag{9}$$

$$\mathbf{h}_t = (1 - \mathbf{z}_t)\tilde{\mathbf{h}}_t + \mathbf{z}_t\mathbf{h}_{t-1} \tag{10}$$

where t is the current time step, $\mathbf{h}$ is the hidden state, $\tilde{\mathbf{h}}$ is the hidden state update, $\mathbf{r}_t$ is the reset gate, and $\mathbf{z}_t$ is the update gate. $\mathbf{W}$ and vbb are the trainable weights and biases that describe the behaviours of the network. In this formulation, when the reset reset gate is close to zero, the network updates using only information from the current time step, ignoring prior states. Meanwhile, the update gate controls whether to update the hidden state with new information or preserve the prior states. This is illustrated in fig. 6.

The purpose of controlling how the hidden states are updated is to separate long–term dependencies from short term ones. Since each element of the hidden state has its own update and reset gates, different hidden units can be trained to record dependencies on differing time scales. As a result, the hidden state is not typically directly used as the final output of the RNN; instead, the hidden state is fed to fully-connected network to produce the desired output. Finally, in much the same way as traditional feed-forward neural network, multiple GRUs can be stacked to identify more complex features.

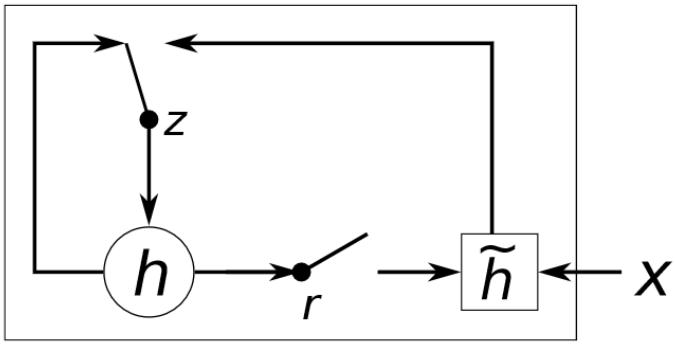


Figure 6: Diagram showing components of a Gated Recurrent Unit (GRU) [24]. x_t is the input and h_t is the hidden state.

3 Gait Generation

Figure 7 illustrates the full optimization and testing structure. Section 3.1 describes the fixed speed optimization, producing a set of optimized variables φ and joint trajectories $\theta^*(t)$ based on Bezier curves (see section 2.1). The optimized variables are used to initialize the CPG pattern generator, producing as a pattern $x(t)$ based on the desired speed $v(t)$) as described in section 3.2.1. Finally, the joint trajectories $\theta^*(t)$ are used to perform the initial training of the RNN, as described in section 3.2.2, which produces joint trajectories $\theta(t)$ based on the pattern $x(t)$. Details on the selection of the simulation environment can be found in appendix A.

3.1 Fixed Speed Optimization

In order to define the boundary conditions for the Bezier curves (section 2.1), a number of parameters need to be defined as optimization variables. These are the Pose (P) variables, which includes the stride length, body pitch, initial off-sagittal velocity, step clearance, joint configuration at AEP, net toe movement during swing phase, and the metacarpal/metatarsal angular velocity at AEP and PEP. The duty factors β of each leg and their relative phase offset ϕ compared to the front left leg are also defined as optimization variables; these are known as the Gait (G) variables. Finally, the target velocity is also optimized. In total, this amounts to 43 optimization variables. All optimization variables are summarised in table 3.

These variables are optimized in a multi-objective optimization against two objective functions:

$$\Gamma_{\mathrm{T}} = \frac{1}{T}\int_0^T \|\tau\|^2 \,\mathrm{d}t \tag{11}$$

$$\Gamma_{\mathrm{S}} = -\frac{1}{T}\int_0^T (\dot{x}_{\mathrm{B}}) \,\mathrm{d}t \tag{12}$$

where τ is a vector of all the joint torques at each time step, $\dot{x}_{\mathrm{B}}$ is the forward velocity of the rover, and T is the total trial time. Equation (11) measures the total torque exerted by each joint across the trial while eq. (12) measures the negative of the

Table 3: The pose, gait and velocity variables for leg k.

φ	**Variables**	**Notes**	**Total: 43**
P_a^k	$\theta^k_{1\,\mathrm{AEP}}$	Hip joint angle at AEP for leg k	4
	$\theta^k_{2\,\mathrm{AEP}}$	Knee joint angle at AEP for leg k	4
	$\theta^k_{3\,\mathrm{AEP}}$	Ankle joint angle at AEP for leg k	4
P_b^k	$\dot{\psi}^k_{\mathrm{AEP}}$	Metacarpal/Metatarsal's angular velocity at AEP for leg k	4
	$\dot{\psi}^k_{\mathrm{PEP}}$	Metacarpal/Metatarsal's angular velocity at PEP for leg k	4
	ψ^k_{net}	Metacarpal/Metatarsal's net angular change for leg k	4
	z^k_{net}	Net change in hip's height during stance for leg k	4
	f_c^k	Toe's step clearance during swing phase for leg k	4
P_c	λ	Stride length	1
	ψ_{B}	Body Pitch	1
	$\dot{y}_{\mathrm{B},i}$	Initial out of sagittal plane velocity	1
G	β^k	Duty factor for leg k	4
	ϕ^k	Relative phase for leg k, excluding the front left leg	3
V	V	Forward velocity	1

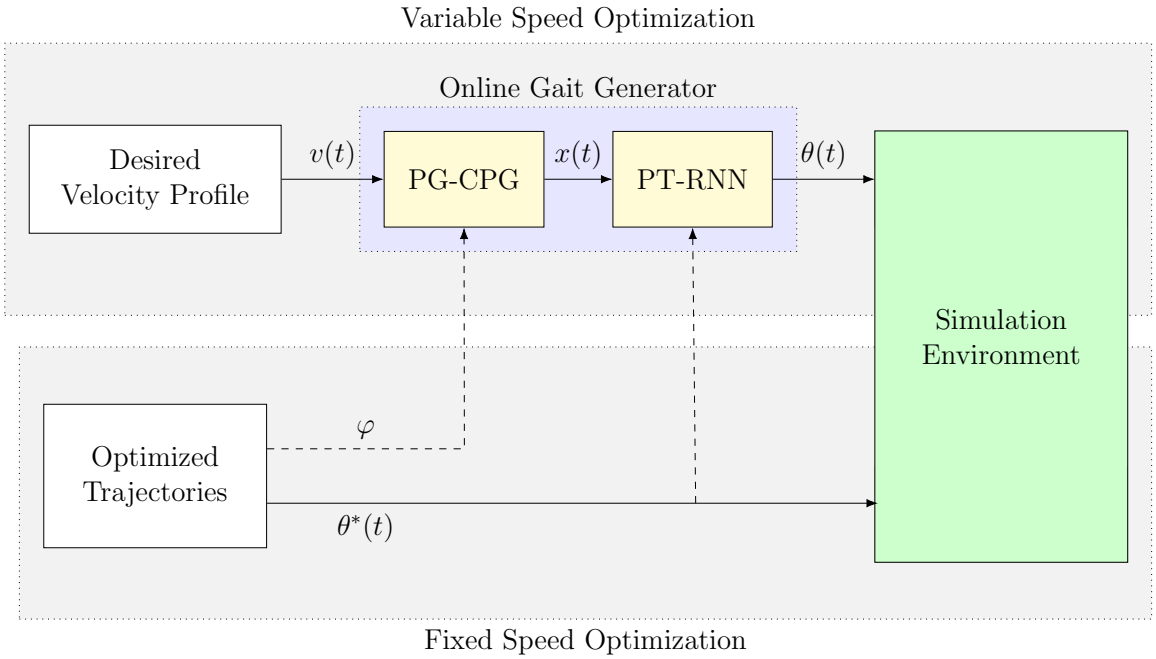


Figure 7: Full optimization and testing structure. Solid lines represent direct data flow while dashed lines represent data used for initialization.

average forward speed. These two objectives are simultaneously minimized using the NSGA-II algorithm [27], a form of genetic algorithm, using a population size of 100, recombination rate of 0.3, and mutation rate of 0.1. From this optimization, a set of optimized trajectories $\theta^*(t)$ at some constant speeds is obtained.

For a more detailed explanation of the optimization variables and algorithm, as well as the constraints imposed on the simulation, refer to section 3 in the original work on gait optimization [4]. Note that formulation presented in the original work also optimized the control gains for a PD controller at each joint. However, no clear relationship was found to exist between the target velocity and the optimized controller gains. Additionally, fixing the gains of each joint of an optimized parameter set to a single, common value had very little effect on the ($< 1\%$) torque objective. Finally, with the control parameters accounting for more than half of the original optimization variables, the rate of objective convergence could be be greatly increased through their emission. For these reasons, the control gains were not optimized in this work.

3.2 Variable Speed Optimization

The online gait generator shown in fig. 7 can be divided into two parts: a Pattern Generating CPG (PG-CPG) and a Pattern Transforming RNN (PT-RNN). The PG-CPG is responsible for producing an oscillatory pattern $x(t)$ that encapsulates the information of the Gait optimization variables. Specifically, it produces an stable oscillatory pattern that represents both the phase offsets between legs and the duty factor of each leg. This oscillatory pattern is provided to the PT-RNN to produce the target joint angles $\theta(t)$ at each time step. Both the PG-CPG and the PT-RNN run with an update frequency of 1 kHz, matching the frequency of the simulation environment.

3.2.1 Pattern Generating CPG

Figure 8 illustrates the structure of the PG-CPG. Twelve neurons are arranged into two layers: a set of four neurons feeding into four pairs of neurons. The output of the eight neurons forming the second layer produce the pattern $x(t)$. $s(t)$ is kept at constant value of 1 for each neuron during steady-state operation with dynamics arising solely through the mutually inhibitory connections between neurons.

The first layer functions as a phase generator, with each neuron producing the phase offset ϕ for a single leg. The phase ratios are encoded into the 4×4 $\mathbf{A}$ matrix by an extension method described in section 2.2 for setting the phase ratio γ of a half-centre oscillator. Instead of optimizing the twelve non-zero elements of $\mathbf{A}$ individually, four skew values δ_i are selected and applied to some initial $\mathbf{A}_0$:

$$\mathbf{A} = \mathbf{A}_0 + \begin{Bmatrix} 0 & \delta_1 - \delta_2 & \delta_1 - \delta_3 & \delta_1 - \delta_4 \\ \delta_2 - \delta_1 & 0 & \delta_2 - \delta_3 & \delta_2 - \delta_4 \\ \delta_3 - \delta_1 & \delta_3 - \delta_2 & 0 & \delta_3 - \delta_4 \\ \delta_4 - \delta_1 & \delta_4 - \delta_2 & \delta_4 - \delta_3 & 0 \end{Bmatrix} \tag{13}$$

In this way, the phase offset ϕ of a leg is encoded as the timing of when the neuron associated with that leg becomes in-phase, relative to the front left leg. A fully

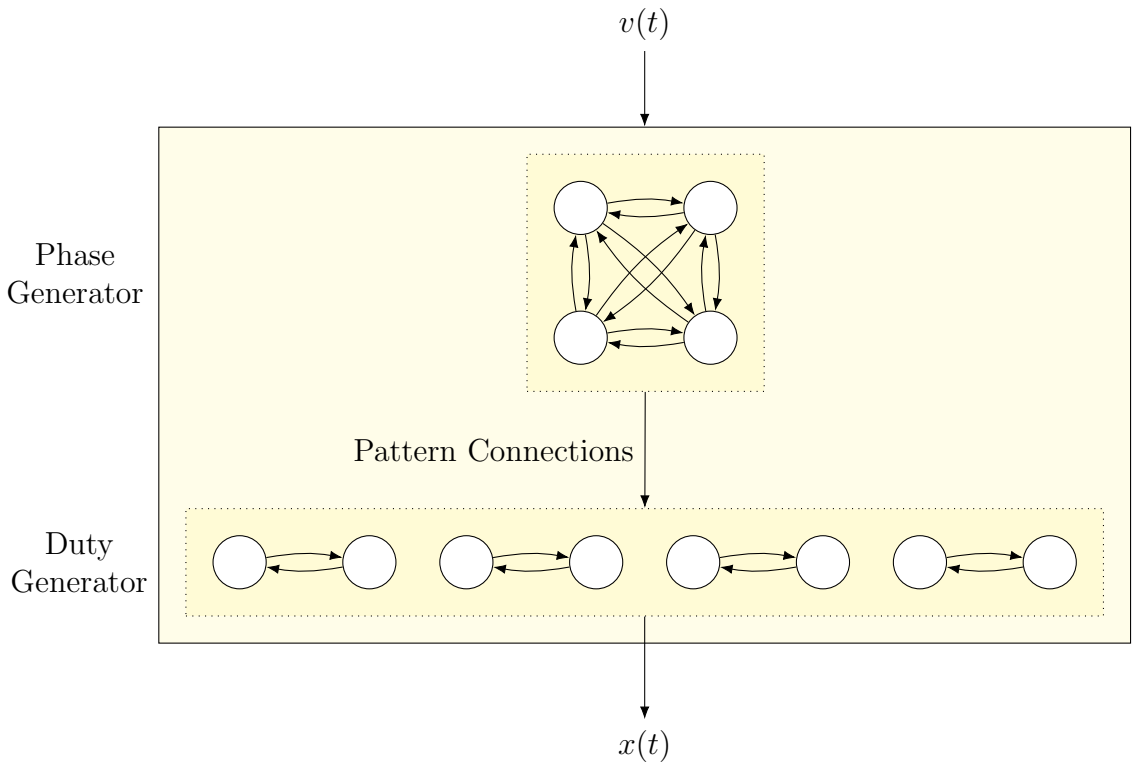


Figure 8: Structure of the Pattern Generating CPG (PG-CPG). Each set of bi-directional arrows represents a mutually inhibitory connections.

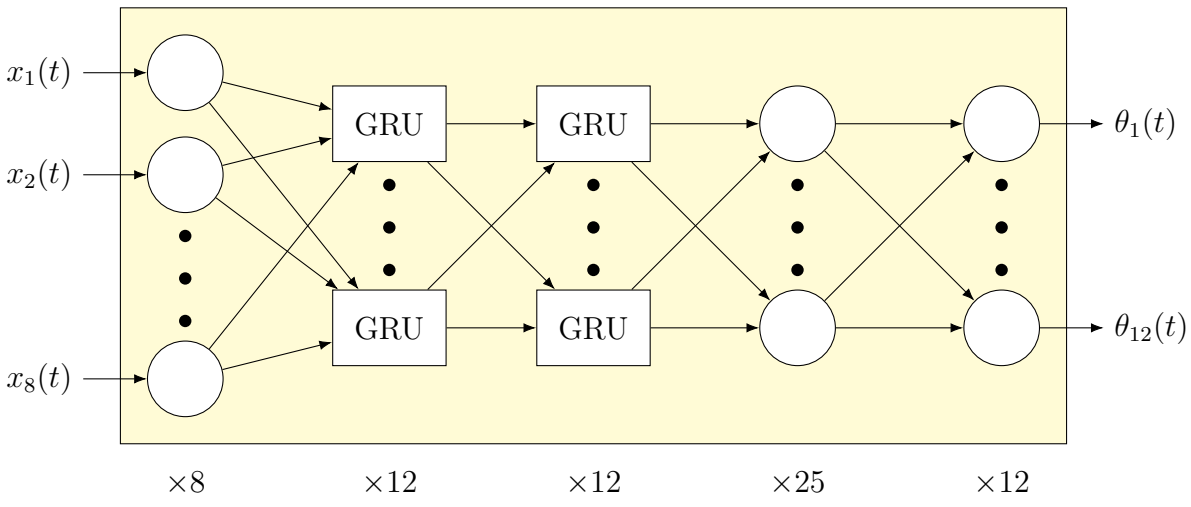


Figure 9: Structure of Pattern Transforming RNN (PT-RNN). GRU layers are represented by boxes while fully connected layers are represented by circles. Dimensions of each layer are labelled.

connected CPG such as this one can produce multiple stable patterns representing the ordering of the legs; to set the desired ordering, $s(t)$ is controlled to a square wave pattern for the first three cycles [14].

Similarly, a skew is applied to each pair of neurons the second layer to encode the duty factor so that one neuron is in-phase during the swing phase and another is in-phase during the stance phase. The stance neuron of each pair is also inhibited by one neuron from the first layer. As this is an inhibitory connection, the stance neuron is connected to the neuron representing the *previous* leg in the ordering, so the end of the previous leg being in-phase marks the start of the stance phase of the current leg. The strength of these connections is not optimized. Rather, they are set to a relatively small value so that the phases of the two layers couple. These connections are labelled as *Pattern Connections* in fig. 8.

The PG-CPG is also responsible for altering the pattern $x(t)$ based on the target velocity $v(t)$. To change the velocity of a single gait, the ratio τ/T is scaled proportionally, as describe in section 2.2. To transition between two gaits, eq. (11) is modified:

$$T\frac{\mathrm{d}\mathbf{x}}{\mathrm{d}t} + \mathbf{x} = \mathbf{s} - \mathbf{A}(t)\mathbf{y} - \mathbf{b}^T\mathbf{f} \tag{14}$$

$$\mathbf{A}(t) = (1 - \sigma_s(v(t) - v_T))\mathbf{A}_1 + \sigma_s(v(t) - v_T)\mathbf{A}_2 \tag{15}$$

$$\sigma_s(v) = \frac{e^{sx}}{e^{sx} + 1} \tag{16}$$

where $\mathbf{A}_1$ and $\mathbf{A}_2$ represent the connections forming the slower and faster gaits, respectively, v_t is the transition velocity, s is the rate of transition, and all other variables are unchanged. In this formulation, 42 parameters can be used to described the gait patterns across velocities: 20 for each of $\mathbf{A}_1$ and $\mathbf{A}_2$ (12 non-zero elements for the first layer and $4 \times 2 = 8$ elements for the second), as well as the transition velocity and rate of transition. A similar formulation can also be created for more than two gaits.

3.2.2 Pattern Transforming RNN

Figure 9 illustrates the architecture of the PT-RNN. Based on the pattern produced by the PG-CPG initialized with two gaits k, the network is initially trained to reduce the loss of $\theta(t)$ relative to both optimized trajectories $\theta_k^*(t)$ at their respective speeds. Here the loss defined as average sum of squared errors across time:

$$\ell_\theta = \frac{1}{T}\int_0^T \left(\|\theta_1^*(t) - \theta_1(t)\|^2 + \|\theta_2^*(t) - \theta_2(t)\|^2\right) \mathrm{d}t \tag{17}$$

In total, the model is composed of 2365 parameters.

3.2.3 Full Model Optimization

Together, the PG-CPG and the PT-RNN have a combined 2407 parameters. Defining these parameters as optimization variables, the NSGA-II algorithm described in section 3.1 is applied to a population size of 1000 individuals. Note that the results of the fixed speed optimization are no longer used during this stage of optimization.

4 Discussion

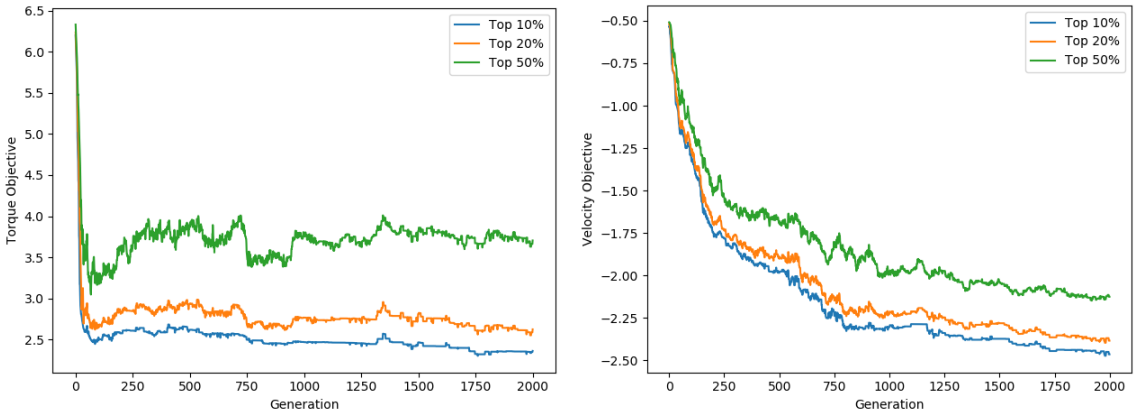
4.1 Fixed Speed Results

Both objective functions were optimized simultaneously using the NSGA-II algorithm to produce the final Pareto front shown in fig. 10. In the prior work, sole optimization of the speed objective (eq. (12)) was found to produce very inefficient gaiting at the boundaries of the simulation constraints. After torque minimization (eq. (11)) at the maximum obtained speed, the resulting gaits show marked differences, suggesting that higher speeds were possible. Results obtained support this, with a maximum obtained speed of $2.55\,\mathrm{m\,s^{-1}}$ in this work compared to $1.97\,\mathrm{m\,s^{-1}}$ in the previous work. Figures 10 to 13 display selected results of this optimization.

Figure 11 shows the torque exerted by each joint of the front left leg across one cycle. Accounting for the difference in gaiting, the torque curves share many similarities. The magnitude of the torques, excluding the larger spike in the knee torque, is very similar. However, the high-frequency oscillation observed in the original simulation are absent, suggesting the desired improvement in numerical stability. The large spike does warrant further investigation, however.

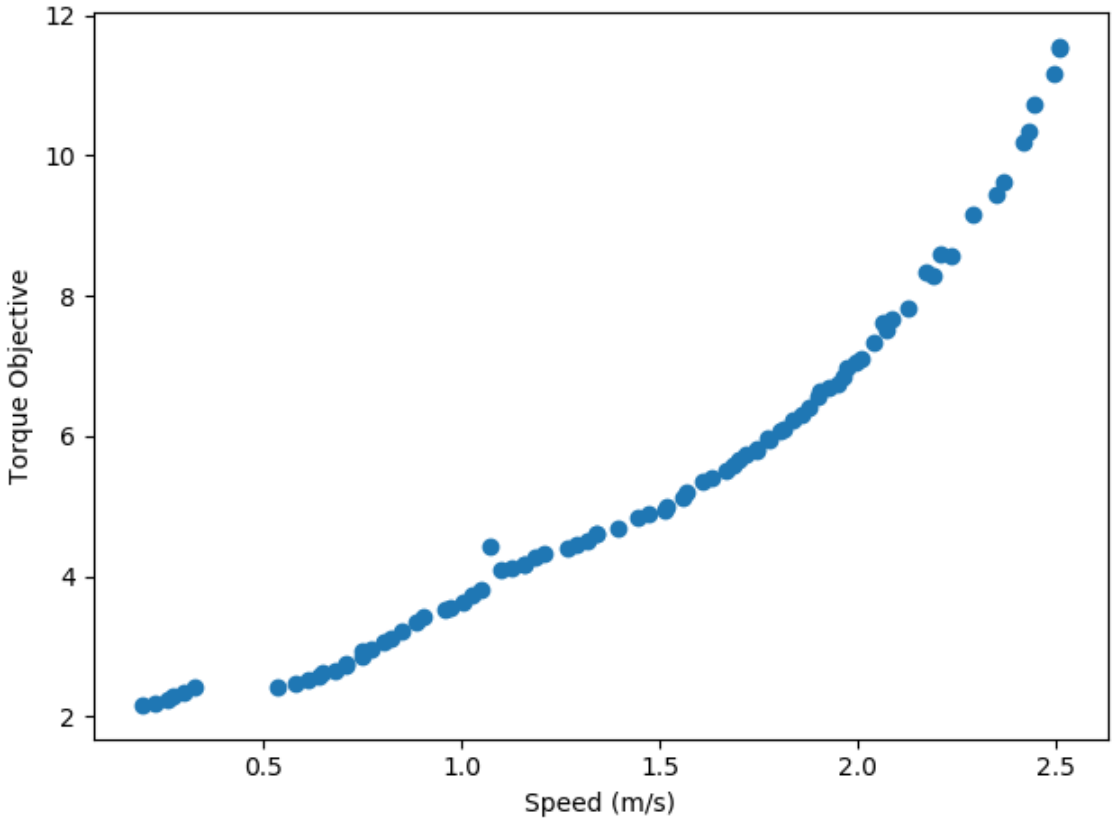
In fig. 12, the trend with respect to increasing speed is almost identical between the two runs. These figures almost provide the clearest examples of the different subpopulations forming the Pareto front, though their correspondence to specific gaits is not clear. Finally, fig. 13 shows how the stride length changes with increasing speed. In contrast to the original results, where the square of the stride length was found to vary linearly with speed, here a linear relationship is observed up until approximately $1.1\,\mathrm{m\,s^{-1}}$ after which the stride length does not increase with increasing speed. Given that only four points were sampled in the original optimization, it is not unreasonable to assume that this relationship may have also held there.

From the fixed speed results, an optimized trajectory at $1.15\,\mathrm{m\,s^{-1}}$ and a trajectory at $1.87\,\mathrm{m\,s^{-1}}$ are selected to initialized the PG-CPG and PT-RNN.



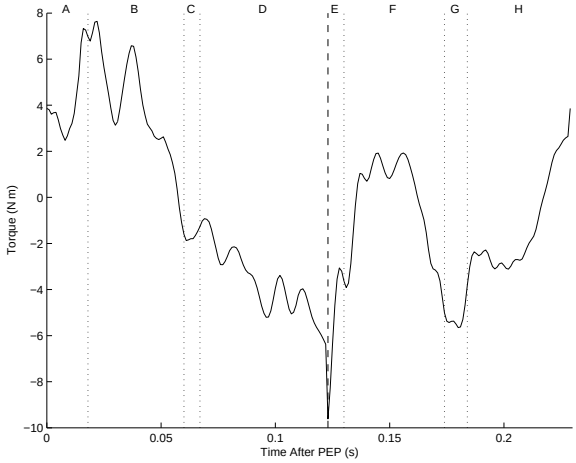
(a) Torque Convergence

(b) Velocity Convergence

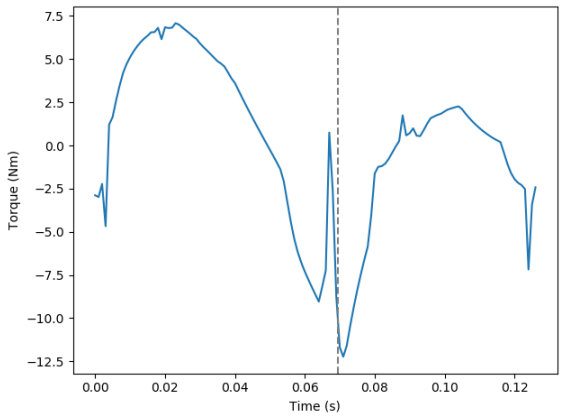


(c) Paretto Front

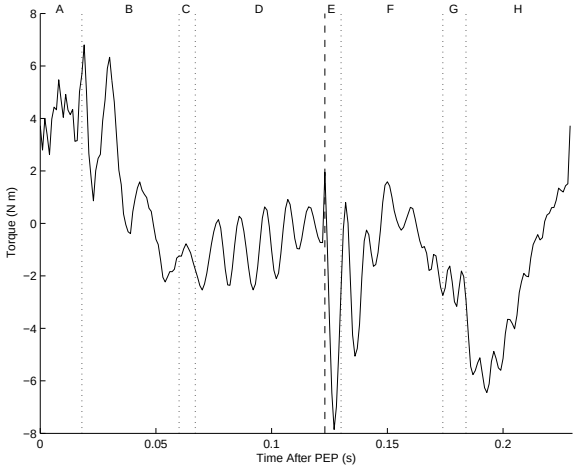
Figure 10: Objective results after training for 2000 generations. Figures 10a and 10b show the convergence for the top 10 %, 20 % and 50 % of the population for the torque objective and velocity objective, respectively. Figure 10c displays the final Pareto front after 2000 generations.



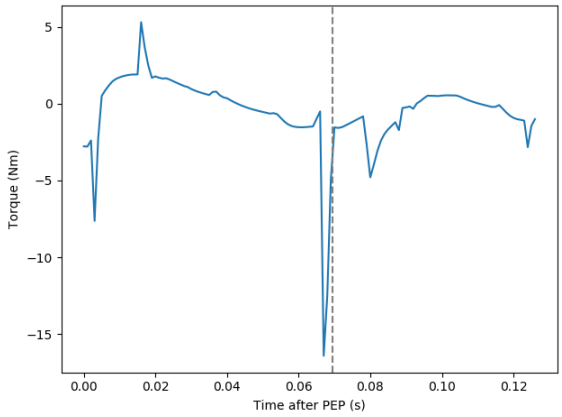
(a) Front left hip torque MATLAB.



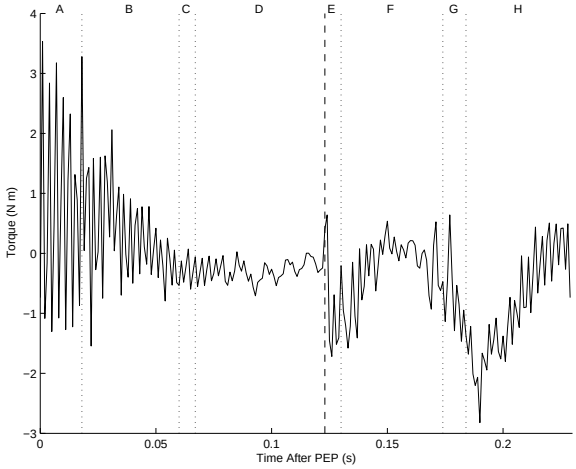
(b) Front left hip torque PyBullet.



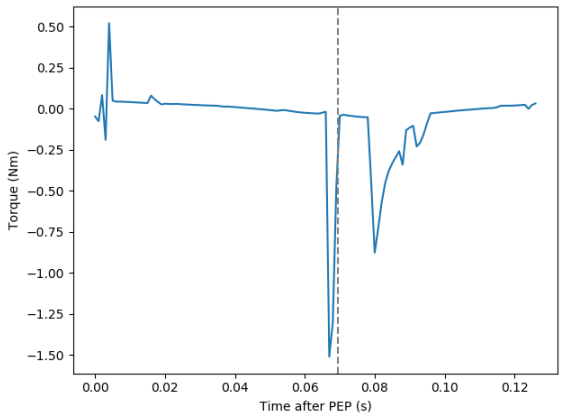
(c) Front left knee torque MATLAB.



(d) Front left knee torque PyBullet.

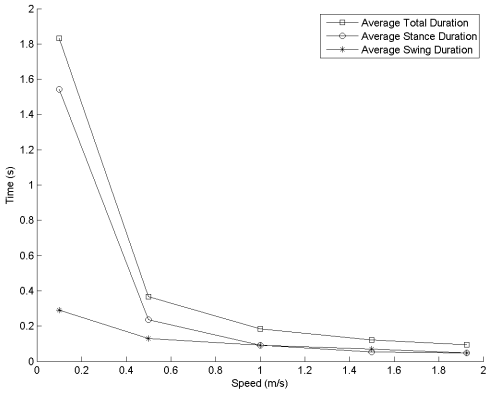


(e) Front left ankle torque MATLAB.

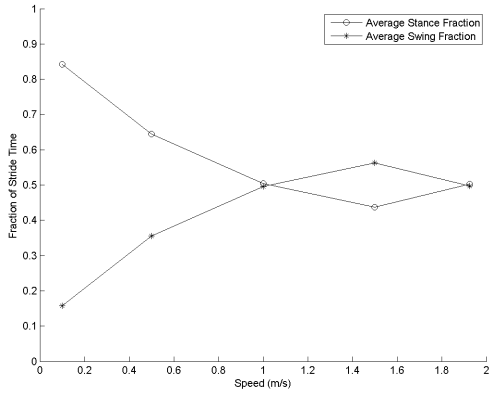


(f) Front left ankle torque PyBullet.

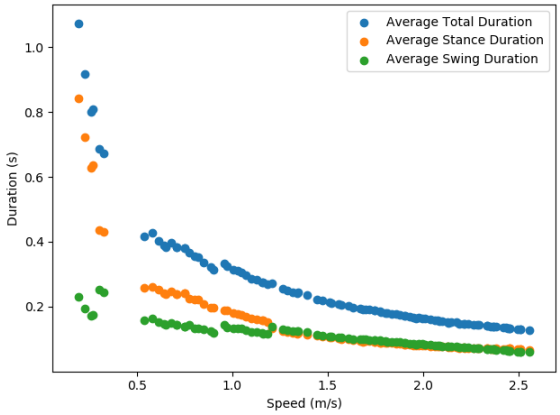
Figure 11: Torques at maximum observed speed in the original optimization (left, $1.97\,\mathrm{m\,s^{-1}}$) and the new optimization (left, $2.55\,\mathrm{m\,s^{-1}}$)



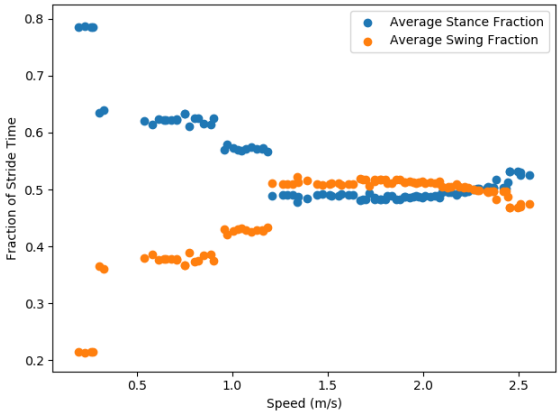
(a) Absolute Duration



(b) Relative Fraction



(c) Absolute Duration



(d) Relative Fraction

Figure 12: Proportion of each cycle in each phase versus attained speed. Figures 12a and 12c show the absolute duration in the original and new optimization, respectively, while figs. 12b and 12d show the relative fraction of the total duration in the original and new optimization, respectively.

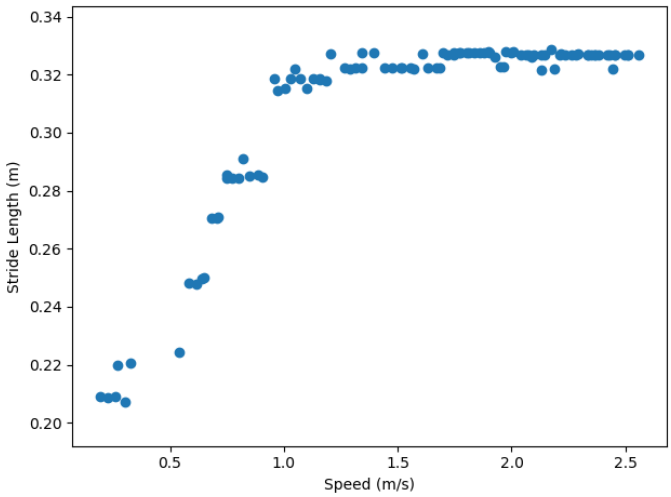


Figure 13: Stride length versus attained speed.

4.2 Variable Speed Results

Unfortunately, results from a full optimization on the model, as described in section 3.2.3, are unavailable. Based on initial attempts, training is estimated to require around a month of full time training. The current situation makes this impractical – a dedicated training machine is not available. Possible approaches to reduce the training load are described in section 5. The rest of this section explores results based solely on the initialized but untrained model in an attempt to provide a proof of concept for the approach describe thus far.

To successfully transform the pattern $x(t)$ into joint trajectories requires that the model proposed in section 3.2.2 has sufficient capacity. Based on results obtained from the initialization, this appears to be the case. Figure 14 shows the evolution of the loss functions across 10 000 evaluations, taking approximately ten minutes. The loss quickly and smoothly converges to a very small value, producing trajectories almost indistinguishable from the original optimized trajectories. This suggests that the model has at least the capacity to encode multiple gaits.

Figure 16 presents the response of the model to a steadily increasing velocity profile. Two features immediately stand out from this figure. The first is that at around $1.5\,\mathrm{m\,s^{-1}}$, the rover falls over and is unable to continue to follow the profile.

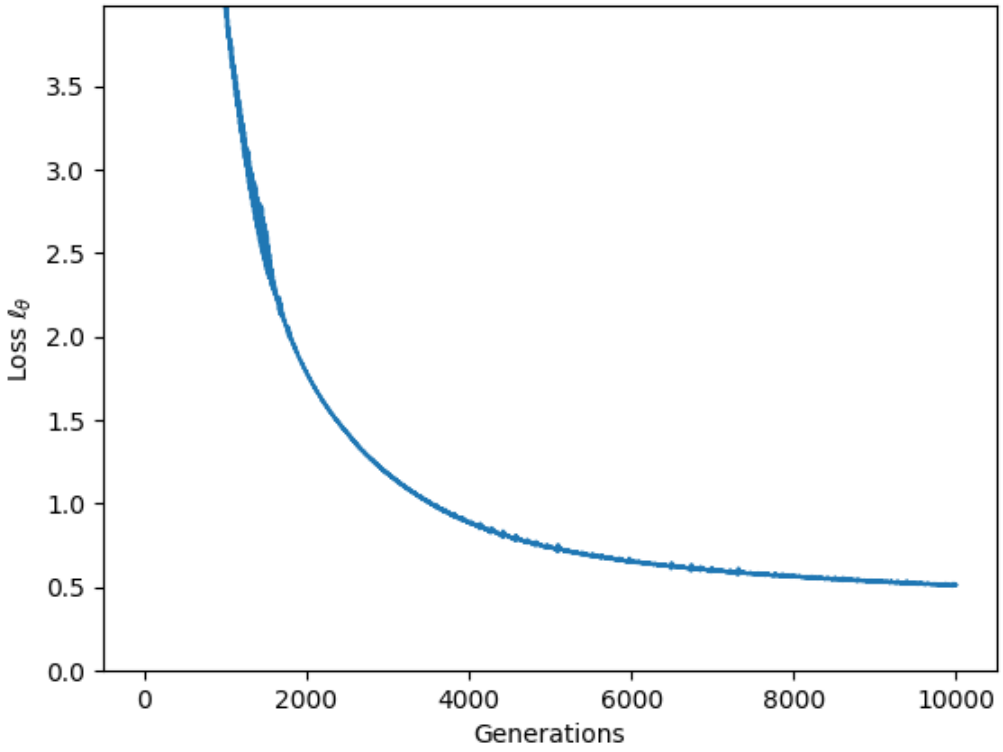
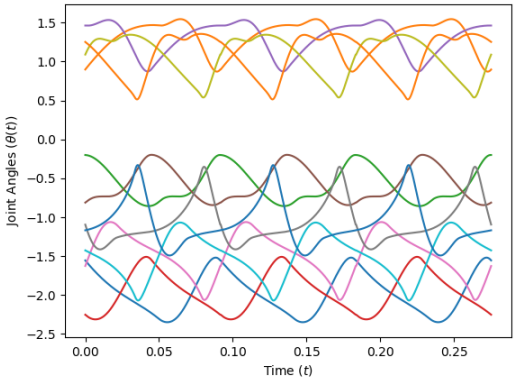
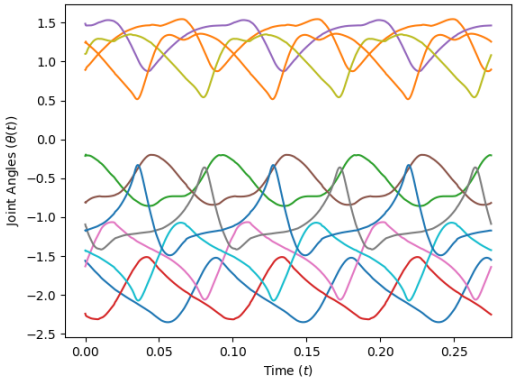


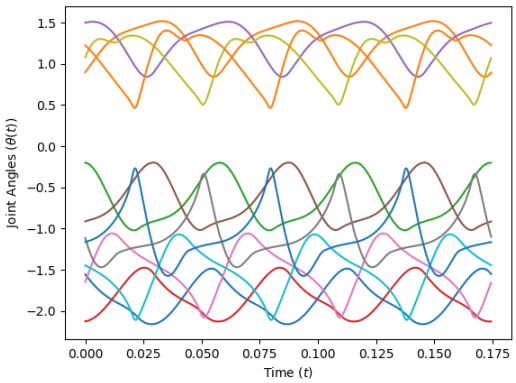
Figure 14: Loss ℓ_θ versus elapsed generations during initialization of PT-RNN.



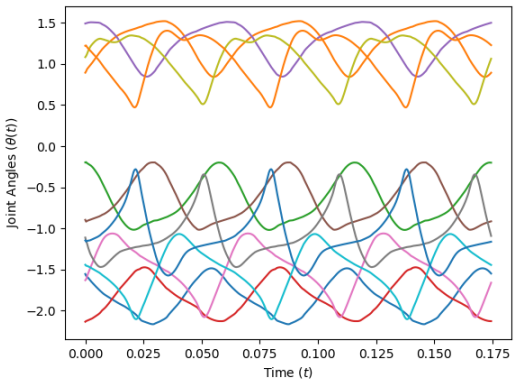
(a) Optimized trajectory at $1.15\,\mathrm{m\,s^{-1}}$.



(b) Reproduced trajectory at $1.15\,\mathrm{m\,s^{-1}}$.



(c) Optimized trajectory at $1.87\,\mathrm{m\,s^{-1}}$.



(d) Reproduced trajectory at $1.87\,\mathrm{m\,s^{-1}}$.

Figure 15: Three cycles of the original optimized trajectories for the two selected gaits and the corresponding reproduced trajectories from the PT-RNN.

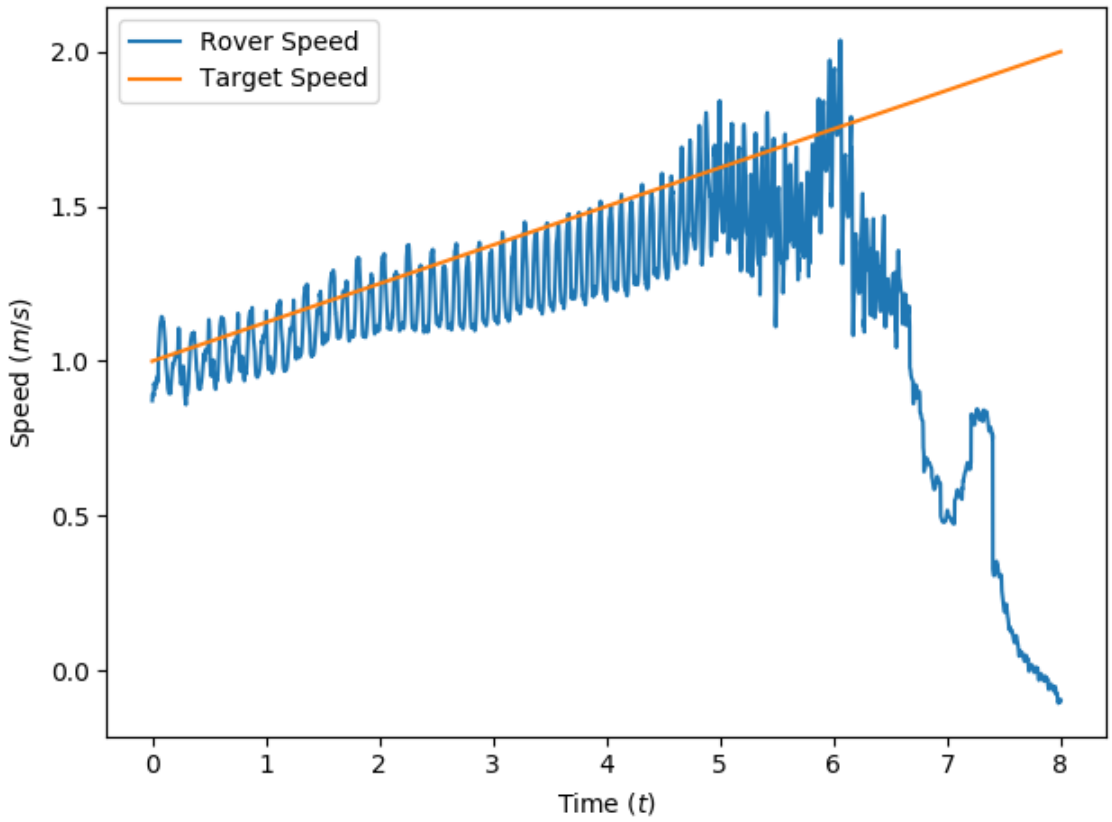


Figure 16: Achieved rover speed compared with the target speed for an eight second profile from $1\,\mathrm{m\,s^{-1}}$ to $2\,\mathrm{m\,s^{-1}}$. The transition speed between gaits was set to $1.5\,\mathrm{m\,s^{-1}}$, about where the rover failed to maintain stability.

This is also the speed set as the transition between the two learned gaits, so this is not surprising[1]. The second is that, at least outside of the transition speeds, the PT-RNN is able to control the period of oscillation to follow the desired speed profile while remaining stable, despite having only been trained on two fixed speed profiles. This further suggests that the PT-RNN has the capacity to encode gaits at a variety of speeds, and, as a result, likely has sufficient capacity for the objectives of this work.

[1]What is, perhaps, surprising is that this is not always the case. On occasion, the rover will complete the trial with violating any constraints. This seems to mostly be dependent on chance, however.

5 Future Work

The most pressing limitation in the current model is that, as formulated in the previous sections, only lateral sequence walks are produced. This is primarily a problem in the fixed speed optimization, however, as all 100 optimized gaits were to some extent a lateral sequence walk. For this reason, the formulation presented in section 3.2.1 assumes a lateral sequence walk. The fixed speed optimization aims, in essence, to find a minimum-torque gait that is open-loop stable. The lack of more dynamic gaits, such as a trot, in the optimization results suggests that it is difficult or impossible to find dynamic gaits that are open-loop stable.

No such limitation presents itself for the PG-CPG. The first layer of the network is the Phase Generator and is responsible for producing the leg ordering – as a fully connected four-neuron CPG, it is easily able to reproduce almost any four-legged walking rhythm [14]. Similarly, the PT-RNN does not make any assumptions about the model and is able to produce an arbitrary set of joint trajectories. While any gait can be encoded into these two networks, no mechanism exists to switch leg ordering.

As mentioned in section 3.2.1, the neuron inputs $s(t)$ can be used to control the leg ordering. This would require an additional network before the PG-CPG; a potential layout for this extended gait generator is illustrated in fig. 17. The addition of the Gait Control network serves two purposes: it allows for the switching of leg orderings during gait transitions and it allows for the incorporation of feedback into the network.

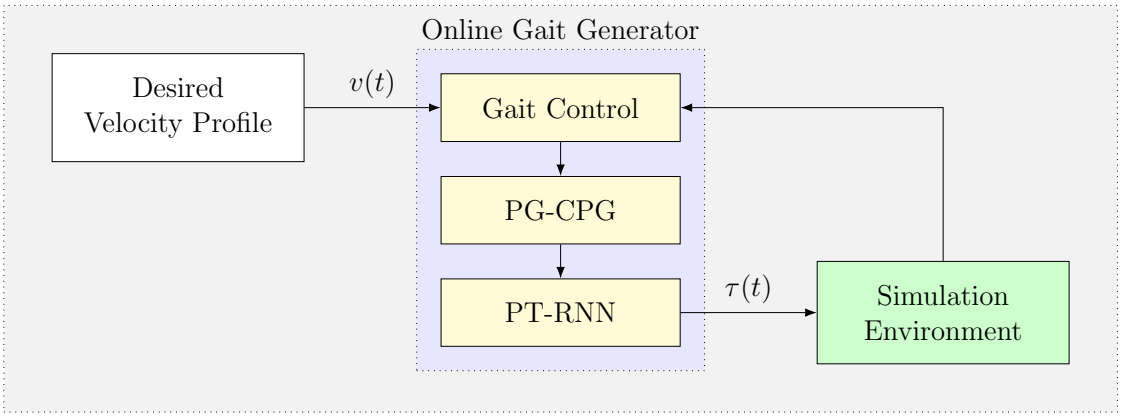


Figure 17: Extended optimization and testing structure.

As a result, this new network is likely to be some form of RNN. By directly training this architecture to produce torque-optimal gaits, the limitations presented by the fixed-speed optimization can be overcome.

This architecture presents even more parameters, so the genetic algorithm used previously is not likely to produce results in a feasible time-frame. Reinforcement-learning algorithms are one approach to reducing the training time, having been used to train a wide variety of quadruped control architectures [28]–[30]. However, recent work presents the potential for even faster training by allowing a gradient to be passed through the entire network. A number of papers have been released in the past couple years exploring differentiable physics engines and showing their efficacy at training quadruped control networks [31], [32]. Through the adjoint-method [33], it is also possible to pass a gradient over the differential equations defining a CPG in a memory and computationally efficient way. Thus, the architecture would be trained end-to-end using traditional gradient descent methods and the CPG would form an oscillatory layer inside a larger RNN. Further exploration is needed to determine the viability of this approach.

6 Conclusion

In a prior work, a method for gait optimization at various fixed speeds was explored and the effects of speed on gait type was examined. This work presents a self-contained gait generation architecture that can produce torque-optimal gaits based only on the desired speed profile. This is accomplished through the combination of a central pattern generator to produce the required gait pattern and a recurrent neural network to transform this pattern into a set of joint trajectories. The results of the multi-objective fixed speed optimization are used to initialize this network.

Unfortunately, time and resources did not permit the full training of the architecture. Preliminary results based on the fixed-speed initialization however, suggest that the presented architecture has sufficient capacity to encode the desired gaits. Particularly, when selecting two gaits from the fixed-speed optimization, the network is able to reproduce them with minimal loss. Furthermore, despite being trained on only two gaits at two fixed speed, a stable gait matching the speed profile is formed. As expected, gait transitions on the untrained model are still unstable.

To address the limitations presented by the fixed-speed optimization and the resultant lateral sequence gaits, an extension on this work is proposed. By incorporating an additional recurrent network providing inputs into the CPG, pattern-switching and feedback can be obtained. This may allow for more dynamic gaits to be produced. Recent work has presented the possibility of end-to-end training, potentially greatly reducing the training time needed.

References

[1] R. M. Alexander, "The gaits of bipedal and quadrupedal animals," *The International Journal of Robotics Research*, vol. 3, no. 2, pp. 49–59, 1984. DOI: 10.1177/027836498400300205.

[2] M. Hildebrand, "The quadrupedal gaits of vertebrates," *BioScience*, vol. 39, no. 11, pp. 766–775, 1989. DOI: 10.2307/1311182.

[3] G. Schöner, W. Jiang, and J. Kelso, "A synergetic theory of quadrupedal gaits and gait transitions," *Journal of Theoretical Biology*, vol. 142, no. 3, pp. 359–391, 1990. DOI: 10.1016/s0022-5193(05)80558-2.

[4] L. Zhornyak and M. R. Emami, "Gait optimization for quadruped rovers," *Robotica*, pp. 1–25, DOI: 10.1017/S0263574719001413.

[5] A. K. Perry, R. Blickhan, A. A. Biewener, N. C. Heglund, and C. R. Taylor, "Preferred speeds in terrestrial vertebrates: Are they equivalent?" *Journal of Experimental Biology*, vol. 137, no. 1, pp. 207–219, 1988.

[6] N. C. Heglund and C. R. Taylor, "Speed, stride frequency and energy cost per stride: How do they change with body size and gait?" *Journal of Experimental Biology*, vol. 138, no. 1, pp. 301–318, 1988.

[7] A. W. English and P. R. Lennard, "Interlimb coordination during stepping in the cat: In-phase stepping and gait transitions," *Brain Research*, vol. 245, no. 2, pp. 353–364, 1982. DOI: 10.1016/0006-8993(82)90818-6.

[8] J. A. Vilensky, J. N. Libii, and A. M. Moore, "Trot-gallop gait transitions in quadrupeds," *Physiology & Behavior*, vol. 50, no. 4, pp. 835–842, 1991. DOI: 10.1016/0031-9384(91)90026-k.

[9] T. G. Brown, "On the nature of the fundamental activity of the nervous centres; together with an analysis of the conditioning of rhythmic activity in progression, and a theory of the evolution of function in the nervous system," *The Journal of Physiology*, vol. 48, no. 1, pp. 18–46, 1914. DOI: 10.1113/jphysiol.1914.sp001646.

[10] E. Marder and D. Bucher, "Central pattern generators and the control of rhythmic movements," *Current Biology*, vol. 11, no. 23, 2001. DOI: 10.1016/s0960-9822(01)00581-4.

[11] P. Arena, "The central pattern generator: A paradigm for artificial locomotion," *Soft Computing*, vol. 4, no. 4, pp. 251–266, 2000. DOI: 10.1007/s005000000051.

[12] M. Okada and Y. Nakamura, "Polynomial design of nonlinear dynamics for brain-like information processing and its application to humanoid whole body motion," *Journal of the Robotics Society of Japan*, vol. 22, no. 8, pp. 1050–1060, 2004. DOI: 10.7210/jrsj.22.1050.

[13] H. Kimura, S. Akiyama, and K. Sakurama, "Realization of dynamic walking and running of the quadruped using neural oscillator," *Autonomous robots*, vol. 7, no. 3, pp. 247–258, 1999.

[14] K. Matsuoka, "Mechanisms of frequency and pattern control in the neural rhythm generators," *Biological Cybernetics*, vol. 56, no. 5-6, pp. 345–353, 1987. DOI: 10.1007/bf00319514.

[15] H. Diedam, D. Dimitrov, P.-B. Wieber, K. Mombaur, and M. Diehl, "Online walking gait generation with adaptive foot positioning through linear model predictive control," in *2008 IEEE/RSJ International Conference on Intelligent Robots and Systems*, IEEE, 2008, pp. 1121–1126.

[16] L. Roussel, C. Canudas-de-Wit, and A. Goswami, "Generation of energy optimal complete gait cycles for biped robots," in *Proceedings. 1998 IEEE International Conference on Robotics and Automation (Cat. No. 98CH36146)*, IEEE, vol. 3, 1998, pp. 2036–2041.

[17] D. J. Lizotte, T. Wang, M. H. Bowling, and D. Schuurmans, "Automatic gait optimization with gaussian process regression.," in *IJCAI*, vol. 7, 2007, pp. 944–949.

[18] T. Fukui, H. Fujisawa, K. Otaka, and Y. Fukuoka, "Autonomous gait transition and galloping over unperceived obstacles of a quadruped robot with cpg modulated by vestibular feedback," *Robotics and Autonomous Systems*, vol. 111, pp. 1–19, 2019, ISSN: 0921-8890. DOI: https://doi.org/10.1016/j.robot.2018.10.002. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0921889018300137.

[19] G. Sartoretti, S. Shaw, K. Lam, N. Fan, M. Travers, and H. Choset, "Central pattern generator with inertial feedback for stable locomotion and climbing in unstructured terrain," Jan. 2018.

[20] A. Spröwitz, A. Tuleu, M. Vespignani, M. Ajallooeian, E. Badri, and A. J. Ijspeert, "Towards dynamic trot gait locomotion: Design, control, and experiments with cheetah-cub, a compliant quadruped robot," *The International Journal of Robotics Research*, vol. 32, no. 8, pp. 932–950, 2013. DOI: 10.1177/0278364913489205. eprint: https://doi.org/10.1177/0278364913489205. [Online]. Available: https://doi.org/10.1177/0278364913489205.

[21] G. Ren, W. Chen, S. Dasgupta, C. Kolodziejski, F. Wörgötter, and P. Manoonpong, "Multiple chaotic central pattern generators with learning for legged locomotion and malfunction compensation," *Information Sciences*, vol. 294, pp. 666–682, 2015, Innovative Applications of Artificial Neural Networks in Engineering, ISSN: 0020-0255. DOI: https://doi.org/10.1016/j.ins.2014.05.001. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0020025514005192.

[22] E. R. Westervelt, J. W. Grizzle, C. Chevallereau, J. H. Choi, and B. Morris, *Feedback control of dynamic bipedal robot locomotion*. CRC press, 2018.

[23] I. Sutskever, O. Vinyals, and Q. V. Le, "Sequence to sequence learning with neural networks," in *Advances in neural information processing systems*, 2014, pp. 3104–3112.

[24] K. Cho, B. Van Merriënboer, C. Gulcehre, D. Bahdanau, F. Bougares, H. Schwenk, and Y. Bengio, "Learning phrase representations using rnn encoder-

decoder for statistical machine translation," *arXiv preprint arXiv:1406.1078*, 2014.

[25] S. H. Park, B. Kim, C. M. Kang, C. C. Chung, and J. W. Choi, "Sequence-to-sequence prediction of vehicle trajectory via lstm encoder-decoder architecture," in *2018 IEEE Intelligent Vehicles Symposium (IV)*, IEEE, 2018, pp. 1672–1678.

[26] C. Ma, C. Yang, F. Yang, Y. Zhuang, Z. Zhang, H. Jia, and X. Xie, "Trajectory factory: Tracklet cleaving and re-connection by deep siamese bi-gru for multiple object tracking," in *2018 IEEE International Conference on Multimedia and Expo (ICME)*, IEEE, 2018, pp. 1–6.

[27] K. Deb, A. Pratap, S. Agarwal, and T. Meyarivan, "A fast and elitist multiobjective genetic algorithm: Nsga-ii," *IEEE transactions on evolutionary computation*, vol. 6, no. 2, pp. 182–197, 2002.

[28] H. Lee, Y. Shen, C.-H. Yu, G. Singh, and A. Y. Ng, "Quadruped robot obstacle negotiation via reinforcement learning," in *Proceedings 2006 IEEE International Conference on Robotics and Automation, 2006. ICRA 2006.*, IEEE, 2006, pp. 3003–3010.

[29] H. Murao, H. Tamaki, and S. Kitamura, "Walking pattern acquisition for quadruped robot by using modular reinforcement learning," in *2001 IEEE International Conference on Systems, Man and Cybernetics. e-Systems and e-Man for Cybernetics in Cyberspace (Cat. No. 01CH37236)*, IEEE, vol. 3, 2001, pp. 1402–1405.

[30] D. Gu and H. Hu, "Reinforcement learning of fuzzy logic controller for quadruped walking robots," in *Proceedings of 15th IFAC World Congress*, 2002, pp. 21–26.

[31] J. Degrave, M. Hermans, J. Dambre, and F. Wyffels, "A differentiable physics engine for deep learning in robotics," *Frontiers in neurorobotics*, vol. 13, p. 6, 2019.

[32] F. de Avila Belbute-Peres, K. Smith, K. Allen, J. Tenenbaum, and J. Z. Kolter, "End-to-end differentiable physics for learning and control," in *Advances in Neural Information Processing Systems*, 2018, pp. 7178–7189.

[33] T. Q. Chen, Y. Rubanova, J. Bettencourt, and D. K. Duvenaud, "Neural ordinary differential equations," in *Advances in neural information processing systems*, 2018, pp. 6571–6583.

[34] D. Kang and J. Hwangho, *Simbenchmark*, 2020. [Online]. Available: `https://leggedrobotics.github.io/SimBenchmark/`.

[35] T. Erez, Y. Tassa, and E. Todorov, "Simulation tools for model-based robotics: Comparison of bullet, havok, mujoco, ode and physx," in *2015 IEEE international conference on robotics and automation (ICRA)*, IEEE, 2015, pp. 4397–4404.

[36] A. Boeing and T. Bräunl, "Evaluation of real-time physics simulation systems," in *Proceedings of the 5th international conference on Computer graphics and interactive techniques in Australia and Southeast Asia*, 2007, pp. 281–288.

[37] E. Coumans and Y. Bai, *Pybullet, a python module for physics simulation for games, robotics and machine learning*, `http://pybullet.org`, 2016–2019.

[38] D. Kalashnikov, A. Irpan, P. Pastor, J. Ibarz, A. Herzog, E. Jang, D. Quillen, E. Holly, M. Kalakrishnan, V. Vanhoucke, *et al.*, "Qt-opt: Scalable deep reinforcement learning for vision-based robotic manipulation," *arXiv preprint arXiv:1806.10293*, 2018.

[39] F. Xia, A. R. Zamir, Z. He, A. Sax, J. Malik, and S. Savarese, "Gibson env: Real-world perception for embodied agents," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2018, pp. 9068–9079.

[40] D. Xu, S. Nair, Y. Zhu, J. Gao, A. Garg, L. Fei-Fei, and S. Savarese, "Neural task programming: Learning to generalize across hierarchical tasks," in *2018 IEEE International Conference on Robotics and Automation (ICRA)*, IEEE, 2018, pp. 1–8.

A Bullet Simulation Environment

The original simulation environment was implemented in MATLAB and Simulink, however, some limitations in this setup encouraged the reimplementation in a new environment. First, several numerical issues were observed. While a foot is in contact with the ground, a very small time step is required to ensure numerical stability in the result. This also means that the performance is very poor, requiring a significant amount of time to complete the optimization (each optimization run required several weeks of run time to converge). Finally, reproducibility of results was quite low, especially at higher speeds: the final parameter set produced by the optimization was often not completely stable.

Given these issues, a new simulator was desired. The following criteria were considered in making the selection of physics engine and simulation environment:

- The physics engine must have a baked-in collision detection algorithm, this being the source of the majority of the performance loss
- The physics engine must provide all necessary functionality (i.e. torque/pd control of joints, joint measurements)
- The physics engine should produce consistent, realistic results
- The simulation environment should be easy to develop and debug
- The simulation environment should be represented in existing robotics literature

Several comparisons of different physics engines were consulted to help select the most suitable engine [34]–[36] – some summary figures are shown in appendix A. While Simulink is not included in these comparisons, the limitations described above mean it would not be suitable for use in this project. It is clear from these comparisons that no one openly available physics engine is superior to the others in every regard, so the simulation environment associated with each physics engine must also be considered.

In this regard, PyBullet [37] emerges as the clear favourite. With a Python binding, PyBullet is exceptional easy to install, develop and debug. It can be used in

tandem with many powerful Python packages as a result, including NumPy, SciPy and PyTorch, diminishing many of downsides of migrating from MATLAB. Additionally, several notable papers have been released recently using PyBullet as their simulation environment [38]–[40]. Thus, PyBullet and, by extension, Bullet are chosen to create the new simulation environment.

Three constraint solvers are available within PyBullet; their performance in terms of deviation from the sagittal plane and average velocity is shown in appendix A. Based primarily on fig. 21a, the PGS solver was selected to run the optimization trial. Additionally, a simulation time step of 1 ms (1 kHz) was found to provide a reasonable trade-off between simulation accuracy and performance.

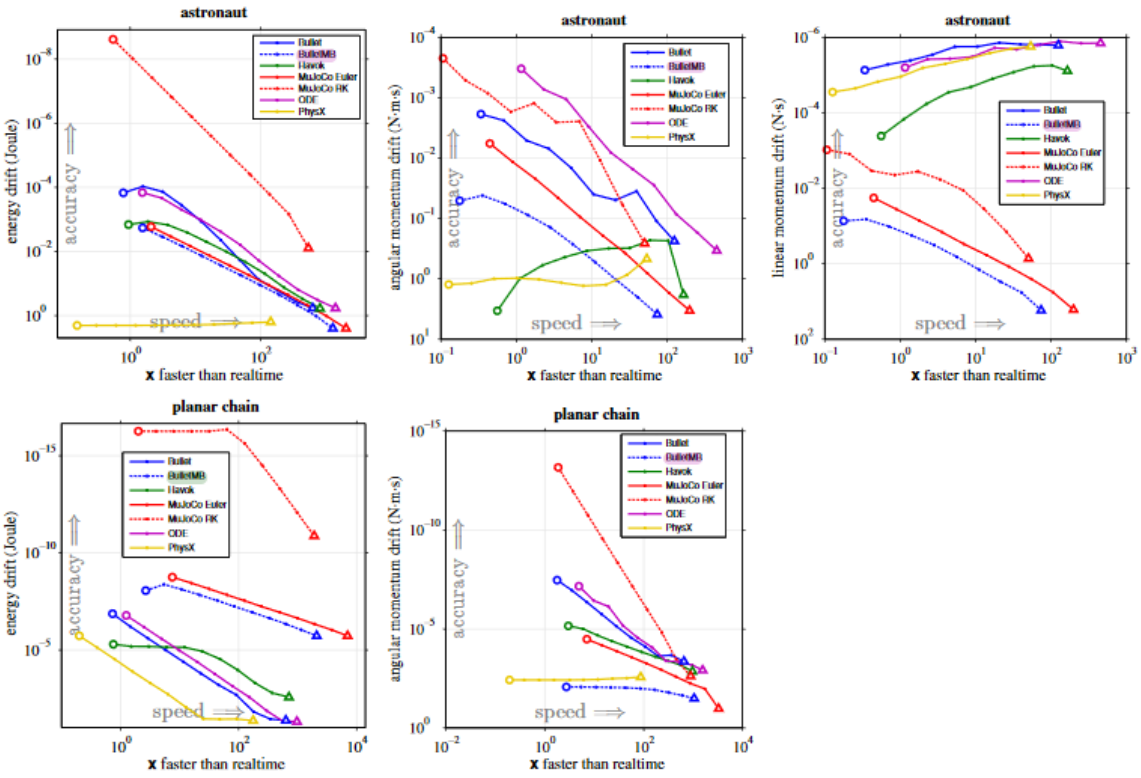


Figure 18: Error in energy and momentum for two different tests (humanoid in zero gravity and planar chain) for a variety of physics engines [35]. In each test, the bullet physics engine maintains comparable or above average performance relative to other physics engines.

	RaiSim	Bullet	ODE	MuJoCo	DART
Rolling	++	+++	-	+	-
Bouncing	++++	++	+++	-	+
666	+++	+	++	+	+
Elastic 666	++++	++	+++	-	+
ANYmal PD	+++++	+++	+	++++	++
ANYmal Momentum	+++	++	+++++	++++ (RK4) ++ (Euler)	+
ANYmal Energy	++++	+++	++	+++++ (RK4) +++ (Euler)	+

Figure 19: Qualitative performance of various physics engines in sample tasks, as determined by creators of RaiSim [34]. Overall, bullet maintains performance comparable to other physics engines.

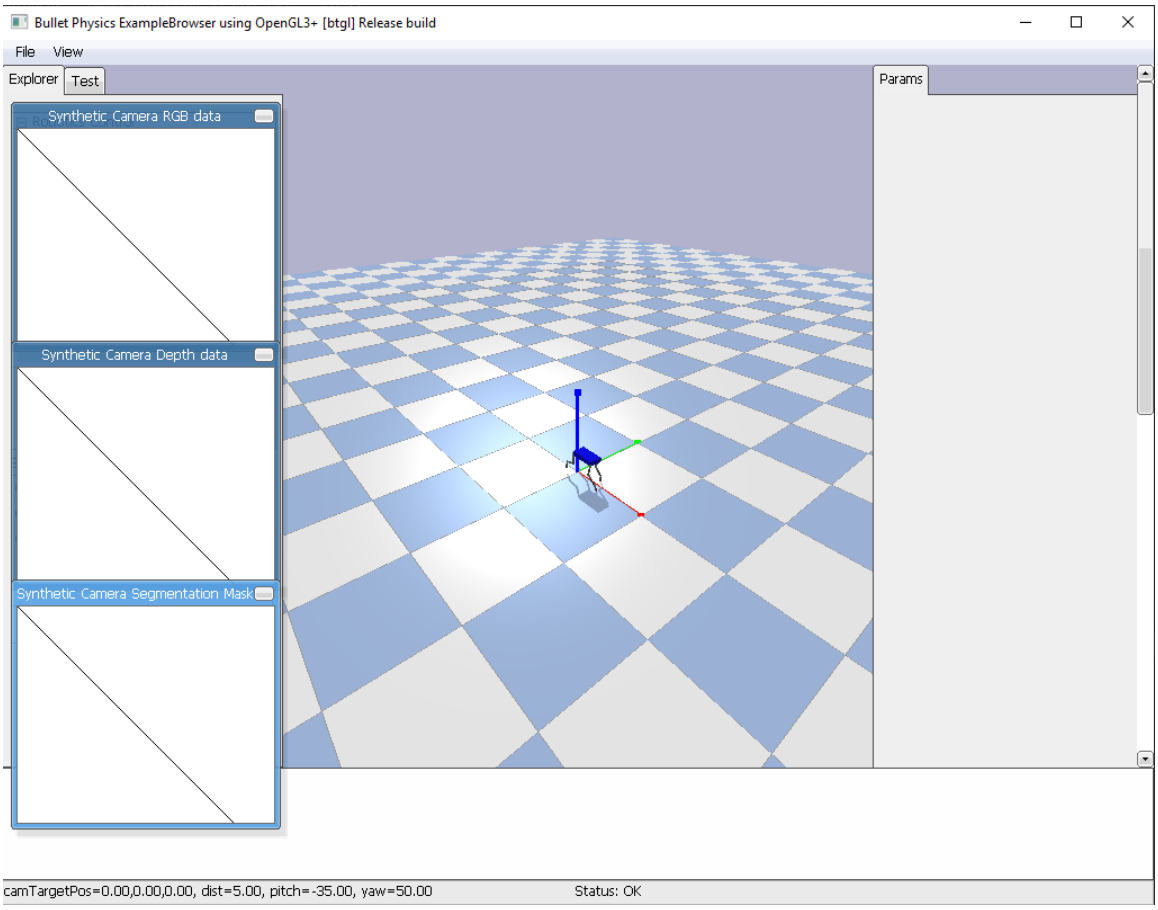
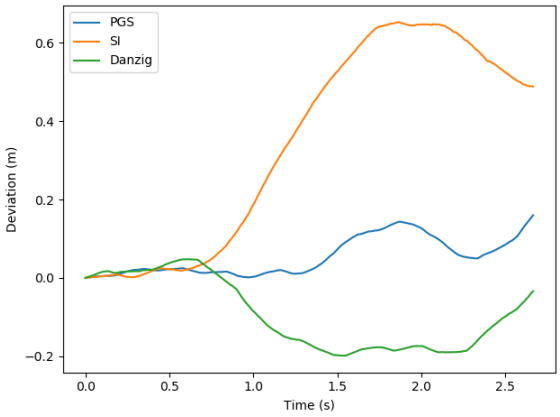
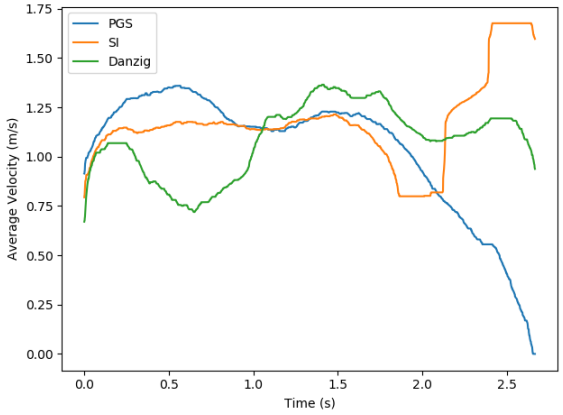


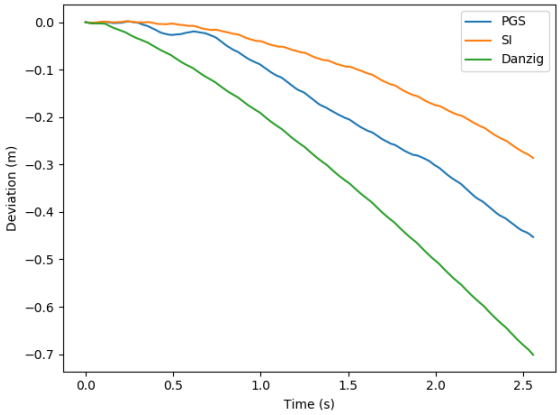
Figure 20: Example simulation window produced by PyBullet.



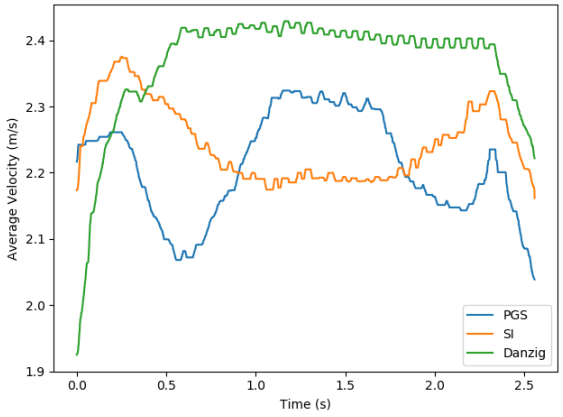
(a) Horizontal deviation from sagittal plane for unoptimized parameter set.



(b) Average forward velocity over half-second interval for unoptimized parameter set.



(c) Horizontal deviation from sagittal plane for optimized parameter set.



(d) Average forward velocity over half-second interval for optimized parameter set.

Figure 21: Comparison of available constraint solvers in PyBullet in terms of horizontal deviation and average velocity. Figures 21a and 21b show the results for an unoptimized parameter set at $1.5\,\mathrm{m\,s^{-1}}$ while figs. 21c and 21d show the results for an optimized parameter set at $2.5\,\mathrm{m\,s^{-1}}$.