Classical and Learned Control Strategies for CG Walker with Stochasticity

A final report for Underactuated Robotics: 16-384

Shuoqi Chen, Ben Freed, Tianyu Wang

Dec. 14, 2018

Table of Content

Abstract

Traditional controllers, such as the linear quadratic regulator, perform well on many systems; however, if the dynamics of the system are nonlinear, such a control strategy is not globally optimal. If the system state is perturbed sufficiently far from the desired state, control can be poor, or worse yet, the system can become unstable. For particularly nonlinear systems, this can result in small basins of attraction. In contrast, neural networks trained via reinforcement learning (RL) can approximate globally-optimal policies for highly nonlinear systems. Once the neural network is trained, computation of the policy is efficient, making them well-suited to real-time control. The primary drawbacks to RL policies is the heavy computational burden during training, and the fact that training does not always converge to a good policy. We propose a novel, hybrid controller design which combines the output of a PD controller and a neural network to help mitigate these issues. We demonstrate greatly improved learning stability compared to a pure neural network, trained to control a compass-gate walker on rough terrain. We also demonstrate the ability of the hybrid-controlled walker to walk larger distances successfully compared to either a PD-controlled or pure neural network-controlled walker.

Introduction and Innovation

We hypothesize that by combining a PD controller with a neural network, we can combine many of the benefits of the two techniques. In this case, "combine" means the control input at a given time step is a weighted sum of the PD controller and neural network policy. We think that the PD controller should act as a prior to help guide neural network policy exploration into regions of the state space in which more optimal trajectories lie, thus allowing more efficient learning. If this hypothesis is correct, we expect to see more optimal control of a dynamic system compared to a PD controller, and more efficient (fewer experiences needed to learn a good policy) or more stable learning (more monotonic increase in performance) compared to a neural network alone.

The system model we used throughout our development is a compass gait walker model, although other more complicated systems could employ methods similar to those described in the methodology sections. The objective function we chose was the distance the walker could walk without failure in a given period of time. We added environmental stochasticity by simulating the walker on an uneven terrain with ground heights randomly sampled from a truncated normal distribution at each step, not visible to the walker. Such stochasticity ensures that the system is occasionally perturbed outside the basin of attraction of the traditional controller. For our traditional controllers, we used both the P and PD controller, similar to the

ones implemented by Tedrake's group [4, 6]. To train the neural network (NN) control policy, we used evolutionary strategies outlined in [5], which is discussed in more detail in the methodology section. The same strategy is then used in combination with the PD controller to form a hybrid controller.

Methodology

The Actuated Compass Gait (CG) Walker Model

We used the simplest compass gait walker model from Garcia et al. [1] where the two legs are massless and there are only three point masses at the two feet and the hip (see Figure 1). There are four state variables we used to describe the system: the angle between the stance leg and the normal direction with respect to the ground $\boldsymbol{\theta}$, the angle between the stance leg and the swing leg $\boldsymbol{\phi}$, the absolute velocity of stance leg $\dot{\boldsymbol{\theta}}$, and the relative velocity of the swing leg $\dot{\boldsymbol{\phi}}$.

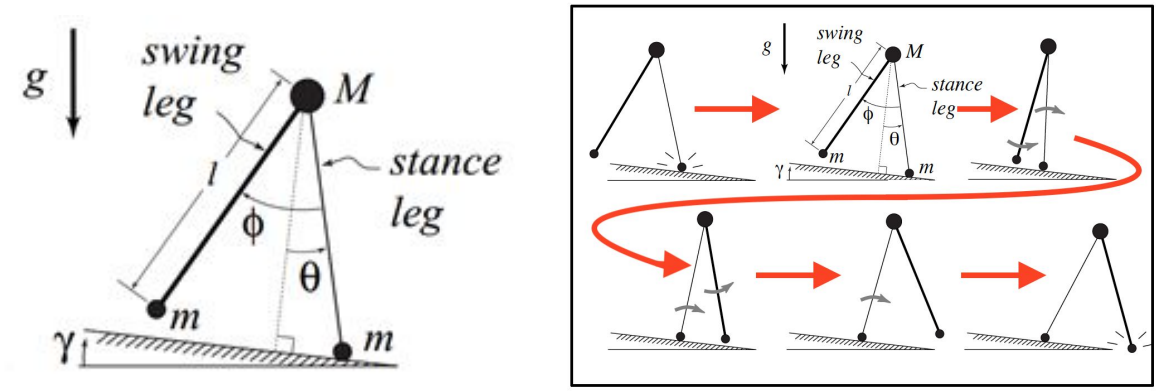


Figure 1) (a) The simplest compass gait walker model [1]. $\boldsymbol{\theta}$ is the angle between the stance leg and the normal direction with respect to the ground. $\boldsymbol{\phi}$ is the angle between the stance leg and the swing leg. $\dot{\boldsymbol{\theta}}$ is the absolute velocity of stance leg. $\dot{\boldsymbol{\phi}}$ is the relative velocity of the swing leg. M is a point mass at the hip. m is the point mass at feet. l is the length of legs. $\boldsymbol{\gamma}$ is the slope of the ground. (b) The compass gait walker motion in a complete gait cycle.

Two kinds of actuation are employed. One is the hip torque $\boldsymbol{\tau}$, which mainly contributes to the relative velocity of the swing leg. All the controllers we discussed in this work are all control strategies that regulate this input torque. Another one is the toe-off impulse P [2], which is just applied at each heel-strike for compensating part of the energy loss due to the ground collision. We set this toe-off impulse to be a constant which makes sure the walker can maintain a stable passive gait. Assuming that the mass on feet is much smaller than the mass on the hip by taking the ratio of m to M as zero, we can get an irreducibly model for the compass gait walker.

$$\dot{x} = \begin{bmatrix} \dot{\theta} \\ \ddot{\theta} \\ \dot{\phi} \\ \ddot{\phi} \end{bmatrix} = \begin{bmatrix} \dot{\theta} \\ \sin(\theta - \gamma) \\ \dot{\phi} \\ \sin(\theta - \gamma) + \sin\phi[\dot{\theta}^2 - \cos(\theta - \gamma)] - \tau \end{bmatrix} \tag{1}$$

Staring from an initial condition, the differential equation (1) can be integrated forward in time, until a heel-strike occurs. We developed two types of conditions to check ground collisions (guard equations). The first one is the angle guard equation,

$$\phi - 2\theta = 0 \tag{2}$$

This guard equation is quite straightforward since two angles are state variables which can be acquired directly. However, if we introduce the stochasticity into the terrain, which is a Gaussian distribution random noise of the height of the ground, it would be hard to transform the change in the height of the ground into an angle change that can be used in this guard equation. A guard equation based on relative distance between the swing foot and the height of the ground is developed as,

$$\Delta h = h(\text{the swing foot}) - h(\text{ground}) = 0, \qquad \text{when} \quad \phi \neq 0 \tag{3}$$

Notice that Δh will equal to zero when the walker reaches the vertical position where two legs overlap, i.e., $\boldsymbol{\phi} = 0$. That is why we need the specific constraint on $\boldsymbol{\phi}$. Compared with the angle guard equation (2), the distance guard equation (3) is computationally expensive since we need to calculate the height of the swing foot in each time step during the integration. However, it proved to be superior in environments with stochastic terrain.

The heel-strike with the ground is considered purely inelastic. The instantaneous state transitions can be derived from the conservation of the angular momentum [3]. As discussed before, a toe-off impulse is applied right before the collision takes place [1]. The state transitions with the constant toe-off impulse are

$$\begin{bmatrix} \theta \\ \dot{\theta} \\ \phi \\ \dot{\phi} \end{bmatrix}^+ = \begin{bmatrix} -1 & 0 & 0 & 0 \\ 0 & \cos 2\theta & 0 & 0 \\ -2 & 0 & 0 & 0 \\ 0 & \cos 2\theta(1 - \cos 2\theta) & 0 & 0 \end{bmatrix} \begin{bmatrix} \theta \\ \dot{\theta} \\ \phi \\ \dot{\phi} \end{bmatrix}^- + \begin{bmatrix} 0 \\ \sin 2\theta \\ 0 \\ \sin 2\theta(1 - \cos 2\theta) \end{bmatrix} P. \tag{4}$$

In this project, we modeled two types of stochasticity following the method by Byl and Tedrake [4]. One of them is the stochasticity in the terrain height, denoted as δ. The δ is sampled from a gaussian distribution with $\boldsymbol{\mu} = 0$ and $\boldsymbol{\sigma} = 1$. However, we specifically chose the sampled δ with magnitude no more than 1% of the walker's leg length. Other than the terrain noise, we also explored the option to add noise to the intra-step dynamics of the walker. This noise can be seen as actuator noise in the hip torque. It represents an error between the desired and actual hip

torque that is input to the system at each time step. The magnitude of this noise and capped by 10% of the input, to the state $\ddot{\phi}$.

Before designing and implementing our NN controller to our compass gait walker, we first explored some classical control strategies, namely P and PD control. The implementation of these two classical controllers serves two purposes. 1) They serve as a basis of comparison for the neural network controller, and 2) They can be combined with the neural network control input in the form of a weighted sum to form a hybrid controller, and should serve as a prior to help guide policy exploration.

In the results section, we demonstrate the performance of our novel hybrid controller and compare it to the classical controllers and pure NN controller on the CG walker model with stochasticity.

Classical Proportional Controller (P controller)

A basic control strategy for the compass gait walker is to apply a hip torque according to a feedback control signal that is linear in the angle at the hip between the swing and stance legs:

$$\tau = -K_p(\phi + 2\alpha), \tag{5}$$

where K_p is a positive proportional gain, and -2α is the inter-leg angle at the collision corresponding to the pre-set desired step length. This P controller is turned on all the time, acting like a spring with stiffness constant K_p that always pulls the swing leg towards the desired position.

Classical Proportional-Derivative Controller (PD controller)

We also implemented the PD controller presented by Byl et al. in [3]. The PD control strategy is to regulate both the inter-leg angle and the relative velocity of the swing leg,

$$\tau = -K_p(\phi + 2\alpha) - K_d(\dot{\phi} - 0), \tag{6}$$

where K_p and K_d are positive proportional gain and positive derivative gain, respectively. The PD controller is only turned on once the swing foot is at vertical position, so as to use only the potential energy for the downs swing, with the controller input being added during the up swing. This PD controller has already been proven to be capable of dealing with the uncertainty of the tough terrain.

Neural Network Controller

Architecture: The network architecture we chose for our pure neural network controller was fully connected with a single hidden layer with 50 neurons. The hidden layer activation was a sigmoid, while the output activation was linear. The output activation was chosen to be linear because we did not want to impose bounds on the control input to the walker; however, tanh or sigmoid activations could have been chosen if our system had actuator constraints.

System Design: During evaluation of the network, the state vector, augmented with the system hamiltonian (which is purely a function of the state), was fed into the neural network at each time step. The network then output a control signal, which was then fed into the system simulation (Figure 2). The reward was set to be the distance the walker walked in a fixed period of time, in our case 50 seconds. During training, the rewards from many sets of perturbed parameters was fed into a gradient estimator. The estimator of the gradient was then fed into an optimizer, which computed a parameter update so as to increase reward in future trials (see subsection on network training).

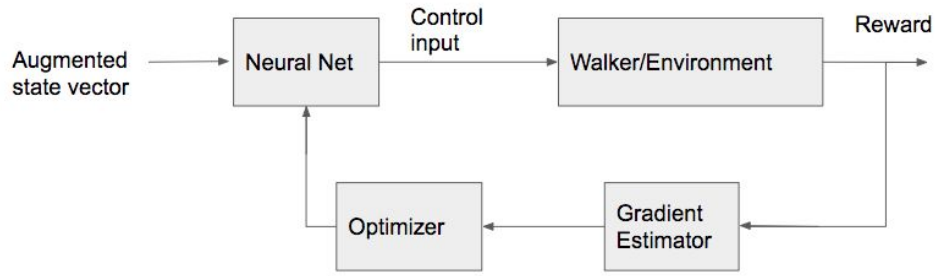


Figure 2) Neural network-controlled system diagram. At each time step, the augmented state vector was fed into the neural network, which output a control signal. The control signal was fed into the walker simulation, and a reward was computed. Rewards from sets of perturbed parameters was used to compute a stochastic gradient estimate, which the optimizer then used to compute an update to the network parameters.

Training: The training algorithm we implemented was similar to the one outlined in [5]. During each training cycle, a set of parameter perturbation vectors $\{\boldsymbol{\varepsilon}_i\}_{i=1}^{n}$ (in this case n=100) was sampled from a zero-mean, unit variance distribution. Each perturbation vector was the scaled by hyperparameter $\boldsymbol{\sigma}$ (in this case 10^{-1}), and added to the current parameter vector $\boldsymbol{\theta}$, creating a perturbed parameter vector. These perturbed parameter vectors were each evaluated in the walking simulation, yielding a set of rewards, $\{F_i\}_{i=1}^{n}$. Our reward function was distance traveled in a fixed period of time. A stochastic estimate of the cost function gradient with respect to the network parameters was computed via the following formula:

$$\nabla_\theta J(\theta) = \frac{1}{n\sigma} \sum_{i=1}^{n} F_i \epsilon_i \qquad (7)$$

Where the cost function J($\boldsymbol{\theta}$) is simply the expected value of reward given the parameter vector $\boldsymbol{\theta}$. Pseudocode for the training procedure is given by algorithm 1, as presented in [5]:

Algorithm 1 Evolution Strategies

Input: Learning rate α, noise standard deviation σ, initial policy parameters θ_0
for $t = 0, 1, 2, \ldots$ **do**
  Sample $\epsilon_1, \ldots \epsilon_n \sim \mathcal{N}(0, I)$
  Compute returns $F_i = F(\theta_t + \sigma\epsilon_i)$ for $i = 1, \ldots, n$
  Set $\theta_{t+1} \leftarrow \theta_t + \alpha \frac{1}{n\sigma} \sum_{i=1}^{n} F_i \epsilon_i$
end for

We experimented with several modifications to this learning algorithm in hopes of achieving more stable training. The modification we found most successful was using an exponentially-decaying step size $\boldsymbol{\alpha}$ rather than a fixed step size as suggested by algorithm 1. This allowed the training to take sizeable steps initially, while making convergence at the end of training more stable. We additionally experimented with other optimizers beyond traditional gradient descent, such as Adam [8], which is state-of-the-art in machine learning, however this was found to have little to no positive impact on training for our problem. Perhaps with more hyperparameter tuning, Adam could have been made to yield favorable results. We also experimented with subtracting a baseline from empirical rewards, where the chosen baseline was the average score of the unperturbed parameters, however this was found to have little to no positive impact on training.

Hybrid Neural Network/Classical Controller

The hybrid controller design and training procedure was very similar to the pure neural network controller, with 1 major modification: rather than simply taking the neural net output as the control input to the walker, the control input was the sum of the classical (PD) controller and the neural network output, multiplied by a small weight (in practice, a weight of 0.1 worked well) (Figure 3). The reasoning behind this controller design was as follows: because the PD controller is already known to work well for controlling the walker system, it should be easier for the neural network to learn to output a "correction term" to the classical controller to make it more optimal. This is because if the walker starts the training process with an already good policy, it will be able to explore the region of the state space in which better trajectories lie more quickly, and make more rapid improvement. The PD controller in this context can thus be thought of as a sort of prior, helping to guide the learning process towards more optimal policies. The weighting is used to prevent the untrained neural network from exerting an overly large control authority on the system initially, such that most of the control input is from the PD controller, and the untrained net effectively acts as a small amount of additive noise in the control signal. Because the output of the net is unbounded and the PD controller is a

deterministic function of the state, the network should be able to eventually learn to "transform" the control input into whatever it found to be most optimal, so incorporating the PD controller into the system should not place any theoretical upper bound on ultimate performance.

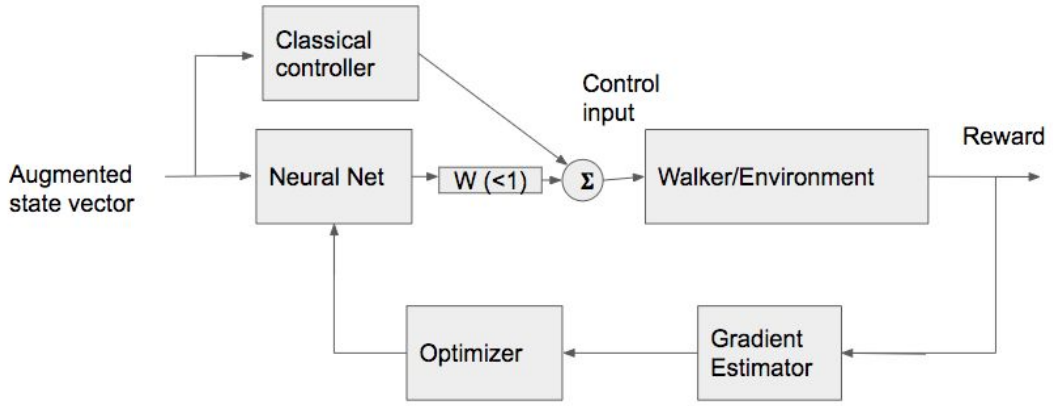
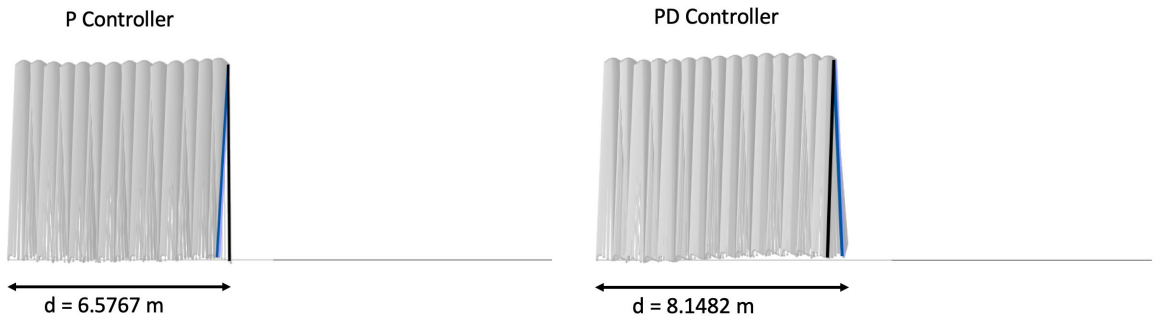


Figure 3) Hybrid Controller System Diagram. The control input to the walker system is computed as the sum of the classical controller output and the weighted neural network output. We found a weight of 0.1 to work well empirically.

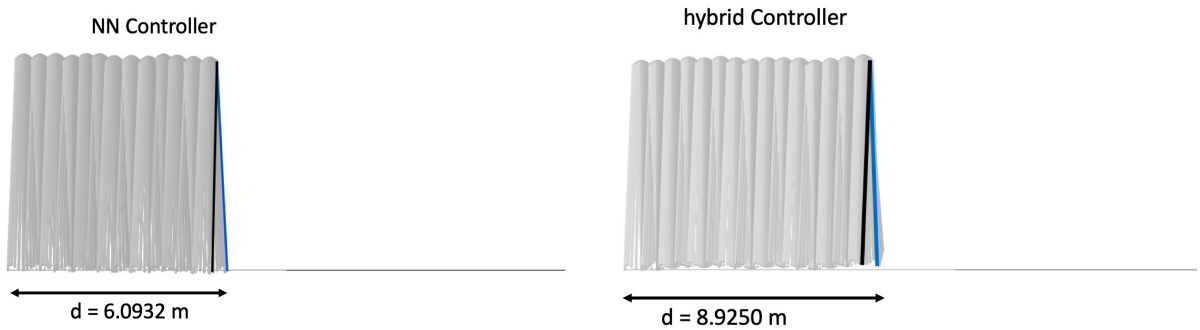
Training of the neural network in the hybrid controller was carried out in the same manner as for the pure neural network controller, with the classical controller input simply considered to be an additional component of the environment.

Results

Simulation behavior analysis



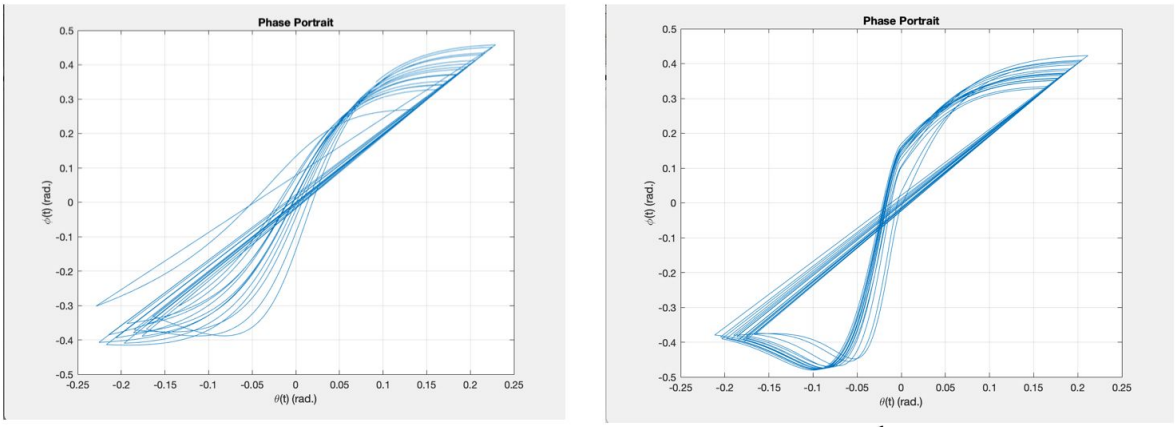
(a) (b)



(c) (d)

Figure 4) The trace plot and average walking distance (over 1000 trials) of the four different controllers successfully traversing a unknown bumpy terrain with terrain height $|\delta| \leq 1\%$ of the walker's leg length. Terminal time = 50 secs. (a) P controller (b) PD controller (c) NN controller (d) hybrid controller. The average walking distance over 1000 trials are indicated as d on the plots.

We performed the walking simulation for each of four controllers that we mentioned before: P controller, PD controller, NN controller and the hybrid controller. The simulation was designed to examine the average walking distance of the CG walker on a unknown bumpy terrain with terrain noise $|\delta| \leq 1\%$ of the walker's leg length within a given time span T = 50 secs. A simulation snapshot of each controller are shown in the Figure 4. The hybrid controller produced longest average walking distance.



a b

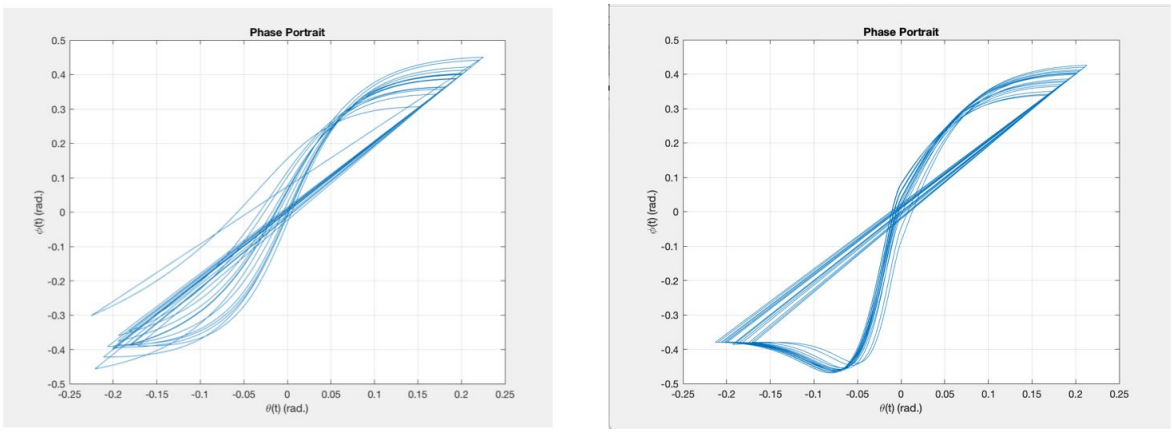


Figure 5) The phase plots of stance leg over multiple gait cycles, the set of initial states chosen to be represented here are: angle of rotation $\boldsymbol{\theta}$ *(x-axis) and angle of inner-leg swing* $\boldsymbol{\phi}$ *(y-axis). (a) P controller (b) PD controller (c) NN controller (d) hybrid controller. Here, the x axis specifies the* $\boldsymbol{\theta}$*, and y axis species* $\boldsymbol{\phi}$*. For each walking cycle, the compass gait walker started from the set of states on the top right corner to the bottom left corner.*

The performances of the four controllers can be analyzed from the lens of returned states, defined as the set of initial states immediately after each heel-strike collision. Due to terrain stochasticity and the differences in control policies, in each case the walker reaches a variable terminal state at the end of each gait cycle and transitioned back to different returned state assuming a constant toe-off impulse transport. From the phase plots for each of the controllers as shown in Figure 5, we can see the variances of the return states are larger than those of PD controller and hybrid controller, and correspondingly, the P controller and NN controller are more prone to failure.

Walking distance

The following table compares the average walking distance of the compass gait walking using the four different controllers averaged over 1000 trials. The walking distance is defined as maximum hip joint displacement in the walker's moving direction before the walker falls or before the terminal time T is reached. In each trial, the walker traverses a zero slope, one-directional terrain with terrain noise $\delta \leq 1\%$ of the leg lengths. The form of the controller output and the parameters used are provided in the appendix.

Table 1: average walking distance of the four controllers over the stochastic terrain (T = 50 secs; $\delta \leq$ *1% leg length)*

	P controller	**PD controller**	**NN controller**	**Hybrid controller**

Ave. distance (m) over 1000 trials	6.58	8.15	6.09	8.93
Ave. steps taken over 1000 trials	13.21	14.94	14.30	15.00

The results show that the pure neural network controller (without the PD controller) performs poorly, yielding the shortest average walking distance and smallest average number of successful steps taken per trial. On the other hand, in combination with the PD controller, the hybrid controller is able to provide a more optimal control strategy, yielding the longest average walking distance and steps taken.

Learning Stability of the Pure Neural Network and Hybrid Controllers

Despite experimenting with a large set of training hyperparameters, various baselines, and gradient optimization techniques, we were unable to achieve stable learning using the pure neural network controller. Even with a conservative learning rate, the average distance walked by the walker oscillated abruptly between around 0 and around 2 meters. However, learning progressed in a quite stable manner for the hybrid controller (Figure 7). We speculate as to why this is the case in the discussion section.

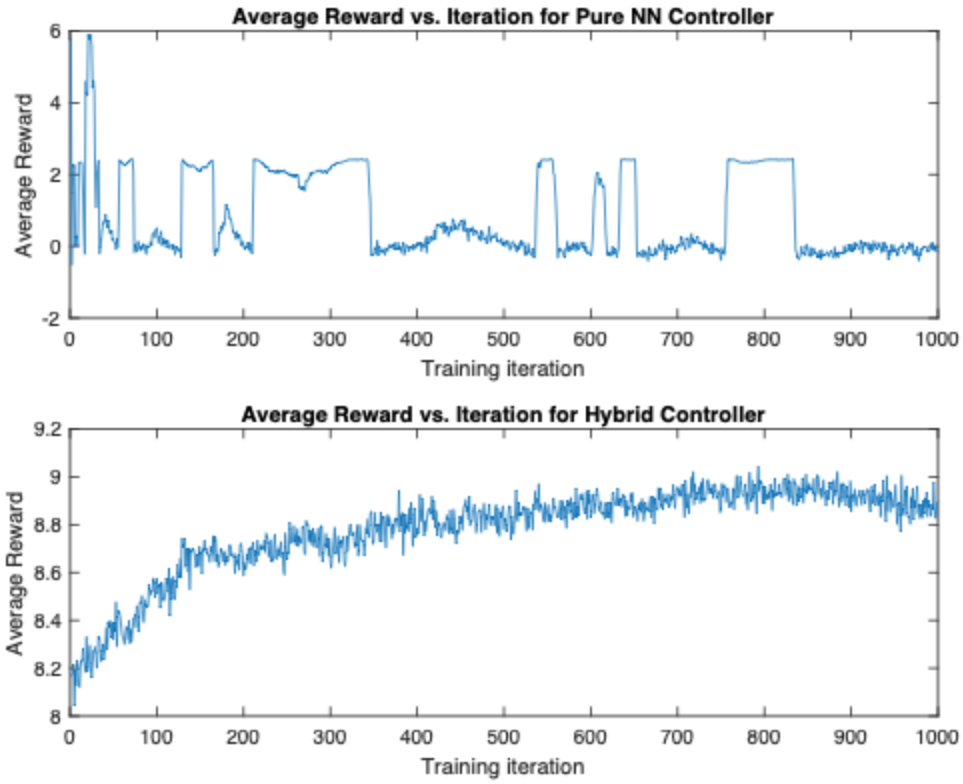


Figure 6) Learning curves of pure neural network and hybrid controllers. Plotted above is reward achieved (distance walked by walker in 50 seconds) by the pure neural network and hybrid controllers, averaged over 100 trials, at each iteration of learning. Training progresses stably for the hybrid controller, with average reward peaking at 8.9250 meters. Training was unstable for the pure neural network controller even with a modest learning rate, as indicated by the large jumps in performance and generally low reward. Average reward for the pure neural network controller peaked very early on in the training, at 6.0932 meters.

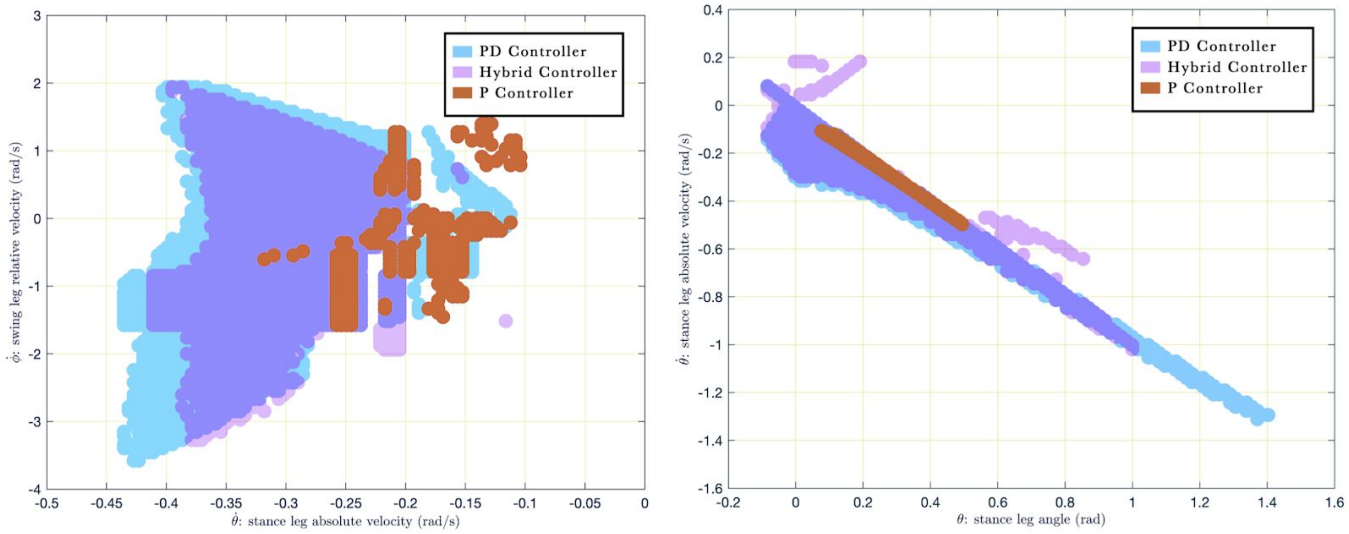
Discussion

Without using the traditional controller as the seed, the control policy produced by the neural network alone was not able to achieve performance superior to P control in the alloted 1000 epochs of training. However, when using the policy from the PD controller as part of its training input, the neural network is able to quickly learn a better controller policy that enables the compass gait walker to travel a longer distance along a terrain of similar setup. This supports our initial hypothesis that a learning-based controller should perform better than a classical controller, and that the hybrid controller does indeed learn more stably than the pure NN controller.

However, the questions that accompanies this promising result is: are the learned controllers, more specifically the hybrid controller, able to perform well in situations previously unseen in their training? How do we know if the hybrid controller is robust and versatile? To provide insights into those questions, we devised two simple tests to help evaluate generality of the learned control strategy.

First, we evaluate the robustness of the hybrid controller by examining the size of its basin of attraction (BOA) in comparison with those of the classical controllers. It is well established that walking systems in deterministic, noise-free environments have fixed points and BOA is introduced as the region of the aforementioned fixed point that, if the walker system dynamics starts from a set of initial states that are within the BOA, the system becomes truly stable. Therefore, basin of attraction is an appropriate evaluation metric to assess the stability of the system.

However, given enough time, stochastic dynamic systems are guaranteed to push the walker out of these deterministic stable regions [3], therefore it is impossible to categorically define bounded regions in which the compass gait walker is stable for indefinitely. To address this limitation, we introduce a modified stability criteria in which a given set of initial condition is said to be in the walker's basin of attraction if the walker can be initialized according to these conditions, and proceed to walk without fail until a terminal time T in a noise-free environment. We set T = 50s for the following discussions. The BOA plots are generated using value iteration.



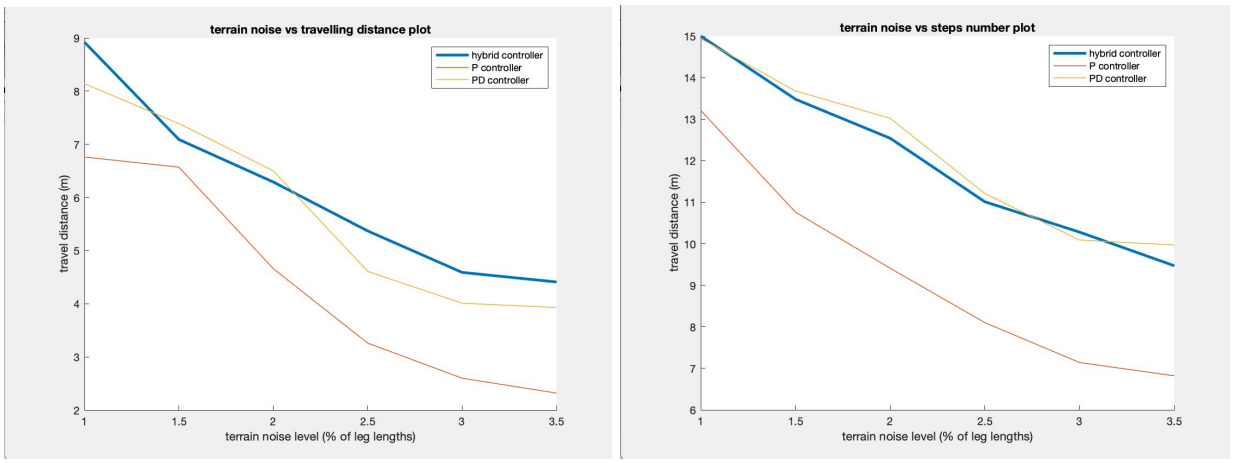
(a) (b)

Figure 7) Basin of attraction comparison between the hybrid controller and the classical controllers. The basin of attraction of the four controllers in two seperate sets of comparison (set (a): θ vs $\dot{\theta}$; set (b): $\dot{\theta}$

vs $\dot{\phi}$). θ *is the stance foot rotation angle;* $\dot{\theta}$ *is the stance foot rotation angular velocity,* $\dot{\phi}$ *is the angular velocity of the inner-leg swing angle*

Since the systems dynamics is parametrized using four states, to visualize the BOA in *2D* we choose two sets of initial state pairs. The θ *vs* $\dot{\theta}$ pair demonstrates whether the walker is approximately stable given different initial setup of the stance leg, and the $\dot{\theta}$ *vs* $\dot{\phi}$ pair demonstrates whether the walker is approximately stable when it starts from the same fixed position but with different initial velocities. Interestingly, although the BOA of the hybrid controller is larger than the P controller in either case, it is smaller than that of the PD controller. This is probably due to the fact that in the NN training phase, the walker always starts from a single set of initial condition that resides within the BOA, so the neural network never explores the effect of different initial conditions on its objective. Therefore, it is reasonable to see the hybrid controller generates a policy that increase its objective performance at the cost of its stability.

Second, we evaluate the versatility of the hybrid controller by examining its average traveling distance and average steps taken in different level of terrain noise.



(a) (b)

Figure 8). Performance of the hybrid controller in comparison with P and PD controller given different levels of terrain noise δ*. In each graph, the performance line of the hybrid controller is colored blue (a) average traveling distances (over 1000 trials) given different level of* δ*. (b) average steps taken before falling or before terminal time T = 50 is reached (over 1000 trials) given different level of* δ*.*

It is within our expectation that the average traveling distance of the hybrid controller would decrease proportionally with respect to the level of terrain noise, as higher noises will result in more instances of failure. The hybrid controller outperforms P controller in both cases and performs relatively better than the PD controller in the cases of higher terrain noises. As shown figure 4), the average steps taken before the walker falls or before the terminal time T is reached

is pretty similar between the hybrid controller and the PD controller, but the hybrid controller achieves a longer travelling distance in general. This is possibly caused by the higher torque signals generated by the hybrid controller which give rise to bigger step lengths. Since we did not set an upper bound to the magnitude of the torque signal, it is very likely that the hybrid controller takes advantage of this freedom and generates a policy that increases the average traveling lengths.

To better understand why training for the pure neural network was less stable than the hybrid controller, we inspected the norm of the parameters of the output layer of both neural networks (fig. 9). The weights of the hybrid controller were multiplied by the weight 0.1 in fig.9, as this offers a more apples-to-apples comparison of network parameters because in our case, multiplying the output of the net by 0.1 is equivalent to multiplying the weights in the final layer by 0.1. Based on our observations, it appears that a likely reason for the learning instability of the pure neural network was the presence of a strong local optimum or saddle point in the region in which output layer weights were tiny. This is likely because the walker actually can walk fairly far, an average of 5.5289 meters with no control input, but is very sensitive to noise when no feedback controller is present, causing slight non-zero input form the neural network to cause failure. This local optimum was likely very difficult to escape, because it appears for the pure neural network controller, performance is incredibly sensitive to slight changes in parameters; compared to the hybrid controller, variation in the norm of output layer weights did not vary that much during training, however average performance changed very abruptly and by large amounts, oscillating between 0 and around 2 meters within the span of only 1 learning update. A possible explanation for this sensitivity is that as soon as the neural network tried to escape the local optimum it seemed to be trapped in when it was walking 2 meters, the walker would immediately fall (thus walking 0 meters), creating a strong pull back to the local optimum. It is possible that the hybrid controller was able to avoid this problem because even when the neural network was taking poor actions (adding effectively unfavorable noise to the control signal), because it was multiplied by a small weight and coupled with a feedback controller, this noise could likely be partially rejected, effectively making the performance less sensitive to changes in neural network parameters and smoothing the objective function the net was trying to optimize. This likely made learning easier for the hybrid controller.

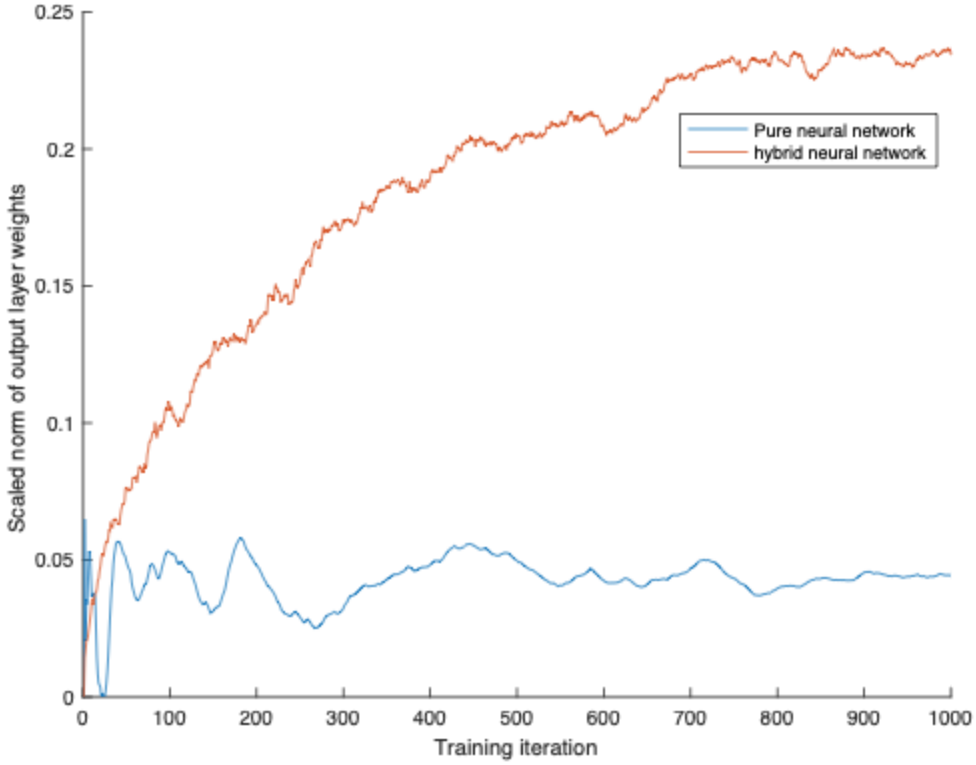


Figure 9) Scaled norm of output layer parameters for the pure neural network and hybrid controllers. The pure neural network weights remain very small as training progresses compared to the hybrid neural network parameters (scaled by the weighting applied to its output), yet performance varies in huge increments (fig 7). One possible explanation for this is that the pure neural network controller is stuck in a local optimum from which it is difficult to escape because even a small perturbation away from it results in the walker immediately failing (walking 0 meters).

Conclusion and future work

We have demonstrated that for a compass gate walker on uneven terrain, a hybrid neural network/PD controller works better than either a neural network or PD controller alone. Additionally, we have demonstrated that incorporating a PD controller improves stability of training, as the PD controller serves as a sort of prior from which the neural network can rapidly improve upon.

We recognize that we cannot guarantee that a global optimum was reached during training, as the training may have prematurely converged to a local optimum. Additionally, we recognize that some performance metrics of the hybrid controller are not as desirable compared to other control control strategies (e.g. basin of attraction).

In future work, we plan to test the generality of our proposed hybrid controller approach by testing it on other dynamic models, such as a kneed-walker or SLIP model bipedal walker, preferably ones with feasible, closed-form but suboptimal control strategies.

We also plan to further explore the robustness and generalizability of learning-based controllers by exploring their ability cope with previously unmodeled stochasticity, such as noise in the state-transition dynamics.

We also plan to investigate more thoroughly why the pure neural network controller failed to train stably and yield a good policy. We began speculating at possible reasons in the discussion section, however this problem warrants further investigation. One possible approach would be to further explore the shape of the objective function around the local optimum the neural net seemed to become trapped in.

Bibliography

[1] Garcia, Mariano, et al. "The simplest walking model: stability, complexity, and scaling." *Journal of biomechanical engineering* 120.2 (1998): 281-288.

[2] Kuo, Arthur D. "Energetics of actively powered locomotion using the simplest walking model." *Journal of biomechanical engineering* 124.1 (2002): 113-120.

[3] McGeer, Tad. "Passive dynamic walking." *The International Journal of Robotics Research* 9.2 (1990): 62-82.

[4] Byl, Katie, and Russ Tedrake. "Metastable walking machines." *The International Journal of Robotics Research* 28.8 (2009): 1040-1064.

[5] Salimans, T., Ho, J., Chen, X., and Sutskever, I. (2017). Evolution Strategies as a Scalable Alternative to Reinforcement Learning. ArXiv e-prints.

[6] Byl, Katie, and Russ Tedrake. “Approximate Optimal Control of the Compass Gait on Rough Terrain.” *2008 IEEE International Conference on Robotics and Automation*, 2008, doi:10.1109/robot.2008.4543376.

[7] Manchester, Ian R., et al. “Regions of Attraction for Hybrid Limit Cycles of Walking Robots.” *IFAC Proceedings Volumes*, vol. 44, no. 1, 2011, pp. 5801–5806., doi:10.3182/20110828-6-it-1002.03069.

[8] P. Kingma and J. Ba. Adam: A Method for Stochastic Optimization. In *International Conference in Learning Representations*, 2015.

Appendix

Code repository: https://github.com/b-freed/ua_robo_final_project