

Two-axis unicycle simulation and control

Nick Stenning
nick@whiteink.com

16th May 2011

Abstract

The unicycle presents a challenging problem in nonlinear control. This report presents a technique in which genetic algorithms are used to search for a suitable controller from a large space of possible control functions, which are encoded as neural networks. The equations of motion of a simplified unicycle are derived and found to be unrewarding in direct analysis. A physical model of the unicycle is developed and verified in simulation, and we show that our approach finds controllers which can successfully stabilise the unicycle, even in the face of significant disturbances. The evolved controllers are effective over a range of different control regimes, and can successfully transition from a standstill to a moving unicycle state. Finally, we address the limitations of our approach and suggest how neurocontrollers of this type can best be integrated into a useful control strategy.

Contents

1	Introduction	2
2	Theory & control approach	3
2.1	Unicycle dynamics	3
2.2	Neural networks	5
2.3	Genetic algorithms	6
2.4	Control system	7
3	Implementation & results	8
3.1	Physical model	8
3.2	Control strategy in 2D	9
3.3	Controller behaviour in 3D	10
3.4	Startup success	12
3.5	Disturbance rejection	13
4	Discussion	13
4.1	Validity of physical model	13
4.2	2D control	14

4.3	3D control	14
4.4	GA and NN observations	15
5	Conclusion	17
	Appendices	19
A	Derivation of equations of motion	19
B	Equations of motion	21
C	Physical parameters	22
D	GA details	22
	D.1 Fitness function	23
	D.2 Mutation operator	23
	D.3 Crossover operator	23

1 Introduction

Unicycles are harder to ride than bicycles: a fact to which anyone who has ridden both will attest. In the language of control engineering, the unicycle is hard to ride because it is unstable — the position in which one rides is energetically unfavourable — and it is also underactuated — the rider has two control inputs to control a six degree-of-freedom system.

This report outlines an unusual approach to the design of an automatic controller for a unicycle. A purely analytic solution to the problem is certainly out of the reach of this author: as will be shown in Section 2.1, the equations of motion of even a simplified unicycle model are highly nonlinear and do not easily yield general solutions. No such solutions will be offered. Instead, inspired by the work of Reil and Husbands [9] on control of bipedal walkers, and combining ideas from several fields, I show how we can use genetic algorithms to search for a suitable controller from a “population” of feed-forward neural networks. The neural networks act as generalised controllers, mapping system state to control response, which can be computationally evolved to find one that serves our purpose. The motivation behind this approach will be discussed in Sections 2.2 – 2.4.

Several previous attempts have been made to stabilise a unicycle, most of which take some linearised approximation of the unicycle equations of motion as their starting point. Vos and von Flotow [10], and Zhao et al. [12], describe gain-scheduled linear controllers which attempt to take advantage of the assumed piecewise linearity of the system. Naveh et al. [7] attempt to improve on the limitations of linear control by fitting a parameterised control law which can be up to quadratic in the state variables. All three groups treat the longitudinal (wheel-plane) and lateral control systems as uncoupled or weakly coupled. In this report, we show that a physical simulation of a unicycle which makes no such assumptions, and with a realistic friction model at the ground-wheel interface, can be stabilised using a neural network. The resulting controllers are stable under perturbation and describe a physically realisable unicycle model.

2 Theory & control approach

2.1 Unicycle dynamics

The equations of motion of a simple unicycle are not difficult to derive once an appropriate set of generalised coordinates has been identified. In what follows, I will describe the unicycle state in terms of four angles (three identifying the orientation of the wheel, one the position of the seatpost) and two position coordinates (identifying the position of the wheel center as projected onto the ground plane along the vertical axis).

Figure 1 illustrates the chosen coordinates: θ , or **roll**, is the angle between the world vertical axis (\hat{Z}) and the line connecting the wheel-ground contact point and wheel center. ϕ , **yaw**, is the angle between \hat{X} and the diameter of the wheel parallel to the plane of the ground. ψ is the rotation angle of the wheel about its own axle, \hat{x} . χ , **pitch**, is the angle of the seatpost about the wheel axle, measured relative to the wheel-local \hat{z} axis. The xyz coordinate system tracks the wheel in roll and yaw, but does not co-rotate. Lastly, coordinates X and Y denote the position of the unicycle on the ground plane as indicated in the figure. We assume the following features of the unicycle:

1. The wheel is modelled as a uniform thin rigid disc, radius r and mass m .
2. A light seatpost of length l connects the wheel axle to a pointlike rider, mass M .
3. No linear slip occurs at the wheel-ground interface.
4. Twist friction is neglected for the time being, but will be discussed in Section 3.1.

We can expect that the general equations of motion will be far from linear. The longitudinal system (in the plane of the wheel) is isomorphic to the classic inverted pendulum problem when $\theta = 0$, but for nonzero roll angles gyroscopic effects will couple the lateral and longitudinal systems (particularly at high wheel and pitch angular rates). For large values of pitch and roll the forces transmitted by the seatpost will also affect both wheel rotation and yaw. This is the key difficulty of unicycle control: since the rider has no direct actuation of the roll angle, in order to correct errors in roll, corrections in both yaw and pitch must be applied, typically quite rapidly. This means that we cannot easily neglect the coupling between the longitudinal and lateral systems if we wish to model a real-world unicyclist.

In Appendix A, the equations of motion for this rigid body system are derived using the Lagrangian method. The degrees of freedom are tightly coupled and equations of motion highly nonlinear, and while small-angle linearisations of roll and pitch are perhaps appropriate, no such approximation can be considered for yaw. By symmetry, we would expect the equations to be independent of the absolute value of ψ , and indeed they are. Furthermore, in the case where motion is constrained to lie within the Z - X plane, we can reduce the system to the

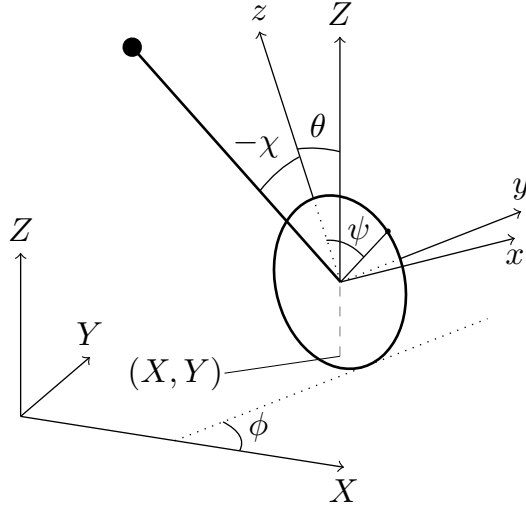


Figure 1: Diagram showing the degrees of freedom of the unicycle system. The unicycle state is specified by the coordinates of the wheel center projected onto the ground plane (X, Y) , and four angles $(\phi, \chi, \theta, \psi)$ corresponding to yaw, pitch, roll and wheel rotation respectively. The minus sign on the χ label indicates that this angle is measured in an anticlockwise sense about the x axis.

equations of a no-slip 2D unicycle¹, confirming our assertion that this system is isomorphic to the inverted pendulum. This provides additional reassurance of the validity of the equations of motion.

The equations, given in Appendix B, are formidably complicated, and reveal no obvious general solutions. Even numerical integration is made difficult by the presence of two unknown Lagrange multipliers, for which initial conditions must be specified. For all this, the model is oversimplified and includes a point-mass rider, no realistic friction model and no means of actuating the yaw degree of freedom. To address these issues, we develop a rigid-body physical model of the unicycle using the *Bullet* physics library [1]. The details of this implementation will be discussed in Section 3.1, after an overview of the control approach.

¹

$$\ddot{\psi} = \frac{D \sin \chi - 2B\ddot{\chi}}{C \cos \chi}$$

$$\ddot{\chi} = \frac{\tau_{\psi} + C\dot{\chi}^2 \sin \chi - 2AD \sin \chi / C \cos \chi}{C \cos \chi - 4AB / C \cos \chi}$$

Here, the four constants refer to the physical parameters of the unicycle. Using symbols defined in Appendix C, $A = I_{ax}/2 + mr^2/2 + Mr^2/2$, $B = Ml^2/2$, $C = Mrl$, and $D = Mgl$. Also included is the nonconservative torque supplied to the wheel, τ_{ψ} , which could include rider input and rolling friction. This system, easily numerically solved, was used in verifying the control approach described in the following paragraphs.

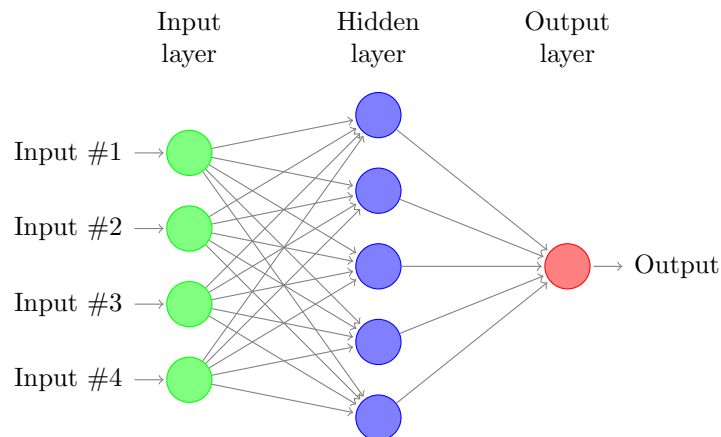


Figure 2: A fully connected 3-layer feed-forward network. Each node in the hidden layer is connected to each node in both the input and output layers. *Image credit: Kjell Magne Fauske.*

2.2 Neural networks

A neural network (NN) is a programming construct that aims to emulate the information processing techniques of biological neurons. In mathematical terms, a NN is a directed graph in which the nodes represent artificial neurons. Each node has some number of inputs and outputs: its neighbours on the graph. Network edges are assigned real-valued weights, and the output of any node is defined by the weighted sum of its inputs and by its transfer or “activation” function $A(x)$, which typically has the form of $\tanh x$, $(1 + e^{-x})^{-1}$ or a similar sigmoid. The output of the i th node, x_i , is given by $A(x)$, the weight matrix w_{ij} , and a bias value for that node, b_i .

$$x_i = A(b_i + \sum_{j \neq i} w_{ij} x_j)$$

Why is such a network useful for our purposes? Because it can be shown that a feed-forward network such as that shown in Figure 2 is a universal function approximator [2, 3]. With a sufficiently large hidden layer, the network can approximate any continuous function of the inputs onto the outputs.

This is the key result, as the design of a control algorithm is the design of a function mapping system state to values of available control inputs. By modifying the number of neurons in the hidden layer, and the weights of the network connections, we can – in principle at least – approximate any continuous control law.

In many applications, an appropriate neural network can be found by directly training the network. A “training set” of inputs are fed into the network, and feedback is pushed back through the network, strengthening connections that led to good output values, and weakening the rest. We will not use this method, known as “back-propagation”, to train our networks, for

reasons which will soon become clear. Instead, a population of randomly-weighted networks is created, and each one is evaluated for its usefulness as a controller. A selection process ensures that only the “fittest” networks survive to the next generation of evaluation. This method is the subject of the next section.

2.3 Genetic algorithms

Genetic algorithms (GA) are a method for finding and optimizing solutions to a problem in a space of possible candidates. For nontrivial problems, such spaces typically have high dimensionality, and, in general, no algorithm can efficiently search them. GAs are most useful when the following criteria are satisfied:

1. No direct optimization of a given candidate is possible. This is usually equivalent to the statement that no solution space gradient information is available.
2. A simple encoding exists for representing a point in the solution space that can be randomly “mutated”, corresponding to a small random step in the space.

There are many variations of GA (see Mitchell and Forrest [5], among others), but the method used here is as follows:

1. Create a population of N candidate solutions, or “individuals”.
2. Evaluate each individual and assign a fitness score.
3. Create the next generation:
 - (a) For some *elitism*, $\eta \in [0, 1]$, pick the $\lfloor \eta N \rfloor$ fittest individuals and add them to the next generation. These individuals are left untouched by the next three steps.
 - (b) Randomly pick a further $N - \lfloor \eta N \rfloor$ individuals from the population: their chance of being picked is proportional to their fitness².
 - (c) For each of the individuals, *crossover* the individual with another individual from the original population with some probability p_c . The second individual is again picked by fitness-proportionate selection. A crossover typically represents some merger of the solutions represented by the two individuals, while the precise method depends on the details of the solution encoding.³
 - (d) For each of the individuals to which the crossover operator was *not* applied, *mutate* the individual. Such a mutation typically corresponds to a small random step in the solution space.
4. Repeat steps 2 and 3 until an individual achieves a suitable fitness score.

²This is known variously as a *roulette-wheel* or *fitness-proportionate* selection. The probability of the i th network (with fitness f_i) being selected is $p_i = f_i / \sum_j f_j$.

³While a detailed discussion of the value of this step is outside the scope of this report, a well designed crossover operation is likely to make a much larger step in solution space than mutation, while ensuring that the expected resulting fitness is larger than if a step of such a size were taken randomly.

To summarize, the GA is parameterised by its population size (N), its elitism (η), its crossover probability (p_c), and three operations: the fitness function, which assigns a fitness score to an individual, and the mutation and crossover operations.

2.4 Control system

Having been introduced to neural networks and genetic algorithms, it is now time to explain why together they form a sensible approach to designing a control system for the unicycle problem.

The unicycle has six degrees of freedom,

$$\{X, Y, \phi, \chi, \theta, \psi\}$$

which with their time derivatives correspond to twelve variables fully specifying the unicycle state. Symmetry considerations allow us to rule out X , Y , and ψ as useful parameters to a control system. Our control inputs are pedal torque, τ_ψ , and a torque associated with the twisting of the rider’s torso about the seatpost, which we will denote τ_r . The task, then, is to design a control function F ,

$$F(\dot{X}, \dot{Y}, \phi, \dot{\phi}, \chi, \dot{\chi}, \theta, \dot{\theta}, \psi) = (\tau_\psi, \tau_r)$$

which results in a stable unicycle. In practice, we will see that only a subset of these parameters are needed in the control function. Indeed, computational feasibility requires that we do not use all the possible inputs.

Many possible functions F can be represented by a feed-forward neural network with input nodes corresponding to the state variables, a hidden layer of a size to be determined experimentally, and two output nodes for the control inputs. Such a network is then evaluated in a tight loop with the physical simulation of the unicycle: at each timestep the state of the unicycle is fed through the NN, and its outputs are in turn fed back into the physical simulation.

We know what long-timescale success looks like: crudely speaking, the unicycle does not fall over. But on short timescales we cannot say whether any given response of the network is “good” or “bad”: that is, given a history of control input to a failed controller, it may not be possible to specify which of its actions contributed to its failure. This means that optimization of the network using back-propagation is not possible. Instead, a GA optimizes a population of networks by searching the space of possible network weightings⁴. Further details of this approach are described by Whitley et al. [11], and Metni [4]. Both show the use of combined GA/NN approaches to the solution of the inverted pendulum problem, and give results that we confirm and extend in studies of a 2D unicycle model.

In summary, a genetic algorithm is used to select a neural network from a large space of such networks. Each network is evaluated on its ability to stabilise the unicycle (for details of this evaluation see Appendix D). We now go on to discuss the success of this approach, and will show that not only are plausible control functions found, the process of finding at least some of them is computationally feasible on a modern consumer laptop.

⁴In principle the GA could also select the network topology. Our GA only modifies network weights in networks of a fixed topology.

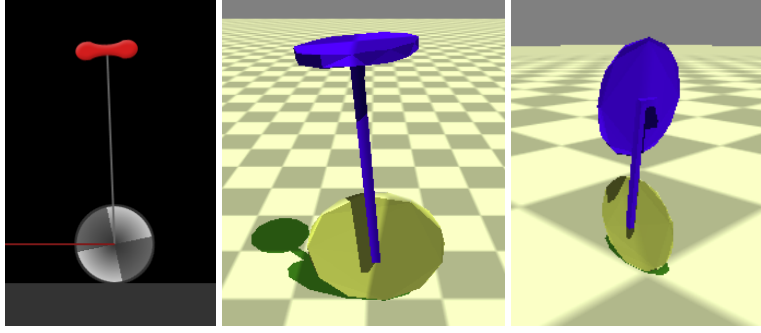


Figure 3: Visualisations of the 2D and 3D unicycle models built for simulation studies. A motorized reaction wheel sits in place of the rider on the 3D model. Most simulations use the middle figure, with a yaw-actuating reaction wheel. The right-hand figure shows a roll-actuating reaction wheel, which has control capabilities fundamentally different from those of a human rider.

3 Implementation & results

The following sections detail the implementation of the unicycle model and some results from evolved controllers. Detailed discussion of the results will follow in Section 4, while commentary here is intended to be explanatory but brief.

3.1 Physical model

A physical model was built using the *Bullet* physics engine, comprising three rigid bodies (wheel, seatpost, reaction wheel) and two rotating constraints connecting them. See Figure 3 for an illustration of this model. Control inputs are torques applied about the wheel and reaction wheel axles. Angular momentum is conserved by applying equal and opposite torques to each side of the constraint.

Twist friction at the wheel-ground interface was modelled as a combination of coulomb and viscous friction, following Naveh et al. [7], giving a friction torque about the Z axis through the wheel of

$$\tau_{\text{fric}} = -W(\gamma_c \text{sgn}(\dot{\phi}) + \gamma_v \dot{\phi})$$

where W is the applied weight of the unicycle on the ground. Accurate calculation of the values of γ_c (with units of m) and γ_v ($\text{m} / \text{rad s}^{-1}$) would have to be performed in a laboratory setting, but a simple model relates these coefficients to the usual linear friction coefficients and the contact area of the wheel. A thin contact area of length $2L$ with even weight distribution leads to $\gamma_c = \frac{\mu_c L}{2}$ and $\gamma_v = \frac{\mu_v L^2}{3}$ where μ_c and μ_v are the usual linear coefficients of friction. Reasonable values of $L = 5 \text{ cm}$, $\mu_c = 0.8$, $\mu_v = 0.2$, give the values of γ_c and γ_v given in Appendix C. Static friction (effective at $\dot{\phi} = 0$) is neglected. The linear slip friction coefficient is set sufficiently high that no linear slip occurs during simulation.

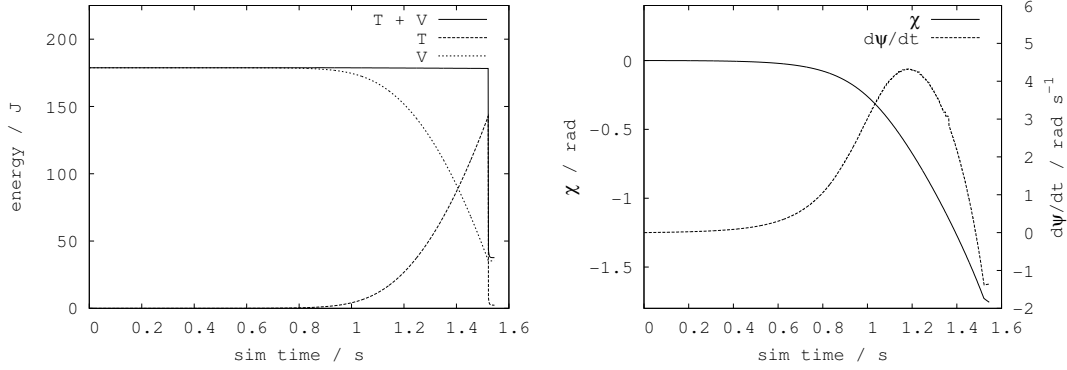


Figure 4: Energy conservation, left, and unicycle parameters in open-loop simulation. The unicycle falls freely in the wheel plane, stopping abruptly when the reaction wheel hits the floor at $t \simeq 1.5$ s.

Limits on control torques are given by reasonable physical assumptions about the torque provided by a human rider. Assuming a pedal lever arm of 0.2 m, a 60 kg rider can provide at least 12 N m of torque to the pedals. Similarly, assuming the rider can turn his upper body ($I \simeq 30 \text{ kg} \times (0.2 \text{ m})^2$) through a quarter turn in half a second, we obtain a lower bound on upper-body torque of $\simeq 10$ N m. Naveh et al. [7] suggest torques $\tau_\psi < 15$ N m, $\tau_r < 50$ N m. These values give a limit to the *recoverable* pitch angle,

$$\chi_{\max} = \arcsin \frac{\tau_{\psi, \max}}{mgl} \simeq 7^\circ \quad (1)$$

and since we may assume that errors in roll are corrected by yawing then pitching, and that the unicycle will continue to fall during this motion, we must have $\theta_{\max} < \chi_{\max}$. If the unicycle exceeds these limits, we can consider the controller to have failed.

Figure 4 shows the behaviour of the unicycle model with the controller switched off. The “open-loop” behaviour serves as verification of the physical model: we can see that as the unicycle falls from a vertical position, it transfers the potential energy of the rider initially to kinetic energy of the wheel, and then back to the rider as the pitch angle increases further. The left-hand plot confirms that our model conserves energy until the reaction wheel reaches the floor.

3.2 Control strategy in 2D

In order to ascertain whether the control strategy described could work in the more difficult system of the full 3D unicycle, a 2D unicycle model was constructed to serve as proof of principle. In this case the model was directly modelled from the Lagrangian and numerically integrated. Figure 5 shows a plot of the unicycle degrees of freedom when controlled by a network that not only balanced the unicycle, but also allowed for “path following” control. At roughly ten second intervals the unicycle is asked to move left, stay still, and then move back to the right.

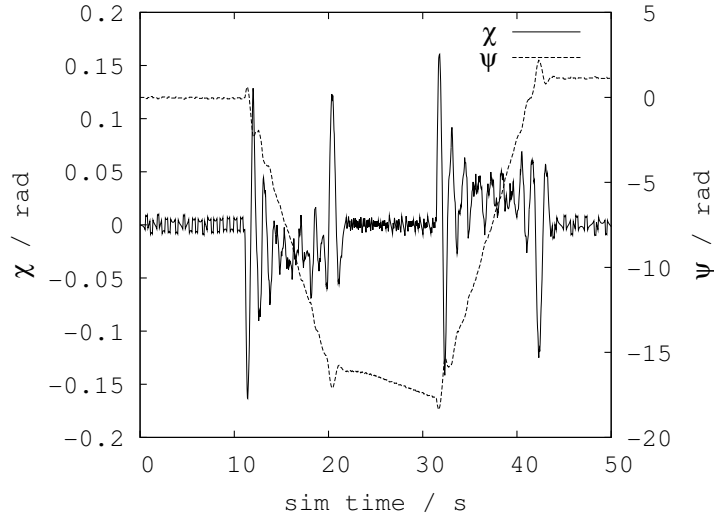


Figure 5: 2D unicycle proof of concept. The figure shows the unicycle horizontal position (given as the wheel angle, ψ) and the pitch angle, χ , as the unicycle is asked to move left, stay still, and then move right.

The unicycle’s movement strategy is to introduce a pitch error. Subsequent partial correction of the pitch error adds wheel speed. Note the reduction in the size of pitch oscillations during steady motion. Movement instructions were passed in to the neural network as an additional input, representing a target wheel velocity, and attention paid to this input was rewarded in the GA fitness function.

3.3 Controller behaviour in 3D

We now present the characteristics of some of the evolved controllers for the 3D model. The physical parameters of the unicycle for these simulations are given in Appendix C.

Figure 6 shows the trajectory and pitch/roll state of the unicycle under the control of two different networks which use only four state variables (θ , χ , $\dot{\chi}$, $\dot{\phi}$) as inputs and have four hidden-layer neurons. The output of these networks is interpreted as a nonlinear “bang-bang” control: output values above 0.75 result in a torque impulse in one direction, and those below -0.75 result in one in the opposite direction. The size of the torque impulse is set such that if applied at every timestep, the delivered torque would correspond to the maximum physically realistic torque, as discussed earlier. Intermediate values leave the unicycle alone.

The two rather different controllers (we will refer to the top controller as \mathcal{A} and the bottom as \mathcal{B}) were evolved with different fitness functions: \mathcal{A} aiming to minimize roll angle and overall kinetic energy of the unicycle, and \mathcal{B} aiming to minimize the imagined “power consumption” of the controlling motors (a small penalty was introduced for each timestep in which controller outputs were nonzero).

A full discussion of the features of these controllers will follow in Section 4. For now we

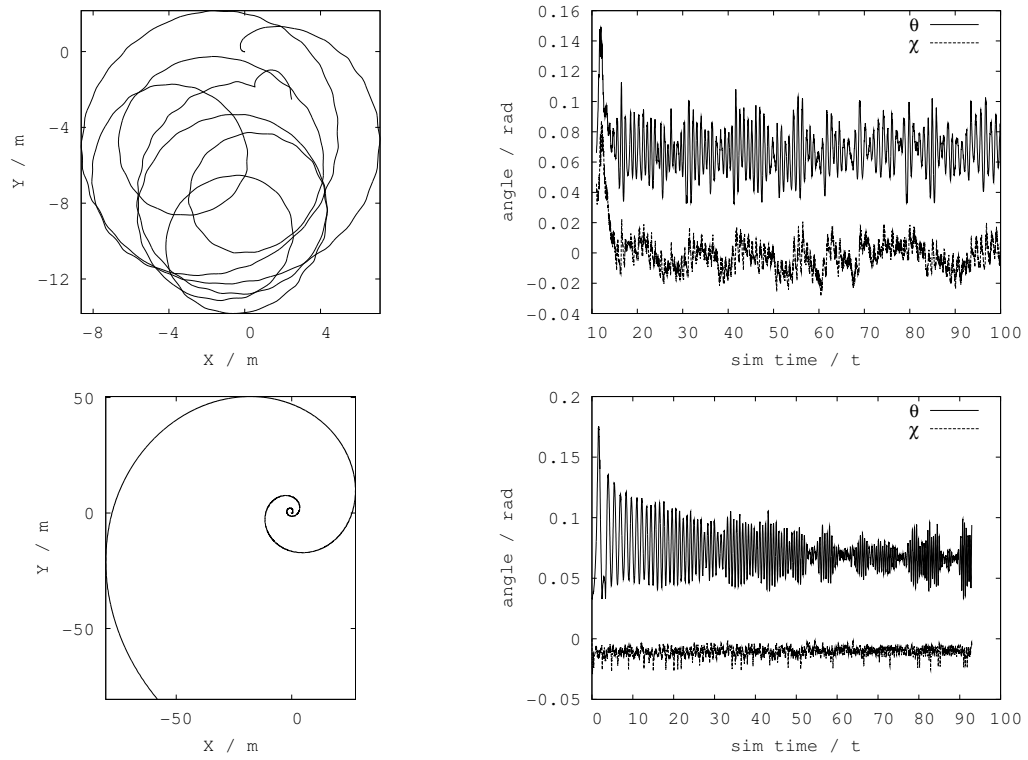


Figure 6: Ground trajectory, left, and pitch/roll in two controlled runs. Controller \mathcal{A} on top, and \mathcal{B} below. Note the vertical scale on the right-hand plots.

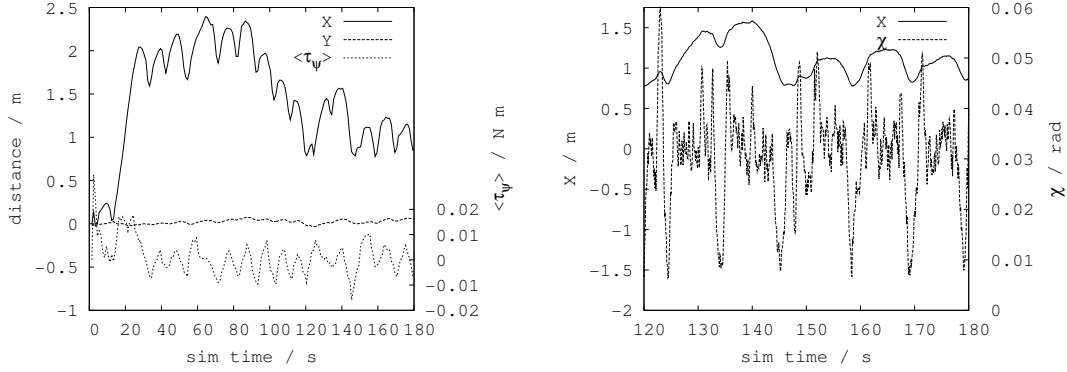


Figure 7: Preliminary results from roll-actuating controller \mathcal{C} . *Left:* trajectory and time-averaged pedal torque (averaged over previous 5 s). *Right:* over a shorter time period, the relationship between pitch variations and position.

Start angle / rad	\mathcal{A} successes	\mathcal{B} successes
0.01	23	39
0.05	29	24
0.10	37	16
0.15	26	6
0.20	15	5

Table 1: Number of runs in which the controller successfully transitioned from a stationary starting state to a stable moving state, over 50 tests.

note only that both controllers succeed in keeping the unicycle upright for long periods of time. In the absence of disturbances, controller \mathcal{A} can control the unicycle for at least 30 minutes of simulation time.

Figure 7 shows preliminary results from a roll-actuating controller, \mathcal{C} , where the axis of the reaction wheel lies along the unicycle y -axis, as shown on the right of Figure 3. These results also include a basic rolling friction model, where the frictional torque on the wheel, $\tau_{\text{fric}} = -Wr\gamma_{\text{roll}} \text{sgn}(\dot{\psi})$ and $\gamma_{\text{roll}} = 0.005$, a commonly used value for rubber on concrete rolling resistance. With a fitness function which aims to find a stationary or “idle” unicycle, we evolve controllers like this one. For this unicycle, $M = 20 \text{ kg}$, $\tau_{r,\text{max}} = 20 \text{ N m}$, $\tau_{\psi,\text{max}} = 15 \text{ N m}$. Unsurprisingly, the provision of direct actuation of the roll degree of freedom allows controller \mathcal{C} to keep the unicycle upright with near-zero wheel speed indefinitely.

3.4 Startup success

Transient behaviour is visible in Figure 6 at the start of each run. During evaluation, the unicycle is started with its post axis lying along a randomly chosen direction within a cone extending

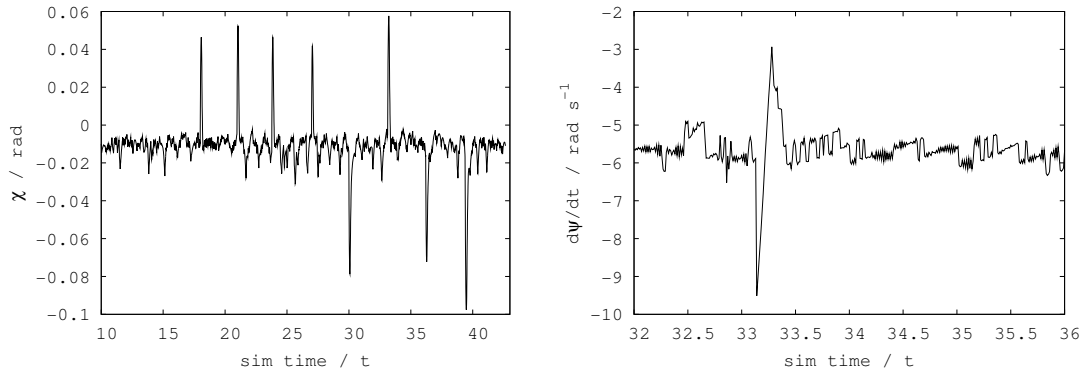


Figure 8: Disturbance rejection: a unicycle is repeatedly subjected to large torque impulses to the pedals. *Left:* the pitch angle is rapidly brought back to the equilibrium position. *Right:* changes in wheel speed as a result of one of the perturbations.

to 0.1 rad from the world vertical. Neither \mathcal{A} nor \mathcal{B} was able to successfully control the unicycle from all starting orientations. In order to get a better idea of how difficult the controllers found it to “start up”, each controller was tested for its ability to make the transition from stationary to a stable moving state at a variety of angles. The unicycle was started with its axis lying at a specified angle to the vertical, but with a random orientation. Table 1 summarises the results of this study.

3.5 Disturbance rejection

The evolved controllers successfully reject large disturbances: Figure 8 shows controller \mathcal{B} being subjected to pedal torque impulses seven times the size of the control torque while continuing to control the unicycle. The right-hand plot shows a close-up of the response to one impulse: the perturbation adds significant speed to the wheel which is quickly removed by the controller. The controller “overshoots,” but this is deliberate: it enables the correction of the pitch error which has been introduced as a result of the perturbation. The height of the correction is less than that of the original perturbation, as expected.

4 Discussion

4.1 Validity of physical model

The wheel-ground friction interaction is one of the most important parts of the physical model. Real human unicyclists control their yaw by taking advantage of both conservation of angular momentum and the presence of significant friction at the ground. In particular, the nonconservative nature of the friction forces allow a rider to make rotationally asymmetric motions in yaw with the limited rotation of the upper body. It is not clear whether the friction model used is valid. In particular, it is possible that static friction plays an important role in real-world

unicycle dynamics. These are questions that would be best resolved in a laboratory with a real unicycle.

Rolling friction, another nonconservative force associated with wheel rotation, may also have been another unfortunate omission. As will be discussed shortly, the absence of a means of dumping excess energy while moving purely longitudinally may mean that certain control modes available in the real world are unavailable in our simulation.

4.2 2D control

The proof-of-concept study done with a 2D model, while not the focus of this report, deserves limited attention as a demonstration of the principle of neural network control. Certainly, the problem is a much easier one to solve (the problem is only mildly underactuated, with one control input for two degrees of freedom), but we were also able to evolve networks that would listen to the instructions of a virtual rider, specifying the direction and speed of desired motion. Yet more impressive, the networks can be trained to ignore out-of-bounds instructions such as “set wheel velocity to 500 rad s^{-1} ”, leading to a robust means of providing a primary objective (“stay upright”) while still providing rewards for secondary goals (“move left”). All of this is possible with only a few minutes processing on a consumer-grade computer.

4.3 3D control

The controllers evolved by the GA, three of which are described in Sections 3.3–3.5, are very different from the more conventional controllers described in the literature. In particular, the GA has found two controllers in which the longitudinal and lateral systems of the unicycle are clearly not treated as separate. Indeed, attempts to evolve such separated controllers (by pre-setting network weights so as to effectively encode two separate networks) consistently “evolve out” the separation, apparently finding more successful controllers when the two systems are coupled.

One particular success of this approach is shown in Table 1: a small but significant fraction of the time, controllers \mathcal{A} and \mathcal{B} are able to stabilise networks that start with pitch or roll angles greater than the supposed recoverable pitch angle shown in Equation 1. We had assumed that such errors were controlled by first yawing into the plane of the error and then correcting the error with the pedal torque. Watching the controllers in simulation shows that this assumption is wrong. Such an error can also be corrected by “turning under” the error: a turn towards the direction of imminent fall cannot (by energy conservation) fully correct such an error, but it allows the time over which the fall occurs to be lengthened (by delivering kinetic energy to the wheel rather than the falling rider) so that pedal torque can supply enough energy to correct the error.

Controllers that have zero average wheel velocity (“idling” controllers) appear to be hard to find when no direct pitch actuation is available. Many runs of the GA with a variety of fitness functions suggest that control is more easily achieved when the unicycle is moving. Successful controllers introduce an initial pitch or roll error, as seen in Figure 6, and it is in the correction of this error that they pick up wheel speed and find a controllable equilibrium. In

fact, most controllers stabilise around a small roll angle⁵. The fact that all the controllers use roll to introduce this kinetic energy may well be down to a previously mentioned oversight in the physical model: a lack of rolling friction.

Consider a unicycle which wishes to stabilise itself about some nonzero pitch angle. On the introduction of, say, a positive pitch error, the unicycle must speed up, effectively “catching up” with the rider, bringing the pitch angle back to the equilibrium position. But the unicycle now has an excess of kinetic energy⁶: in the absence of air resistance or wheel rolling resistance, there is nowhere to dump this energy. By contrast, the same reasoning does not apply to roll angle because the energy can indeed be dumped by yawing friction.

We also notice that while the first controller remains near the origin of coordinates, circling in paths of a few wheel diameters in radius, the second controller spirals out, picking up speed as it goes. This second controller is an example of how a badly designed fitness function can result in controllers that keep the unicycle upright for a long time, but are not strictly stable. The increasing kinetic energy of the system eventually causes the unicycle to fall over. Requiring minimal (or at least bounded) kinetic energy of the system appears to be an important feature of the fitness function. Nonetheless, this controller does demonstrate the ability to reject noise and disturbances, as seen in Figure 8. The plot of wheel velocity clearly shows how the “bang-bang” control corrects the error, applying maximum torque (from $t = 33.15$ to 33.25 s) to reverse the pitch error, and subsequently applying torque impulses separated by pauses to damp the correction.

Although the results are preliminary, it is worth briefly discussing controller \mathcal{C} , a controller that can actuate pitch and roll. The GA easily finds solutions in which at equilibrium the unicycle remains largely stationary. In the left-hand plot of Figure 7, the large initial deviation in horizontal position corrects a starting pitch error, after which the unicycle stabilises and undergoes small oscillations in pitch and position. These oscillations are asymmetrical, and the right-hand plot shows how the shorter excursions (to lower values of X) correspond with smaller pitch angles. This asymmetry and the nonzero equilibrium pitch position may be understood as a consequence of the geometry of the unicycle: the centre of mass of the rider/reaction wheel is directly above the wheel axle at $\chi \simeq 0.03$ rad.

Lastly, it is noteworthy that controller \mathcal{A} appears to be more successful when started at a nonzero angle (see Table 1). This effect does appear to be statistically significant (difference from other nearby angles $> \sqrt{50}$) although the sample size is rather small. This may perhaps be understood by comparison with Figure 6, in which we can see that the equilibrium roll angle is roughly 0.07 rad while the equilibrium pitch is very close to upright. This gives a particular value for the equilibrium potential energy of the unicycle, and starting the unicycle with a nearby value of potential energy appears to help the controller stabilise the system.

4.4 GA and NN observations

The success of any GA depends strongly on its fitness function. A wide variety of fitness functions were tested in this study, and if one thing is clear it is that GA is unforgiving of any

⁵The fact that both yaw-actuated controllers presented stabilise around positive roll angles here is of little relevance, as the GA also finds controllers that stabilise around a negative roll angle

⁶We are tacitly assuming that the controller is not perfect.

slip-ups in fitness function design⁷. A few general principles present themselves:

1. Rewarding controllers that minimize or limit kinetic energy tends to lead to solutions that are stable over the long term.
2. There is a fine balance to be struck between “pickiness” and “learnability”. Broadly speaking, this corresponds to the width of the hill the fitness function presents in solution space. A sharper peak may eventually result in better solutions, but if it is too sharp, GA cannot beat a brute-force search of the solution space, making the problem computationally infeasible. On the other hand, a peak that is too broad may waste computational power evolving controllers that are inadequate.
3. Fail fast: it is better to abort evaluation early than continue adding zero to the score of an individual for many physics integration steps.

Similarly, we can make some general observations about working with neural networks that may help further study:

1. More hidden-layer nodes are not always better. Even if the best 10-node network is better than the best 5-node network, the addition of these nodes may add more than 50 (!) dimensions to the search space, making the search computationally infeasible.
2. Choice of coordinates is important, but the amount this matters depends on network architecture. A feedforward network without bias will find the inverted pendulum problem hard if the vertical axis has been chosen to be $\chi \neq 0$. If the vertical is $\chi = 0$, the network need only perform multiplications and can learn the solution faster (and with fewer nodes). Even with biases, helpful coordinate choices can minimize the number of mutations required to give a functional network.

Despite its success, there are a number of important issues with this approach to controller design. In particular, the current implementation results in controllers that are “overdesigned”, that is, they are useful for only one unicycle with fixed physical parameters. We could attempt to evolve a more general controller that takes some of the physical parameters of the unicycle as inputs (say m, M, r, l), and is evaluated against a wide range of physically plausible unicycles, but this is adding yet more weights to the network, increasing the size of the search space. It might be more fruitful to attack this problem by normalising all angular rate inputs to the unicycle time constant, $\sim \sqrt{l/g}$. It might even be possible to adapt an existing controller in this way.

In addition, a few limited experiments suggest that obtaining controllers that respect a target wheel velocity or yaw rate is not as easy as in the 2D case. While a self-balancing unicycle might be useful, it would be much more useful to be able to direct its motion. As such, this remains an important limitation of the results described in this report.

⁷Early attempts neglected to penalise controllers that span up the reaction wheel to unphysical speeds, pinning themselves in the vertical position by conservation of angular momentum. An effective but no doubt nauseating control method...

5 Conclusion

The results presented suggest there is considerable mileage in this approach to controller design, but also indicate significant challenges not yet addressed.

First, the positive: from a randomly seeded population, the GA successfully bred networks that were able to keep the unicycle upright over long periods of time. Not only this, but the networks were able to transition from stationary to stable moving states, often regarded to be one of the hardest parts of learning to ride a unicycle. The best controllers treat the longitudinal and lateral systems together, allowing recovery from steeper angles than do uncoupled controllers.

On the other hand, there are major outstanding issues with this control approach. Neural networks are “black boxes”, and notoriously difficult to analyse. More work is needed before the results obtained here become a general control strategy for unicycles. In addition, the computational requirements of the genetic algorithm do not scale favourably as we increase the demands on the neural network. As such, it seems that the best place for a neuro-controller is as a core component of a higher-level control scheme. Such a scheme might achieve control of yaw and wheel speed by artificially altering the coordinates fed into the network⁸.

The strength of supervised learning techniques for difficult control problems lies in their generality. In this report we have presented an approach applicable to a wide variety of problems, including those not accessible to direct analysis, and have shown that even with modest computing resources good progress can be made. In particular, this author thinks there is a certain elegance in an approach that, in its own simplistic way, attempts to mimic some of the processes underlying biological learning.

References

- [1] E. Coumans. Bullet physics library. URL <http://bulletphysics.com>. 4
- [2] G. Cybenko. Approximation by superpositions of a sigmoidal function. *Math. Control Signal Systems*, 2(4):303–314, Dec 1989. doi: 10.1007/BF02551274. 5
- [3] K. Hornik. Approximation capabilities of multilayer feedforward networks. *Neural Networks*, 4(2):251–257, Jan 1991. doi: 10.1016/0893-6080(91)90009-T. 5
- [4] N. Metni. Neuro-control of an inverted pendulum using genetic algorithm. *ACTEA*, pages 27–33, Apr 2009. doi: 10.1109/ACTEA.2009.5227952. 7
- [5] M. Mitchell and S. Forrest. Genetic algorithms and artificial life. *Artificial Life*, Jan 1994. doi: 10.1162/artl.1994.1.267. 6
- [6] D. J. Montana and L. Davis. Training feedforward neural networks using genetic algorithms. *Proceedings of the International Joint Conference on Artificial Intelligence*, pages 762–767, May 1989. 22

⁸Yaw could be modified by applying an offset to the measured yaw coordinate, and wheel speed by tricking the unicycle into thinking that it was on an incline.

- [7] Y. Naveh, P. Z. Bar-Yoseph, and Y. Halevi. Nonlinear modeling and control of a unicycle. *Dynamics and Control*, 9(4):279–296, 1999. 2, 8, 9
- [8] P. C. Paris and L. Zhang. A disk rolling on a horizontal surface without slip. *Mathematical and computer modelling*, Jan 2002. 19
- [9] T. Reil and P. Husbands. Evolution of central pattern generators for bipedal walking in a real-time physics environment. *Evolutionary Computation, IEEE Transactions on*, 6(2): 159–168, 2002. 2
- [10] D. W. Vos and A. H. von Flotow. Dynamics and nonlinear adaptive control of an autonomous unicycle: theory and experiment. *Proceedings of the 29th IEEE Conference on Decision and Control*, pages 182–187 vol. 1, 1990. 2
- [11] D. Whitley, S. Dominic, R. Das, and C. Anderson. Genetic reinforcement learning for neurocontrol problems. *Machine Learning*, 13(2):259–284, 1993. 7
- [12] J. Zhao, M. Xiong, and H. Jin. Dynamics and a convenient control design approach for a unicycle robot. *IEEE International Conference on Information and Automation*, pages 706–711, 2010. 2

4962 words.

Appendices

A Derivation of equations of motion

The equations of motion of the unicycle are easily written down – if not easily solved – using the Lagrangian formulation of Newtonian mechanics. What follows is an extension of the equations of motion for a disk on a horizontal surface as derived by Paris and Zhang [8]. The no-slip constraint at the wheel contact point is non-holonomic⁹ and will be addressed using the method of Lagrange multipliers. The constraint representing the fixed length of the unicycle seatpost is addressed more simply, by judicious use of the system coordinates in the Lagrangian.

There are three contributions to the kinetic energy, corresponding to the velocity of the wheel, that of the rider, and the angular velocity of the wheel. We express these velocities in terms of vectors \mathbf{a} and \mathbf{b} , the positions of the wheel centre and rider respectively,

$$\mathbf{a} = \begin{pmatrix} X \\ Y \\ r \cos \theta \end{pmatrix}$$

$$\mathbf{b} = \mathbf{a} + l \begin{pmatrix} \sin \chi \cos \phi - \cos \chi \sin \theta \sin \phi \\ \sin \chi \sin \phi + \cos \chi \sin \theta \cos \phi \\ \cos \theta \cos \chi \end{pmatrix}$$

and the angular velocity of the disc, measured in the local coordinate basis as indicated in Figure 1:

$$\boldsymbol{\omega} = \begin{pmatrix} -\dot{\psi} + \dot{\phi} \sin \theta \\ -\dot{\theta} \\ \dot{\phi} \cos \theta \end{pmatrix}$$

The wheel is modelled as a thin disc, radius r , and mass m . The “rider” is a point mass M on the end of a pole, length l , free to rotate about the wheel axle. In the wheel-local coordinate basis, the inertia tensor of the wheel is

$$I = \text{diag}(I_{\text{ax}}, I_{\text{tr}}, I_{\text{tr}}), \quad I_{\text{ax}} = \frac{mr^2}{2}, \quad I_{\text{tr}} = \frac{mr^2}{4}$$

There are two simple contributions to the potential energy from the masses of the wheel and rider, giving an overall Lagrangian, L :

$$T = \frac{m}{2} \dot{\mathbf{a}}^2 + \frac{M}{2} \dot{\mathbf{b}}^2 + \frac{\boldsymbol{\omega}^T I \boldsymbol{\omega}}{2} \quad (2)$$

$$V = m g \mathbf{a} \cdot \hat{\mathbf{Z}} + M g \mathbf{b} \cdot \hat{\mathbf{Z}} \quad (3)$$

$$L = T - V \quad (4)$$

⁹That is, there are an infinite number of paths through the system state space connecting any two states of the system. This is most easily understood as the fact that a unicyclist may trace out any number of distinct curved paths on the ground while travelling between two fixed points.

which, written explicitly in terms of the generalised coordinates, gives¹⁰

$$\begin{aligned}
L = & \frac{m}{2} \left(\dot{X}^2 + \dot{Y}^2 + r^2 c_\theta^2 \dot{\theta}^2 \right) \\
& + \frac{I_{\text{tr}}}{2} \left(c_\theta^2 \dot{\phi}^2 + \dot{\theta}^2 \right) \\
& + \frac{I_{\text{ax}}}{2} \left(-s_\theta \dot{\phi} + \dot{\psi} \right)^2 \\
& + \frac{M}{2} \left(l s_\chi c_\theta \dot{\chi} + (r + l c_\chi) s_\theta \dot{\theta} \right)^2 \\
& + \frac{M}{2} \left(\dot{X} - l \left((c_\phi c_\chi s_\theta + s_\phi s_\chi) \dot{\phi} - (s_\phi s_\chi s_\theta + c_\phi c_\chi) \dot{\chi} + s_\phi c_\chi c_\theta \dot{\theta} \right) \right)^2 \\
& + \frac{M}{2} \left(\dot{Y} - l \left((s_\phi c_\chi s_\theta - c_\phi s_\chi) \dot{\phi} + (c_\phi s_\chi s_\theta - s_\phi c_\chi) \dot{\chi} - c_\phi c_\chi c_\theta \dot{\theta} \right) \right)^2 \\
& - g c_\theta (m r + M(r + l c_\chi))
\end{aligned}$$

The equations of motion of the system are then given by the Euler-Lagrange equations,

$$\frac{d}{dt} \left(\frac{\partial L}{\partial \dot{q}_i} \right) - \frac{\partial L}{\partial q_i} = Q_i \quad (5)$$

where the q_i are the coordinates $\{X, Y, \phi, \chi, \theta, \psi\}$, and the Q_i are nonconservative “virtual” forces and torques acting to satisfy the no-slip constraint at the wheel-ground contact point.

Geometric considerations give the form for the no-slip constraints,

$$\begin{aligned}
\delta X &= r \cos \phi \delta \psi - r \sin \theta \cos \phi \delta \phi - r \cos \theta \sin \phi \delta \theta \\
\delta Y &= r \sin \phi \delta \psi - r \sin \theta \sin \phi \delta \phi + r \cos \theta \cos \phi \delta \theta
\end{aligned}$$

and d’Alembert’s principle gives the form of the constraint forces, which must do no work. That is,

$$\begin{aligned}
\delta W &= 0 \\
&= \sum Q_i \delta q_i \\
&= \lambda_X (-r \cos \phi \delta \psi + r \sin \theta \cos \phi \delta \phi + r \cos \theta \sin \phi \delta \theta) + \\
&\quad \lambda_Y (-r \sin \phi \delta \psi + r \sin \theta \sin \phi \delta \phi - r \cos \theta \cos \phi \delta \theta)
\end{aligned}$$

Comparison of coefficients gives the virtual forces:

$$\begin{aligned}
Q_X &= \lambda_X \\
Q_Y &= \lambda_Y \\
Q_\phi &= \lambda_X r \sin \theta \cos \phi + \lambda_Y r \sin \theta \sin \phi \\
Q_\theta &= \lambda_X r \cos \theta \sin \phi + \lambda_Y r \cos \theta \cos \phi \\
Q_\psi &= -\lambda_X r \cos \phi - \lambda_Y r \sin \phi \\
Q_\chi &= 0
\end{aligned}$$

The result of all this is a set of formidable nonlinear differential equations for a hopelessly oversimplified model of unicycle and rider. See Appendix B for the full equations of motion.

¹⁰NB: $c_\xi = \cos \xi$ and $s_\xi = \sin \xi$.

B Equations of motion

For the sake of completeness, we include the full equations of motion.

$$\begin{aligned} \frac{(m+M)\ddot{X}}{Ml} = & - (s_\phi c_\chi s_\theta - c_\phi s_\chi) (\dot{\phi}^2 + \dot{\chi}^2) - s_\phi c_\chi s_\theta \dot{\theta}^2 \\ & - 2(c_\phi s_\chi s_\theta - s_\phi c_\chi) \dot{\phi} \dot{\chi} \\ & + 2c_\theta \dot{\theta} (c_\phi c_\chi \dot{\phi} - s_\phi s_\chi \dot{\chi}) \\ & + (c_\phi c_\chi s_\theta + s_\phi s_\chi) \ddot{\phi} \\ & - (s_\phi s_\chi s_\theta + c_\phi c_\chi) \ddot{\chi} \\ & + s_\phi c_\chi c_\theta \ddot{\theta} + \lambda_X / Ml \end{aligned}$$

$$\begin{aligned} \frac{(m+M)\ddot{Y}}{Ml} = & (c_\phi c_\chi s_\theta + s_\phi s_\chi) (\dot{\phi}^2 + \dot{\chi}^2) + c_\phi c_\chi s_\theta \dot{\theta}^2 \\ & - 2(s_\phi s_\chi s_\theta + c_\phi c_\chi) \dot{\phi} \dot{\chi} \\ & + 2c_\theta \dot{\theta} (s_\phi c_\chi \dot{\phi} + c_\phi s_\chi \dot{\chi}) \\ & + (s_\phi c_\chi s_\theta - c_\phi s_\chi) \ddot{\phi} \\ & + (c_\phi s_\chi s_\theta - s_\phi c_\chi) \ddot{\chi} \\ & - c_\theta c_\phi c_\chi \ddot{\theta} + \lambda_Y / Ml \end{aligned}$$

$$\begin{aligned} (mr^2 c_\theta^2 + Mrs_\theta^2 (r + 2lc_\chi) + Ml^2 c_\chi^2 + I_{tr}) \ddot{\theta} = & c_\theta s_\theta \left\{ (Ml^2 c_\chi^2 + I_{ax} - I_{tr}) \dot{\phi}^2 \right. \\ & \left. - Mlrc_\chi \dot{\chi}^2 \right. \\ & \left. + (mr^2 - Mr^2 - 2rlc_\chi) \dot{\theta}^2 \right\} \\ & - 2Ml^2 c_\chi^2 c_\theta \dot{\phi} \dot{\chi} + 2Ml (lc_\chi + rs_\theta^2) s_\chi \dot{\theta} \dot{\chi} - c_\theta I_{ax} \dot{\phi} \dot{\psi} \\ & + Mlc_\theta (s_\phi c_\chi \ddot{X} - c_\phi c_\chi \ddot{Y} - ls_\chi c_\chi \ddot{\phi} - rs_\chi s_\theta \ddot{\chi}) \\ & + g (mr + M (r + lc_\chi)) s_\theta \\ & + rc_\theta (s_\phi \lambda_X - c_\phi \lambda_Y) \end{aligned}$$

$$\begin{aligned}
(I_{\text{tr}}c_\theta^2 + I_{\text{ax}}s_\theta^2 + Ml^2 (c_\chi^2s_\theta^2 + s_\chi^2)) \ddot{\phi} &= I_{\text{tr}}c_{2\theta}\dot{\theta}\dot{\phi} + I_{\text{ax}} (c_\theta\dot{\theta}\dot{\psi} + s_\theta\ddot{\psi} - c_{2\theta}\dot{\theta}\dot{\phi}) \\
&+ Ml (c_\phi c_\chi s_\theta + s_\phi s_\chi) \ddot{X} \\
&+ Ml (s_\phi c_\chi s_\theta - c_\phi s_\chi) \ddot{Y} \\
&+ Ml^2 (c_\chi s_\chi s_\theta \dot{\theta}^2 - s_{2\chi} c_\theta^2 \dot{\phi} \dot{\chi} - c_\chi^2 s_{2\theta} \dot{\phi} \dot{\theta} + 2s_\chi^2 c_\theta \dot{\chi} \dot{\theta} - c_\chi s_\chi c_\theta \ddot{\theta} + s_\theta \dot{\chi} \dot{\chi}) \\
&+ r s_\theta (c_\phi \lambda_X - s_\phi \lambda_Y) \\
I_{\text{ax}} \ddot{\psi} &= I_{\text{ax}} (c_\theta \dot{\theta} \dot{\phi} + s_\theta \ddot{\phi}) - r (c_\phi \lambda_X + s_\phi \lambda_Y) \\
l \ddot{\chi} &= l s_\theta \ddot{\phi} - r s_\theta c_\theta s_\chi \ddot{\theta} \\
&- (s_\phi s_\chi s_\theta + c_\phi c_\chi) \ddot{X} + (c_\phi s_\chi s_\theta - s_\phi c_\chi) \ddot{Y} \\
&- (r c_\theta^2 s_\chi + l c_\chi s_\chi) \dot{\theta}^2 - l s_\chi c_\chi c_\theta^2 \dot{\phi}^2 \\
&- 2l c_\chi^2 c_\theta \dot{\theta} \dot{\phi} + g c_\theta s_\chi \\
\dot{X} &= r \cos \phi \dot{\psi} - r \sin \theta \cos \phi \dot{\phi} - r \cos \theta \sin \phi \dot{\theta} \\
\dot{Y} &= r \sin \phi \dot{\psi} - r \sin \theta \sin \phi \dot{\phi} + r \cos \theta \cos \phi \dot{\theta}
\end{aligned}$$

C Physical parameters

Symbol	Value	Description
m	1.0 kg	wheel mass
M	10.0 kg	rider mass ¹¹
r	0.5 m	wheel radius
l	1.2 m	seatpost length
dt	0.01 s	simulation integration time
$\tau_{\psi, \text{max}}$	10 N m	maximum pedal torque
$\tau_{r, \text{max}}$	10 N m	maximum seatpost twist torque
γ_c	0.02 m	rotational coulomb friction coefficient
γ_v	0.0001 m s rad ⁻¹	rotational viscous friction coefficient

D GA details

The success of the GA depends strongly on the details of the fitness function, mutation, and crossover operators. The crossover operator used is that described by Montana and Davis [6].

¹¹Note that the rider mass is not intended to represent a human, but rather a payload of batteries, electronics and reaction wheel needed to build an automated unicycle.

D.1 Fitness function

The fitness of a unicycle, given as f in the pseudocode below, is determined primarily by how long it can avoid falling over. $\delta(\xi)$ is a gaussian peaked at $\xi = 0$ with $\delta(0) = 1$.

```
 $f \leftarrow 0$ 
while  $t < T$  do
  if unicycle has fallen over then
    exit loop
5: else
   $f \leftarrow f + dt \cdot \delta(|\theta| + |\dot{\phi}| + \dots)$ 
  end if
  step unicycle physics by  $dt$ 
   $t \leftarrow t + dt$ 
10: end while
return  $f$ 
```

The unicycle is deemed to have fallen over if the seat lies outside a cone extending to some fixed angle about the vertical. In line 6 the unicycle is rewarded for keeping its roll angle near zero, and having a low rate of yaw. Many other sets of arguments to the delta function are possible, but this serves as a simple example.

Note that the maximum fitness is T , the maximum evaluation time.

D.2 Mutation operator

The mutation operator is parameterised by the *mutation rate* (set such that on average only one weight is mutated per network) and the mutation size, σ .

```
for each weight  $w$  in NN do
  if  $\text{random.uniform}(0, 1) < \text{mutation rate}$  then
     $w \leftarrow w + \text{random.gauss}(0, \sigma)$ 
  end if
end for
```

Here $\text{random.uniform}(a, b)$ returns a random real number in the range $[a, b)$, and $\text{random.gauss}(\mu, \sigma)$ returns a random number picked from a gaussian probability distribution with mean μ and standard deviation σ .

D.3 Crossover operator

The crossover operator takes two NNs, \mathcal{A} and \mathcal{B} , and creates a new NN with the same topology. For each non-input neuron in the new network, the incoming weights are copied from either \mathcal{A} or \mathcal{B} , chosen at random.

```
create new network  $\mathcal{C}$ 
for each non-input neuron  $n$  in  $\mathcal{C}$  do
  if  $\text{random.uniform}(0, 1) < 0.5$  then
    copy incoming weights for  $n$  from corresponding neuron in  $\mathcal{A}$ 
  else
```

```
    copy incoming weights for  $n$  from corresponding neuron in  $\mathcal{B}$   
  end if  
end for
```