ÉCOLE POLYTECHNIQUE FÉDÉRALE DE LAUSANNE

# DEVELOPING A WALKING CONTROLLER FOR A THREE-LINK 2D BIPED

MICRO-507: LEGGED ROBOTS MINI-PROJECT

---

Alessandra Capurro, Savyaraj Deshmukh, and Lauren Wright

2nd March 2020

# CHAPTER 1

# INTRODUCTION

## 1.1 CONTEXT

In order to understand the complexities of controlling legged locomotion, we modeled the kinematics and dynamics of a planar three link bipedal robot in MATLAB then simulated walking using two different control architectures. The two controllers evaluated were a PD controller and a neuronal controller inspired by RunBot[1]. In this paper we will cover the simulation, modelling, and controller implementation, and compare the performance of these two controllers on the basis of efficiency and robustness to perturbations over a defined set of walking speeds.

## 1.2 BIPED CONSTRUCTION AND CONTROL

The biped is constructed of three segments: a torso and two legs. Each segment has a centrally located point mass. Movement is controlled via actuation at the "hip" point where all three segments connect. The biped has three degrees of freedom but only two controllable inputs, as seen in figure 2.2, and is thus under-actuated.

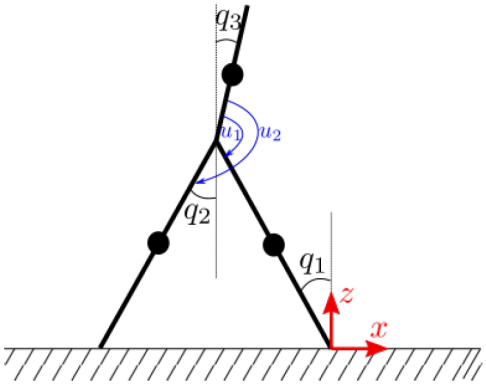


**FIGURE 1.1**
Biped with angular positions q1, q2, and q3 and control inputs u1 and u2

[1] *Geng, Tao, et al. "Fast Biped Walking with a Sensor-Driven Neuronal Controller and Real-Time Online Learning." The International Journal of Robotics Research, vol. 25, no. 3, Mar. 2006, pp. 243–259*

# CHAPTER 2

# METHODS

## 2.1 MODELLING AND VISUALIZATION OF THE 3-LINK

We start with a kinematic model of the robot derived from the kinematics of a double pendulum. This is an appropriate modelling simplification as the biped will always have one limb in contact with the ground. The dynamic model is a hybrid of a single support phase and double support phase. In the single support phase, the canonical variables are the three joint angles q1, q2, and q3 as shown in figure 1.1. The governing dynamic equations during this phase are derived by calculating the Lagrangian of this three link system. Because the environment is assumed to be ideal, the only terms in the Lagrangian are the kinetic energies and gravitational potential energies of three point masses without any losses.

During the double support phase, dynamics are modeled through an impact map which gives a mapping from pre-impact variables to post-impact variables. The impact is assumed to be an instantaneous switching of stance and swing legs and the new variables are calculated using conservation of angular momentum assuming a perfectly inelastic impact.

## 2.2 PD CONTROLLER ARCHITECTURE

Because of under-actuation, the PD controller was designed so that it correlates all three variables $(q_1, q_2, q_3)$, as seen in matrix B:

$$B = \begin{bmatrix} 1 & 0 \\ 0 & 1 \\ -1 & -1 \end{bmatrix}$$

The first input $u_1$ is directly controlling the position of the torso constraining $q_3$ to remain close to the desired angle, which was set to $\pi$/40 after experimentation. The second input, $u_2$, is imposing a relationship between $q_1$ and $q_2$

$$q_2 = -arctan(q_1) \tag{2.1}$$

in order to control both variables with the same actuator. Hence, the controller operates under the following equations:

$$u_1 = Kp_1(q_{3d} - q_3) - Kd_1(\dot{q}_3) \tag{2.2}$$

$$u_2 = Kp_2(q_{2d} + arctan(q_1)) + Kd_2(\dot{q}_{2d} + \frac{1}{1 + q_1^2}) \tag{2.3}$$

The limit on the actuators was imposed by the following constraints

$$if \quad |u_1| > 30 \quad then \quad u_1 = 30 * sign(u_1) \tag{2.4}$$

$$if \quad |u_2| > 30 \quad then \quad u_2 = 30 * sign(u_2) \tag{2.5}$$

The controller parameters are $K_{p_1}, K_{d_1}, K_{p_2}$, and $K_{d_2}$ and can be understood in the following ways:

- $K_{p_1}$ and $K_{p_2}$ are the proportional gains: the higher they are set, the larger the output torque will be provided for a given error between the actual and the desired value of our controlled variables. If we set this gains too low, the system will be less robust to disturbances. The values of $K_{p_1}$ and $K_{p_2}$ vary with the velocity we want to obtain.
- $K_{d_1}$ and $K_{d_2}$ are the derivative gains. We decided to keep those two gains equal and constant to -10 in all of the different settings for different velocities.

## 2.3 NEURONAL CONTROLLER ARCHITECTURE

This controller is based on on the architecture of RunBot which has a similar design (Figure 2.1). Each leg is actuated through a pair of extensor and flexor muscles that are in turn controlled by motor neuron models. These neurons also get inputs from the ground sensors and hip extensor neurons. These sensors are simply the impact detection and joint angle limit detection for swing leg.

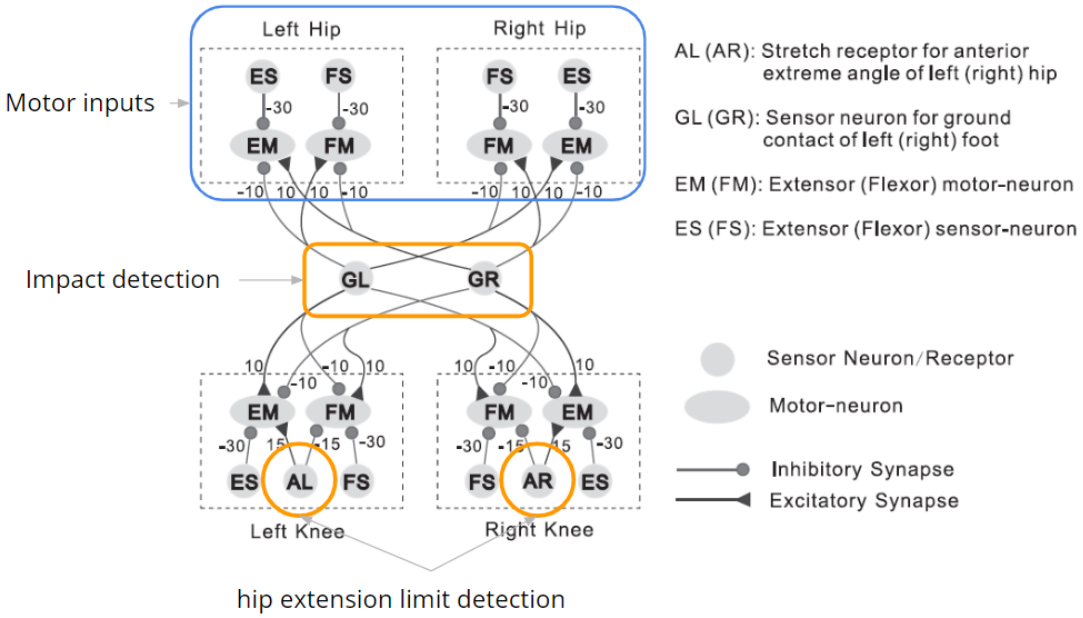


**FIGURE 2.1**
Neuronal controller architecture for RunBot and the relevant modules for our controller (Geng, Porr and Wörgötter 2006)

An important difference between RunBot and this model is that RunBot does not have a torso. In our case, the torso is a vital part of the dynamics because the stability of the whole body depends on the stability of torso. Also, this control system is underactuated since we have three state variables and only two control inputs.

To handle the underactuation, priority is given to torso stabilization. The control input torques are calculated as:

$$\begin{bmatrix} \tau_{q_1} \\ \tau_{q_2} \\ \tau_{q_3} \end{bmatrix} = \begin{bmatrix} 1 & 0 \\ 0 & 1 \\ -1 & -1 \end{bmatrix} \begin{bmatrix} u_1 \\ u_2 \end{bmatrix}$$

Torso controller is a combination of inverse dynamics controller along with PD stabilization. Hence, we set the control inputs such that

$$u_{q_3} = -u_1 - u_2 = u_{PD} + u_{inv} \tag{2.6}$$

where

$$u_{PD} = K_p(q_3^{des} - q_3) + K_d(\dot{q_3}^{des} - \dot{q_3}) \tag{2.7}$$

After stabilizing the torso, remaining control space is used for achieving forward motion. We use a hybrid system of stance leg rotation and swing leg retraction. The switching is determined by sensor inputs, i.e., ground contact and swing extension limit detection.

Ground contact is used to switch between the swing leg and stance leg. The swing leg extension limit is used to switch from the stance leg rotation regime to the swing leg retraction limit. This is simply the action of pulling back the swing leg after sufficient rotation. Although the RunBot architecture uses sensors to be sigmoid activated functions, we found that the time constant for this function was fast compared to muscle activation and could be approximated as instantaneous.

The control inputs are calculated using outputs from two motor neurons, viz. extensor and flexor. The motor neuron is a simplified version of leaky integrator model with a sigmoid activation function and the joint torque is given by difference between extensor and flexor neuron outputs scaled by a gain.
As a general overview of the control algorithm, after the impact, we have

$$u_1 = G_{stance} * (u_{lf} - u_{le}) \tag{2.8}$$

$$u_2 = -u_1 - u_{q_3} \tag{2.9}$$

where the left leg is assumed to be the stance leg. As the hip extension limit is reached, we shift to

$$u_2 = G_{swing} * (u_{rf} - u_{re}) \tag{2.10}$$

$$u_1 = -u_2 - u_{q_3} \tag{2.11}$$

Once impact is detected, the stance and swing legs switch positions and the above procedure is followed again.

## 2.4 SIMULATION

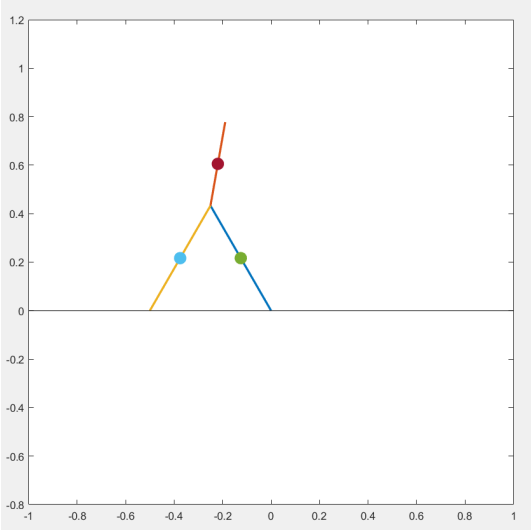


**FIGURE 2.2**
Three-Link Biped in MATLAB Simulation Environment

The dynamic and kinematic equations are integrated in MATLAB using the *ode45* solver before applying the controllers. Simulating in MATLAB gives us considerable freedom in tuning controller parameters and in influencing the simulation environment. Parameter tuning for the PD controller was accomplished via iterative experimentation to find a combination of gains that resulted in stable gaits. Parameters for the neuronal controller primarily came from the RunBot controller and some of them were originally taken from biological data.

In order to assess controller robustness, both internal and external perturbations were applied to the biped. Internal perturbations are applied as an absolute random noise distribution with a ceiling where each noise value is tied to a specific time step. Originally, we attempted to apply "pure" noise using a random distribution, however we found that the *ode45* solver would get stalled when performing integration back and forth in time. External perturbations are applied as quick torque applied to the torso - equivalent to pushing the torso once it's achieved steady state. This torque is only applied for a fraction of the gait, once per step. When considering the maximum perturbations that the biped can handle under each controller, we only count gaits that have a regular limit cycle even though in some cases the biped can continue forward locomotion despite perturbation.

# CHAPTER 3

# RESULTS

## 3.1 KINEMATICS AND DYNAMICS

Step transitions, modeled through the impact map, reveal some interesting qualities in energy transmission and loss. Because the impact is modeled as an instantaneous inelastic collision, potential energy before and after impact is conserved. However, the kinetic energy is not perfectly preserved; given starting angular positions and velocities $q_m = [\pi/6, -\pi/6, \pi/10]$ and $dq_m = [1, 0.2, 0]$, 27.93% of kinetic energy is lost in the impact. The kinetic energy loss due to impact as a function of initial angle can be seen in figure 3.1 below. Energy loss is also dependent on step length; for a larger step size energy loss increases.

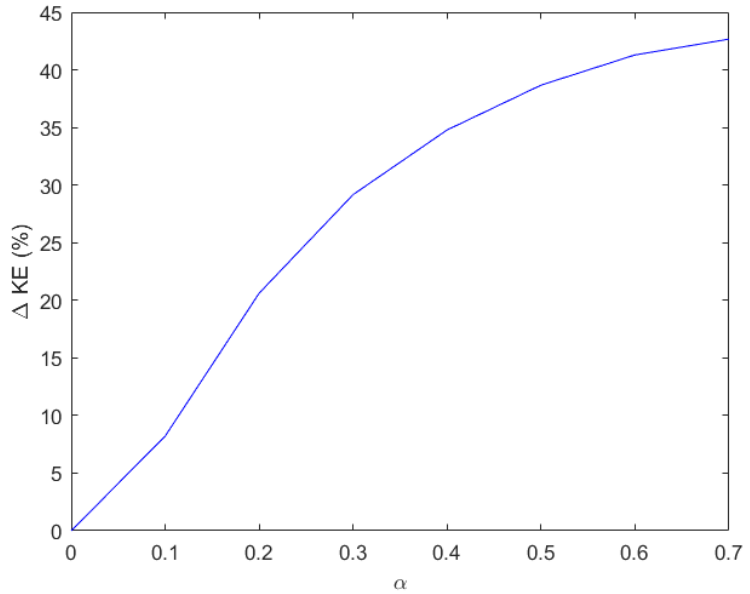


**FIGURE 3.1**
Kinetic energy loss as a function of step length variation ($\sim sin(2\alpha)$)

## 3.2 SIMULATION

The biped's gait was animated in MATLAB by visualizing the solutions to the equations of motion as they evolve over time. The animation is characterized by the real-time factor - the ratio between the time elapsed in the simulation and the actual time that the simulation takes. When we compute the system equations, a vector TE contains the time elapsed at the end of each subsequent step from the beginning of simulation. Dividing the last of these values - the total time of the simulation - by the real-time factor will calculate how much time it takes for the whole simulation to run in real-time. A real-time factor of 1 means that the actual time and the simulation time correspond exactly, while a real-time factor smaller than 1 means that the simulation will take more time than what is displayed in the vector TE. The animation is controlled in part by a "skip" parameter which controls how much time elapses between step visualizations and r0, the global position of the stance leg which affects forward movement. The real-time factor increases as the time skip is increased. This means that a larger skip value corresponds to a faster speed of animation. We point out three examples in order to better understand the "skip" parameter's influence:

| Simulation time | Skip | Real time | Real-time factor |
|---|---|---|---|
| 5.6403 | 5 | 28.7226 | 0.1964 |
| 5.6403 | 25 | 5.9150 | 0.9536 |
| 5.6403 | 50 | 3.9083 | 1.4432 |

## 3.3 PD CONTROLLER

The PD controller is quite sensitive to initial conditions when considering long term stability. Starting angles need to be sufficiently small and there needs to be a starting velocity applied to the torso otherwise the biped won't begin moving. The initial conditions we used are:

$$q0 = [\pi/12; -\pi/12; \pi/20]$$
$$dq0 = [0; 0; 5]$$

The PD controller is very sensitive to internal perturbation as well. Internal perturbation prevents the PD controller from being able to accurately control the relative angles of the limbs; the PD controller cannot handle noise of a magnitude larger than 0.005 at a speed of 0.5 m/s. However, external perturbation is only an issue at a speed of 0.5 m/s. In fact, using our method of applying external perturbations as only a short impulse per step, we were not able to find a maximum perturbation. When adjusting the length of time over which an external perturbation is applied, we found much more reasonable maximum perturbations for all three velocities. The cost of transport, that quantifies the energy efficiency of transporting the robot, using the PD controller is fairly low meaning that the controller leads to efficient gaits. Among the three chosen speeds, we can notice that the COT decreases increasing the speed until 0.7 m/s; after such value, the COT again increases indicating a loss in energy efficiency for high speeds.

| Parameter | 0.5 m/s | 0.7 m/s | 0.8 m/s |
|---|---|---|---|
| Kp1 | -100 | -100 | -500 |
| Kd1 | -10 | -10 | -10 |
| Kp2 | -1000 | -500 | -500 |
| Kd2 | -10 | -10 | -10 |
| Max Int. Perturbation | 0.005 | 0.015 | 0.012 |
| Max Ext. Perturbation | -10/+10 | $\infty$ | $\infty$ |
| COT | 23.6696 | 21.8263 | 28.2668 |

Plotting the joint angles against time we can notice that q1 and q2 behave in an expected oscillatory manner, while q3 reveals the compensatory nature of the torso; the fluctuation at the bottom of the cycle is an indication of the balancing effect of the torso which happens during the double support phase.

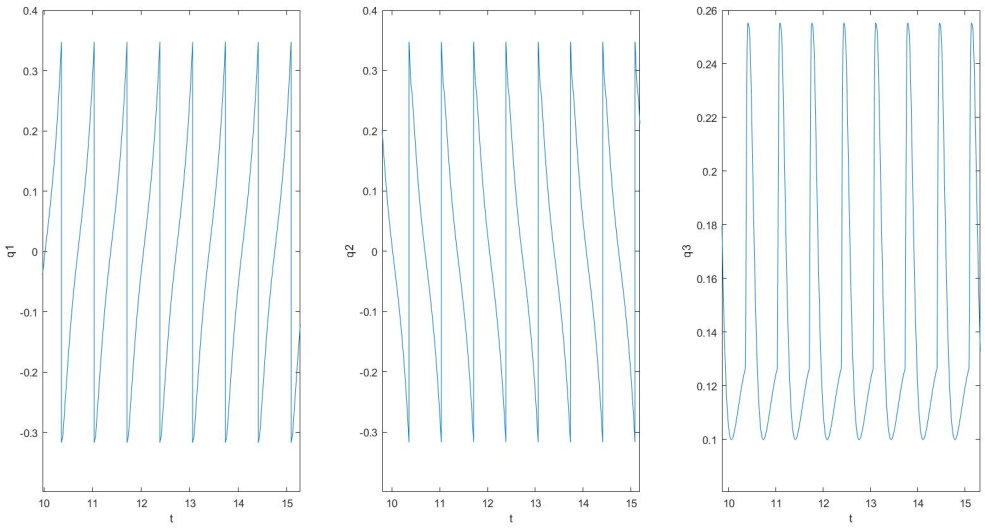


**FIGURE 3.2**
Joint angle position evolving in time at 0.5 m/s speed

From figure 3.3, the phase plot of each joint angle relative to its angular velocity, we can infer that steady state is achieved after about one step cycle when there is no perturbation applied. Additionally, we can see the limits of the joint angles during the gait. In figure 3.4, we can see how the shape of the limit cycle changes under the application of external perturbation.

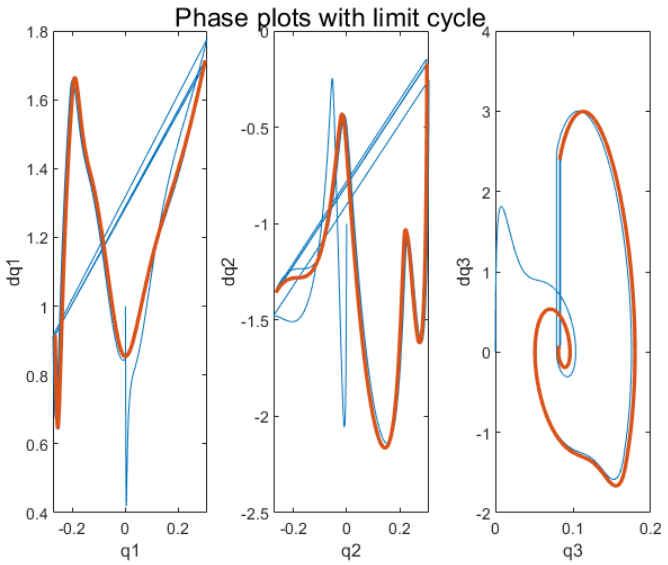


**FIGURE 3.3**
Phase plots and limit cycles of each joint angle relative to its angular velocity without perturbation at 0.5 m/s speed

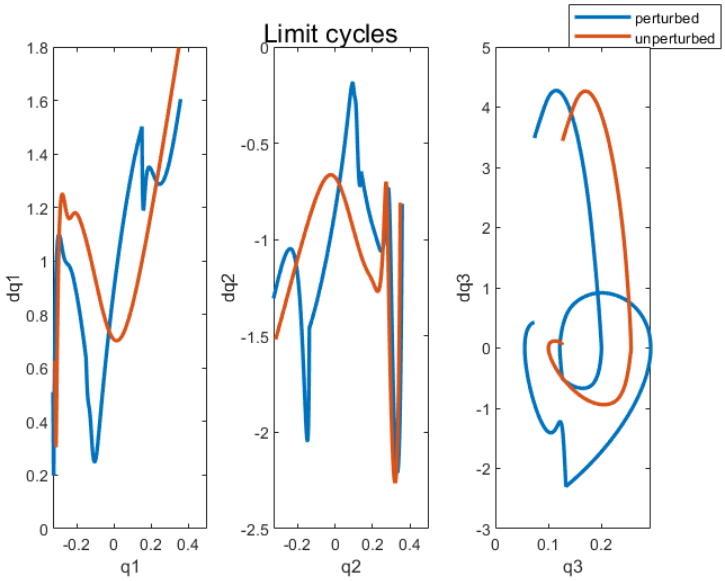


**FIGURE 3.4**
Limit cycles with unperturbed bot and with maximum applied external perturbation (-10) at 0.5 m/s speed

## 3.4 NEURONAL CONTROLLER

The neuronal controller is not very sensitive to perturbations. It can even achieve a stable limit cycle from many initial conditions, implying that the attraction basin is very large. Some of the main features of this controller are:

- Design parameters: Most of the parameters for the controller are related to the controller design for RunBot (Geng, Porr and Wörgötter 2006). Controlling velocity is achieved by a single variable which is the motor torque gain ($G$).
- Perturbations: The controller was found to be the most robust for intermediate velocities. The performance was better than PD controller for internal sensor perturbations, but worse for the external torque perturbations
- COT: Cost of Transport shows an increasing trend as the velocity increases. The values are significantly higher than the PD controller. This can be attributed to the fact that the swing leg retraction part of the gait decreases the forward momentum in order to step down, i.e., to achieve stability. Another energy consuming process is the torso stabilization ($\sim 15 - 30\%$ of the control input), which doesn't contribute to the forward motion.

| Parameter | 0.43 m/s | 0.57 m/s | 0.63 m/s |
|---|---|---|---|
| G (gain) | 5 | 15 | 20 |
| $K_p^{q_3}$ | 2 | 2 | 2 |
| $K_d^{q_3}$ | 10 | 10 | 10 |
| tau | 0.01 | 0.01 | 0.01 |
| theta | 1 | 1 | 1 |
| $w_{ground}$ | 10 | 10 | 10 |
| $w_{stretch}$ | 15 | 15 | 15 |
| $q_2^{threshold}$ | -0.05 | -0.05 | -0.05 |
| Max Int. Perturbation | 0.1 | 0.3 | 0.05 |
| Max Ext. Perturbation | 0.2 | 1 | 0.2 |
| COT | 11.1804 | 37.9107 | 48.6797 |

Below we can see the time series and phase plots of the simulation for one of the gaits. As evident from the animations, torso angle variations are very small compared to the other two legs. The joint angle time series for $q_2$ is also rather smooth (Figure 3.5) owing to the smooth dynamics of muscle model used for control. This is an interesting quality of the controller also pointed out in Geng, Porr and Wörgötter 2006; which can generate smooth motor input trajectories.

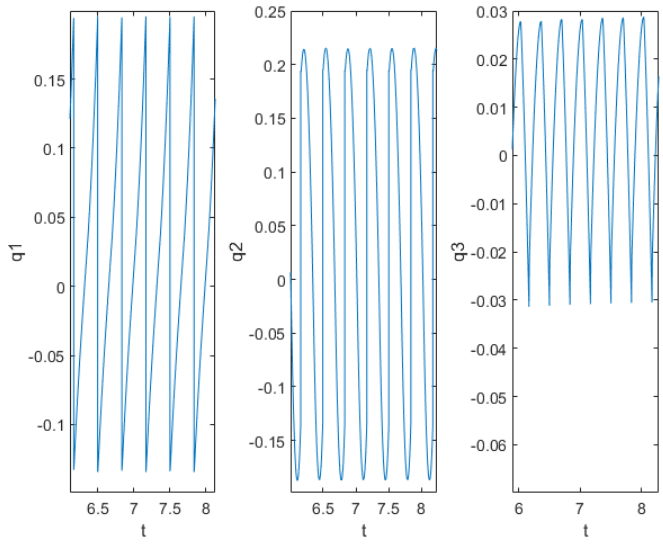


**FIGURE 3.5**
Joint angles time series

In Figure 3.6, the swing leg retraction reflex is can be visualized. The phase plot of $q_2$ shows the reversal of swing leg motion after $q_2$ crosses its threshold, which makes the transition from swing to stance.

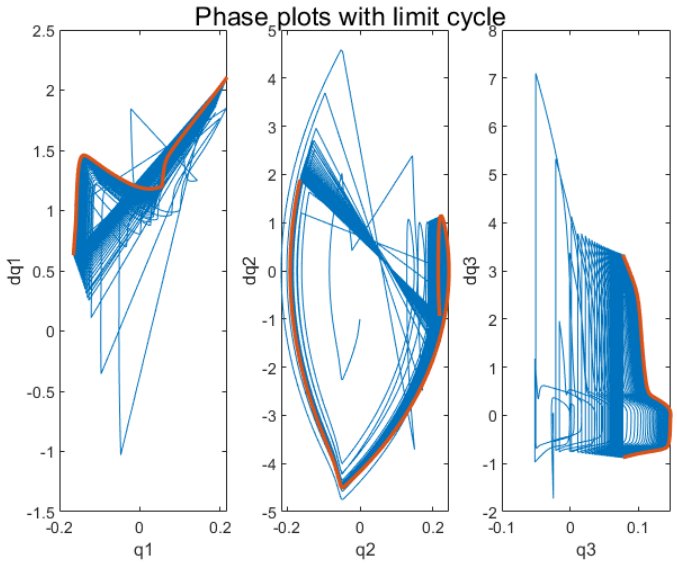


**FIGURE 3.6**
Phase plots and limit cycles of each joint angle relative to its angular velocity without perturbation

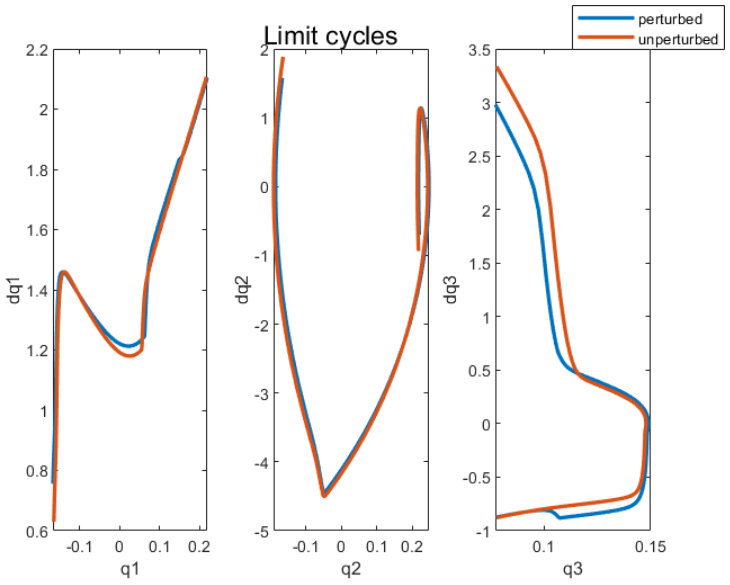


**FIGURE 3.7**
Limit cycles with unperturbed bot and with maximum applied external perturbation (0.2) at 0.63 m/s speed

## 3.5 CONTROLLER COMPARISON

Below we can see the limit cycles for both controllers at similar velocities. $q_1$ variation is somewhat similar, but we see significant differences in $q_2$ and $q_3$ behavior. In particular, $q_3$ is fairly limited for the neuronal controller as compared to the PD controller, but this makes $q_2$ less controllable, whereas PD controller uses considerable actuation to keep $\dot{q}_2$ at a sufficient value which comes at the expense of increase in $q_3$ variation.

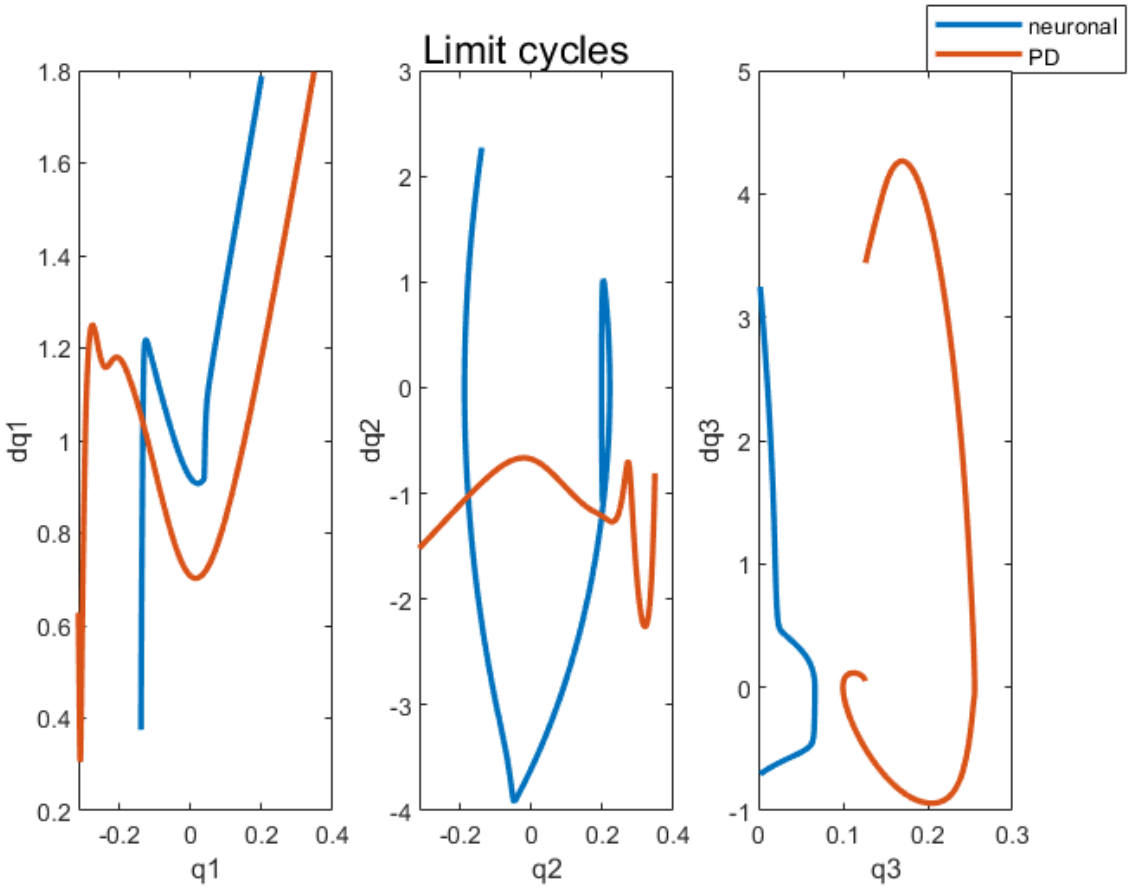


**FIGURE 3.8**
Comparison of limit cycles for the two controllers at comparable velocities

# CHAPTER 4

# DISCUSSION

## 4.1 PD CONTROLLER

The strength of the PD controller lies in the simplicity of the design. Without the need of any tracking or complex controller, we can reach a fairly large range of stable velocities with varying parameters. A PD controller has the added benefit of computational efficiency - this is less of an issue with a simple simulation but in a more complex model this method of control is well-suited. In a real system a PD controller won't eliminate steady state error, but in the perfect environment of a simulation this isn't a concern. The weaknesses we noticed are mainly the sensitivity of the controller to initial conditions and internal perturbations. In a real system, PD might be insufficient if these perturbations are a concern since it is not well suited to fast responding systems.

## 4.2 NEURONAL CONTROLLER

This controller is primarily a reflex based controller. One of the most important things to notice is that the controller parameters do not need any information about the dynamic properties of the robot. The only information from the environment used is the sensor values. This makes the controller design quite universal for different types of models. Another interesting property is the gait which looks very natural owing to its relation with biologically inspired controller. This type of controller has been applied in a real system, RunBot, which has a similar two dimensional restriction to this simulation. Another advantage gained by reflex based control is minimal reliance on the feedback compared to the PD controller. This can also be used to utilize the natural dynamics of the legs without actuation such as in passive walkers, which has been demonstrated in RunBot. However, the lower range of achievable velocities makes this control architecture less attractive.

## 4.3 CONTROLLER COMPARISON

At similar velocities, an immediate difference in control methods is apparent in the animations. The torso under the neuronal controller is maintained at a steady angle, whereas the PD controller oscillates the torso slightly to provide forward momentum. In some ways, having the torso so tightly controlled detracts from the efficiency of the neuronal controller; in contrast, the PD controller is dependent on the torso's movement for forward momentum.

Since the neuronal controller acts through limited feedback, it can easily converge to a stable limit cycle for a range of initial conditions. But the PD controller needs comparatively more precise initial

drive in order to exploit the torso's momentum before reaching steady state.

The controllers have very different ranges of achievable velocities. The PD controller is able to achieve a larger range of speeds than the neuronal controller mainly because most of the control input in the neuronal controller is used in order to stabilize the torso. Instead, as aforementioned, the PD controller is also exploiting the torso's momentum during the gait, which leads to higher possible speeds.

The controllers have completely diverging weaknesses in terms of response to perturbations. Since the PD controller is strongly dependant on feedback, perturbations in sensing have a significant effect on its performance. However, the neuronal controller uses events as a feedback and needs only two reflexes during one cycle, the ground impact reflex and swing leg extension reflex. Thus, sensing perturbations have a very small effect on the performance. When considering external perturbations the controllers switch rankings. The PD controller is incredibly robust to external perturbations in the format we are applying them. The neuronal controller is more sensitive to external perturbations by several orders of magnitude, which may also be due to the method of applying perturbations to the torso.

For similar velocities, the cost of transportation is lower for the PD controller than for the neuronal controller; this COT difference becomes more noticeable at higher velocities. This may be due to the control strategy for the torso; the neuronal controller spends a large portion of its control input stabilizing the torso without using that motion to its advantage. In contrast, the PD controller uses the torso in a more efficient manner for locomotion.

Both controllers could be reasonably applied to a real system depending on the system's limitations. In a two-dimensional walking robot, neuronal control has already been quite successfully demonstrated in RunBot. However a pure PD controller may not be sufficient for a non-ideal environment. In a controlled environment with a two-dimensional robot PD could be an adequate control strategy.

# CHAPTER 5

# CONCLUSION

## 5.1

Using a kinematic and dynamic model of a three-link biped we were able to simulate locomotion using two different feedback-based control architectures. The PD controller functions by controlling the position of the torso and the relative angles of the limbs. The neuronal controller, based on the control structure for RunBot, operates on a reflex based system which dictates how the stance and swing legs switch based on extension, while the torso is controlled similarly to the PD method via position control. The two controllers have quite disparate strengths when sensitivity to initial conditions, achievable velocities, and robustness to both internal and external perturbations. In terms of efficiency, for a similar velocity the PD controller has a lower cost of transportation than the neuronal controller, likely due to its use of the torso for forward momentum. When considering applying controllers to a real system, however, the neuronal controller has been demonstrated as effective whereas the PD controller's usability would be highly dependent on the environment. Overall, both control architectures have strengths suited to specific usages and environments and neither can be considered to be universally superior.

# BIBLIOGRAPHY

Geng, Tao, Bernd Porr and Florentin Wörgötter (Mar. 2006). 'Fast Biped Walking with a Sensor-driven Neuronal Controller and Real-time Online Learning'. en. In: *The International Journal of Robotics Research* 25.3, pp. 243–259. ISSN: 0278-3649. DOI: 10.1177/0278364906063822.