# Control Systems Descriptions for the Self-Balancing Unicycle Robot

## Introduction and problem description

In order to construct a proper controller scheme for this project, the team had to identify the inputs and outputs of the system as well as to have an idea of how to couple the states in question. Although several approaches to this type of control systems has already been done [1][2][3][4], after research and careful considerations the team decided to create an original and innovative design that would model a human riding a unicycle by applying torque to systems to change the yaw and pitch angles. The team decided to model the plant system using a non-linear systems approach that would then be linearized and discretized by the real-time controller. Using a state-variable feedback approach, the status of the various dynamic states can be tracked and a control effort can be computed to achieve a steady-state behavior [5].

## Using the state-variable feedback system

State variable feedback is a commonly used method in modern control systems since it allows the placement of poles anywhere for the system to reach steady-state [5]. After modeling the plant system using Lagrangian equations, the states, inputs, and outputs of the system would then be parsed and linearized using the LabVIEW control and simulation toolkit.

|  |
| --- |
|  |
| Figure 1. **Flow diagram of a basic discrete-time state variable system.** |

In the flow diagram, the feedback gain matrix K is used to place the closed loop poles while the prefilter Gpf is used to reduce steady-state oscillations which may lead to errors [5]. The state vector *x(k)* contains all the states for our system. The states in question are the lateral position of the system, the lateral velocity of the system, the angular position of the system, and the angular velocity of the system.

|  |
| --- |
|  |
| Figure 2. **Unicycle system model in LabVIEW.** |

## Selecting a controller type: LQR and Ackermann Pole Placement

In modern control systems, the steady-state time, the controllability, and number of states are deciding factors when choosing the type of controller for a non-linear system. The equations of motion for a unicycle are highly non-linear and contain couplings between different motions [1]. After linearizing the non-linear system in LabVIEW, the state-space model contains only states and not a non-linear set of state equations. Unfortunately, linearizing the system of equations gets rid of the couplings which reduce the robustness of the system [1].

For the unicycle control system, a Linear Quadratic Regulator (LQR) control and an Ackermann pole placement control were implemented as viable solutions for the controller.

### Using the Linear Quadratic Regulator (LQR) control approach

The LQR control is an optimality procedure that minimizes the quadratic cost function of the inputs and outputs to return the optimal state feedback vector that stabilizes the system by assigning a weighting mechanism to the states [3]. The LQR algorithm assigns weights on the control effort, not matching the reference signal at the final time, and not matching the reference signal during the transient time [5]. For this type of controller, the system must be linearized and the weights, or penalties, are quadratic [5]. The cost function that minimizes the penalties on the states is:

Where x is the state vector and u is the input vector, in this case the torque applied to the motors [3]. Finally, Q and R are state vectors that set the relative weights of the various states and the input [3.]. Since the unicycle system contains four states as mentioned earlier, Q is then a 4x4 matrix while R is a scalar. The cost function has to be set so that the weights have a greater magnitude towards the angular states than the lateral translation of the system.

### Using the Ackermann pole placement approach

In the other hand, Ackermann pole placement is an approach that allows us to identify the controllability matrix assuming that the system is controllable [5]. By finding the controllability matrix, the state-feedback gain matrix can then be found which will enable the system to reach a controlled steady-state response.

Starting from the discrete-time state variable description of a plant system in a state variable model, some procedures from the Cayley-Hamilton theorem [5] are taken to form a controllability matrix *K*.

For a system with a state variable feedback where the input u(k) is a scalar input, the input then becomes scaled. A new description of the plant can then be derived from the new input [2].

Now utilizing the same approaches as the Cayley-Hamilton theorem and rearranging terms the Δ() then becomes the following:

The controllability matrix from the above equation is then given as

From this, the equations can then be rearranged to find *K* and compute the controllabilitymatrix from then known *G* and *H* state matrices.

## LabVIEW implementation of the controller

The first step to implement the dynamic controller design is to import the Langrangian model description into a subsystem VI as shown in Figure 3. The plant in this case is modeled as a set of non-linear states where the coupling between lateral translation and angular information is shown.

|  |
| --- |
|  |
| Figure 3. **Lagrange model description of the unicycle system as a set of non-linear states.** |

The outputs of the two main states are *d2xPos* and *d2alpha*. To extract the position and velocity information, an integrator is applied to each of the states to which serve as input to the system. After developing the plant model, the LQR, or Ackermann pole placement, algorithm can then constructed to generate the control efforts for the plant. These control efforts are set to hit a limit for safety precautions. In the LabVIEW environment, the control efforts affect the simulated model of the plant which can then be controlled using either LQR or Ackermann controllers.

## Deployment of the controller in the LabVIEW environment

Using modern control approaches, the controller implemented in LabVIEW would set the control efforts so that the RoboteQ motor controller can set the necessary speeds for each of the motors so that the system can achieve steady-state. As the motor controller set new speeds for the hub motors, the system will start to lean or respond to the changes of in the plant (i.e. changes in torque which in turn affect the moments of inertia of the plant). Acting as the feedback sensor, the IMU would then capture the changes in angular position and velocity and feed the new states as inputs to the controller. Thus the whole process is repeated until the system reaches a steady-state.

|  |
| --- |
|  |
| Figure 4. **Controller diagram depiction in the LabVIEW environment.** |

From the controller description shown above, the angular position of the unicycle was given greater weight in the LQR algorithm by setting a higher penalty value (i.e. a value of 100) than the angular position of the unicycle. A scalar value of 0.01 is used for the R vector that multiplies the input vector of the plant (i.e. the torque values). The optimal gains for the motors are then computed by the controller.

In the LabVIEW block diagram panel, the controller was created using the Control and Simulation toolkit which allowed us to create control subsystems. Our main control subsystem is modeled as a finite state machine.

By starting from an initial position upon enabling the controller (i.e. an approximately upright position), the system will decide if it is still within the operational bounds of the safety emergency fail safe. The safety emergency fail safe range for our robot was to be ±10o from completely upright. If the system is within the bounds, the system will proceed to the Up-Up Gain state where the controller computes the controller effort necessary to achieve steady state. If the system is out of bounds, however, the system will then pass to the emergency state. When the unicycle passes to the emergency state, power to the motors is cut-off to prevent damage to the system.

|  |
| --- |
|  |
| Figure 5. **Upper layer block diagram layout of the state machine controller using the LQR algorithm.** |

|  |
| --- |
|  |
| Figure 6. **Inner layer block diagram layout of the state machine controller.** In this state-machine algorithm, the controller makes the decisions whether to go into emergency state or go into Up-Up gain state. |

## Observing the steady-state response of the model

Using LabVIEW, real-time data can be acquired from the controller to determine the stability of the system, steady-state impulse response, Bode magnitude and phase, as well as the poles of the system. Using the LQR algorithm description as mentioned above, the following data was captured.

|  |
| --- |
|  |
| Figure 7. **Controller real-time status display of stability, settling time, and impulse response.** |

For the system set-up using the LQR parameters described earlier, the settling time of the system is 3.72371 seconds. It shows that the system is stable and thus stead-state can be reached by the system. The impulse response shows the overshoot and the settling response after oscillating. This controller does not, however, take into account systematic and non-systematic errors that are encountered in a real-life scenario.

After using the controller, the simulated model can display the control effort and behavioral response of the model after applying the torque to the plant. The figure below displays how the system, although unstable, oscillates back and forth in an attempt to achieve a steady balanced position. The simulation also gives feedback information for the angular position and angular velocity of the unicycle. For this simulated model capture, the run-time achieve was 13 seconds.

|  |
| --- |
|  |
| Figure 8. **Simulated response of the controller using the LQR algorithm.** The state status (i.e. Linear Balance) indicates that the controller being used is taking into account the plant system and either LQR algorithm or Ackermann pole placement. |

|  |
| --- |
|  |
| Figure 9. **Emergency state triggered after system reaches the out-of-bounds control range.** In this state, the control signal is shown to be zero since no power to the motors is cut-off to prevent unwanted damage. |

## Alternative PD controller option explored to validate simulation capabilities

Working in parallel with advanced LQR and Ackermann pole placement approaches, a PID controller was also developed to demonstrate a unicycle system in a balanced state. This system does not alter the plant (i.e. it has a static model description) and it is not meant to be the final controller deployed on the unicycle.

For this controller, a proportional gain (P) of 10 was used with a derivate gain (D) of 3. Due to the fast sampling rate of the controller of 10 ms, the integrator error (I) makes the system become unstable way to quickly before it can compensate for the error. As a result, the integrator gain was omitted from the PID controller thus technically becoming a PD type controller. While using this system, no information for the plant or impulse response is updated since it is isolated from the plant model.

|  |
| --- |
|  |
| Figure 10. **Implementation of a PD controller to demonstrate steady-state response.** After 24 seconds of simulation, the system starts to show signs of converging to steady-state. |

## REFERENCES

[1] Kappeler, Fabian. *Unicycle Robot.* June 28, 2007.

[2] A. Kadis, D. Caldecott, A. Edwards, M. Jerbic, R. Madigan, M. Haynes, B. Cazoolato, Z. Prime, *Modeling, Simulation, and Control of an Electric Unicycle*, University of Adelaide, Australia.

[3] D’Souza-Mathew, Neil, *Balancing of a Robotic Unicycle*, 2008

[4] S. Majima, T. Kasai, *A Controller for Changing the Yaw Direction of an Undergraduate Unicycle Robot,* University of Tsukaba, Japan.

[5] R. Throne, *Discrete-Time Control Systems*, Rose-Hulman Institute of Technology, Chapter 6, 7, and 11.