Independent Study Report

Line following with a Differential Drive Mobile Robot using the Zeigler-Nichols PID tuning method

Jeremy Browne

I. Nomenclature

$\dot{X}^I$ – x velocity matrix for the *Inertial Coordinate System* for the DDRM

$\dot{X}^r$ – x velocity matrix for the *Robot Coordinate System* for the DDRM

$\dot{x}_I$, $\dot{y}_I$ – x and y velocity components for the *Inertial Coordinate System*

$\dot{x}_r$, $\dot{y}_r$ – x and y velocity components for the *Robot Coordinate System*

$\dot{\theta}_I$, $\dot{\theta}_r$ – Angular velocity of the DDMR for the *Inertial and Robot Coordinate System*

v – Velocity of the DDMR

v_R, v_L – Tangential velocity of the right and left wheel respectively for the DDMR

$\dot{\varphi}_R$, $\dot{\varphi}_L$ – Angular velocity of the right and left wheel respectively of the DDMR

ω – Angular velocity of the DDMR

R – Radius of the DDMR wheels

L – Half Length of the DDMR drive axel

$R(\theta)$ – Rotational matrix for the DDMR position analysis

k_p – Proportional gain

k_i – Integral gain

k_d – Derivative gain

$u(t)$ – Correction term generated from the controller law

$e(t)$ – Error term used for the controller law

T_i – Time integral term

T_d – Time derivative term

T_u – Ultimate period term

k_u – Ultimate gain

$AR(t)$ – Amplitude ratio

II. Introduction

A Differential Drive Mobile Robot (DDMR) is a type of wheeled robot used in many industrial and academic applications such as an educational tool for a controls projects. One application of such a vehicle is in self-controlled path following using closed-loop feedback control, which is one focus of this paper. The other focus is to implement an auto-tuning method known as the Zeigler-Nichols (ZN) tuning method, in which the DDMR will be capable of determining its own controller parameters. This system will be evaluated through simulation. First, the kinematics of a DDMR will be modeled using MATLAB code to characterize and predict the behavior the vehicle. Next, a closed-loop feedback controller will be implement providing a line following capability to the model. The control system will initially use a manually tuned controller. Lastly, the control system will be updated to use the Ziegler-Nichols (ZN) auto-tuning method so that the system may determine its own controller parameters. The two methods will then be compared.

III. Background

A. DDMR Kinematics

The DDRM model, shown in Figure (1), exists in two reference frames, the local reference of the DDMR known as the *Robot Coordinate System* with the origin on the vehicle itself, and the global reference frame known as the *Inertial Coordinate System* whose origin is fixed with the environment. To build the kinematic model, the first step is to define the position of the DDMR in both reference frames as well as how to map between each frame. Please review Ref (1) for the derivation of mapping between the Robot and Inertial reference frames for the DDMR. Here the kinematics of the DDMR, also defined in Ref (1), will be presented.

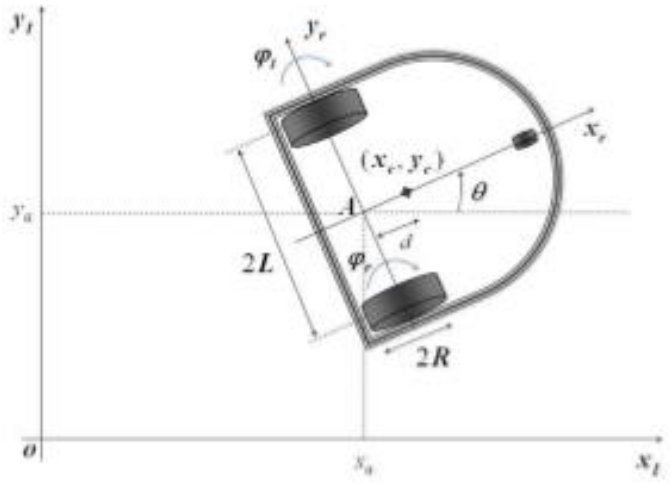

Figure (1): Differential Drive Mobile Robot model (Image taken from Ref(1))

A DDMR will move based on the speed of each wheel defined by

$$v_R = R\dot{\varphi}_R, \tag{1}$$

$$v_L = R\dot{\varphi}_L, \tag{2}$$

where v_R and v_L are the tangential velocities for the right and left wheels respectively, R is the wheel radius and $\dot{\varphi}_R$ and $\dot{\varphi}_L$ is the angular velocity for the right and left wheels respectively. The linear velocity of each

wheel will then in turn determine the velocity v of the DDRM which is defined by the average of the two wheels:

$$v = \frac{v_R + v_L}{2}. \tag{3}$$

The angular velocity ω of the DDMR is then defined as

$$\omega = \frac{v_R - v_L}{2L} \tag{4}$$

where L is half the length of the drive axel.

The velocity of the DDRM can now be expressed in both the Robot and Inertial reference frames with the following relationships:

$$\begin{bmatrix} \dot{x}_r \\ \dot{y}_r \\ \dot{\theta}_r \end{bmatrix} = \begin{bmatrix} \frac{R}{2} & \frac{R}{2} \\ 0 & 0 \\ \frac{R}{2L} & -\frac{R}{2L} \end{bmatrix} \begin{bmatrix} \dot{\varphi}_R \\ \dot{\varphi}_L \end{bmatrix} \tag{5}$$

$$\begin{bmatrix} \dot{x}_I \\ \dot{y}_I \\ \dot{\theta}_I \end{bmatrix} = \begin{bmatrix} \frac{R}{2}\cos\theta & \frac{R}{2}\cos\theta \\ \frac{R}{2}\sin\theta & \frac{R}{2}\sin\theta \\ \frac{R}{2L} & -\frac{R}{2L} \end{bmatrix} \begin{bmatrix} \dot{\varphi}_R \\ \dot{\varphi}_L \end{bmatrix} \tag{6}$$

$$X^I = R(\theta)X^r \tag{7}$$

where $\dot{x}_r$, $\dot{y}_r$, and $\dot{\theta}_r$, are the x, y, and angular velocities of the DDMR in the Robot Coordinate System. Conversely, the $\dot{x}_I$, $\dot{y}_I$, and $\dot{\theta}_I$, are the x, y, and angular velocities of the DDMR in the Inertial Coordinate System. The two systems are related by the transform defined by equation (7). The y velocity of the Robot reference frame is zero due to the non-holonomic constraints placed on the model which is described in Ref (1). The $R(\theta)$ term in Equation (7) is the rotational matrix needed to perform the transformation between the two reference frames and is also described in the position analysis in Ref (1) . Equation (6) then yields the following system of equations used in the MATLAB model:

$$\dot{x}_I = \frac{R\cos\theta}{2}(\dot{\varphi}_R + \dot{\varphi}_L) \tag{8}$$

$$\dot{y}_I = \frac{R\sin\theta}{2}(\dot{\varphi}_R + \dot{\varphi}_L) \tag{9}$$

$$\dot{\theta}_I = \frac{R}{2L}(\dot{\varphi}_R - \dot{\varphi}_L) \tag{10}$$

B. Tuning Methods

The Ziegler-Nichols (ZN) tuning method is an auto-tuning method for tuning a controller. This technique uses the stability of a controlled system to determine initial estimates for the controller parameters. The ZN method requires that the initial values of the proportional, integral, and derivative gains [k_p, k_I, and k_d respectively] of the controller to be set to zero. The k_p component is then steadily increased until a steady state response is seen. This k_p value is known as the ultimate gain, k_u , and represents the stability limit of the controlled system. The period of the steady state response is also recorded as the ultimate period T_u. These two values T_u and k_u are used to determine the k_p, k_i, and k_d values of the controller. The following table shows how each component is estimated:

Table 1: Ziegler-Nichols controller gains Ref (2)

Control Type	kp	Ti	Td
P	0.50 *ku*	-	-
PI	0.45 *ku*	*Tu*/1.2	-
PD	0.80 *ku*	-	*Tu*/8
Classic PID	0.60 *ku*	*Tu*/2	*Tu*/8

The results from the table are then used in the following equation,

$$u(t) = k_p \left[e(t) + \frac{1}{T_i} \int e(t)dt \ + T_d \frac{de(t)}{dt} \right] \tag{11}$$

with

$$k_i = \frac{k_p}{T_i} \tag{12}$$

$$k_d = \ k_p T_d \tag{13}$$

where $u(t)$ is the correction for the system, T_d and T_i are derivative and integral time respectively, and $e(t)$ is the system error. The value of k_u is then determined by the following relation:

$$k_u = \frac{1}{AR(t)} \tag{14}$$

where *AR* is the Amplitude Ratio of the input and output of the control loop. The k_u value is continuously updated as the system performs until k_p becomes greater than k_u. Once this occurs, the ultimate period, T_u, is recorded and the rest of the controller gains are calculated using the above table.

IV. Methods

A. MATLAB model

The DDMR was modeled in MATLAB using classes for each major component of the vehicle. These components include a path plotting class, DDMR class, controller class, and the Infra-Red (IR) sensor class. Each class controls a specific aspect of the model. The path plotting class creates the line the robot

will follow, in this case a sine wave. The DDMR class controls the kinematics of the robot and includes the dimensions of the robot, which in this case the model has a 5-inch-long drive axel and a 2.5-inch wheel radius. The controller class implements all the PID control and the ZN auto-tuning. The error for the feedback loop of the model is also calculated in the controller class using a MATLAB function called InterX, which calculates the intersection between two lines. The InterX function is used in this model to calculate the intersection of the modeled IR sensor and the to-be followed path. The intersection value is then compared to the target value, in this case the middle of the IR sensor, to generate an error for the controller class. The controller class will then calculate a value to correct the DDMR's motion using one of the control methods described in Table (1). The correction is then used to modify the velocity of each wheel of the DDMR as follows:

$$v_R = R\,(\dot{\varphi}_R + u(t)), \tag{15}$$

$$v_L = R\,(\dot{\varphi}_R - u(t)). \tag{16}$$

The kinematics of the DDMR are then calculated for the new wheel speeds.

The MATLAB code has two modes, the first is to run the simulation using a manual tuning method for the controller gains, and then the ZN method. The code first generates a virtual environment with a path for the DDMR to follow. For this report, the path is a sine wave. Next, the DDMR model is placed in this environment, away from the path, moving at prescribed constant speed, until the path is found. Once the path is found, the ZN method calculates the ultimate gain k_u by constantly calculating an ultimate gain value while increasing the proportional gain k_p (note that all the controller gains were initially set to zero). During the simulation, the proportional value is continuously compared to the absolute value of the calculated ultimate gain. Once the proportional value is larger than the ultimate gain, the ultimate gain takes that value and the other gains are calculated as stated in Table (1). While the ultimate gain remains static after this point, the ultimate period T_u is continuously calculated for the duration of the simulation to preserve stability as the robot follows the path. The simulation would fail, with the robot rejecting the path, if the ultimate gain is also continuously updated.

B. Robot Validation

To determine the success of the tuning method the simulation of the DDMR's controller gains k_p, k_i, and k_d were first manually tuned and then compared to the ZN auto tuning method. To accomplish this, the method imposed was to start with a proportional controller with a gain k_p equal to 0.5, 1, 1.5, then 2. After finding that a $k_p$of 2 having the most success in following the path, it was used for a PI, PD, and PID manual tuning. For example, while testing the PI controller under a manual tuning condition, the proportional gain is set to 2, while the integral gain was incremented from 0.5 to 2 in 0.5 increments. The same was done from the PD controller where the derivative gain was varied. Lastly, for the PID controller, the proportional gain is set to 2, while the other two gains are incremented from 0.5 to 2 in 0.5 increments in succession. The ZN tuning method was then run for each controller type for comparison. Furthermore, the target point of the path to be in the middle of the IR sensor is also periodically updated to see how well the controller can accommodate for a new setting. In the results presented, the target value can be seen increasing from 0.5 to 0.6 (of the length of the IR sensor) after 300 simulated seconds.

V. Results

Figure (2) presents the manual tuning and auto tuning method for a proportional controller. From the figure, it can be seen that a proportional gain of 2 is the most effective at following the path while the auto tuning method is slightly better. The large spikes in the error response is due to the curvature of the sine

wave shaped path. The only problem presented by the ZN method is a large overshoot seen right as the robot finds the line (time zero). This problem persists for each controller type. The target value is also plotted to illustrate how well the controller can accommodate for this change.

Figures (3) - (5) demonstrate the effectiveness of the ZN tuning method versus a manual method for a PI, PD, and PID controller respectively. These tuners already demonstrate an improvement in following the path eliminating the large spikes seen for the P controller in Figure (2). It can also be seen for each of these controller types that the ZN method is just as good as its manually tuned counter-part. For example, in Figure (4), the ZN method for a PD controller is correcting just as quickly as the manually tuned controllers and in Figure (3) for the PI controller the ZN method is showing less noise.

Lastly, since the ZN method has proven an effective way to determine controller gains for the line following DDMR, the system is stress tested for a model that increases speed as well as the target value over time. Figure (6) shows the results for a P, PI, PD, and PID controllers using the ZN method. From the figure, it can be seen that a PI controller is the most effective at reducing error and keeping the vehicle on the line, however. The controller gains for each controller type are presented in Table (2) for the increased speed test.

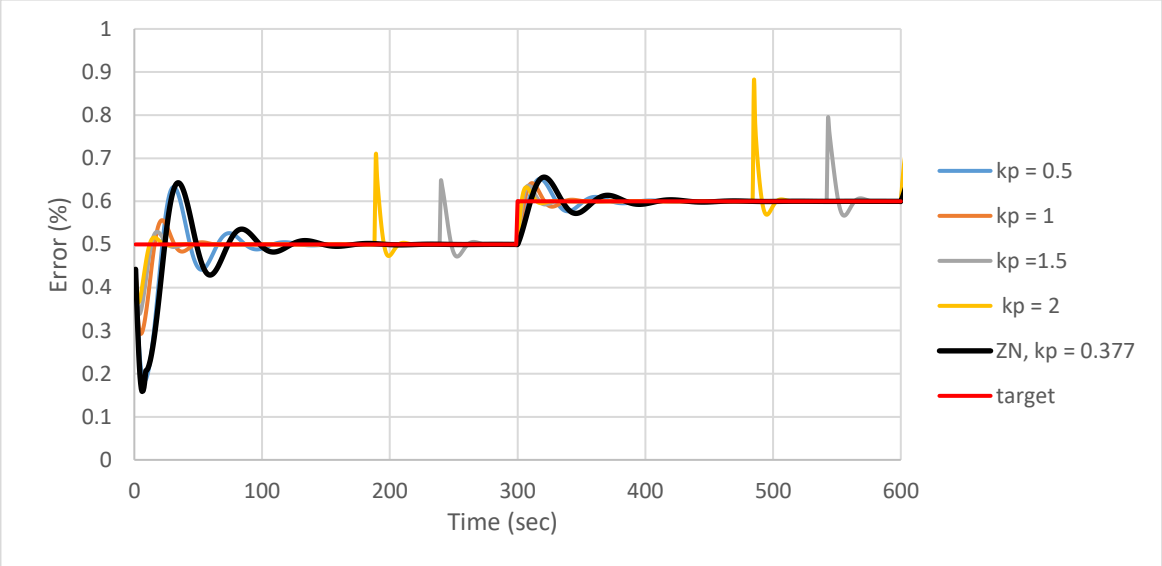

Figure (2): Manual and Auto-Tuning results for a Proportional (P) controller for a DDMR following a sine wave path.

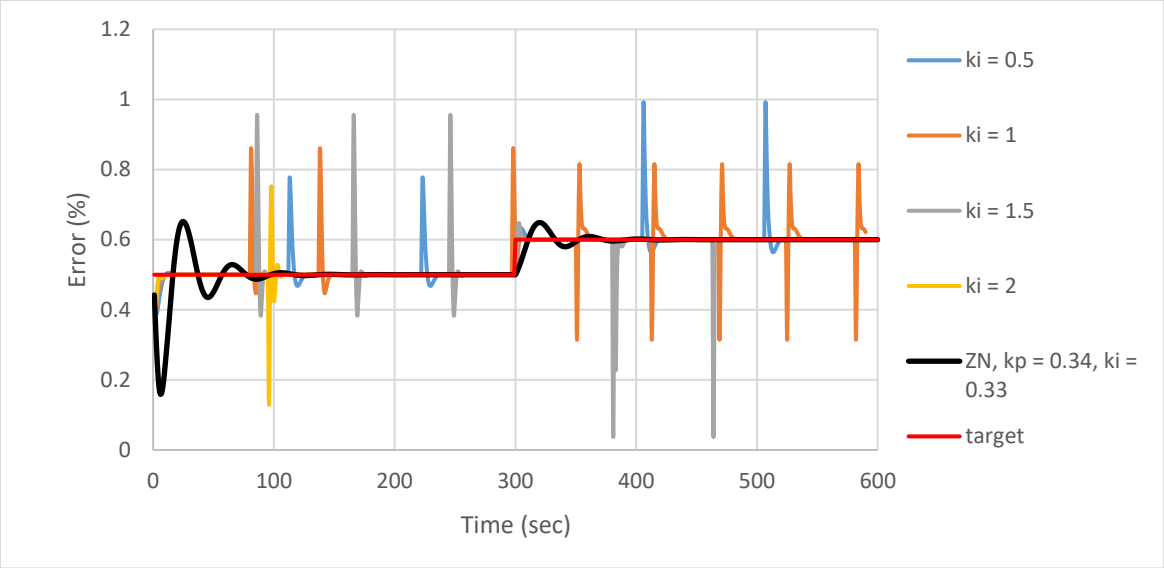

Figure (3): Manual and Auto-Tuning results for a Proportional-Integral (PI) controller for a DDMR following a sine wave path. Note that P = 2 and D = 0

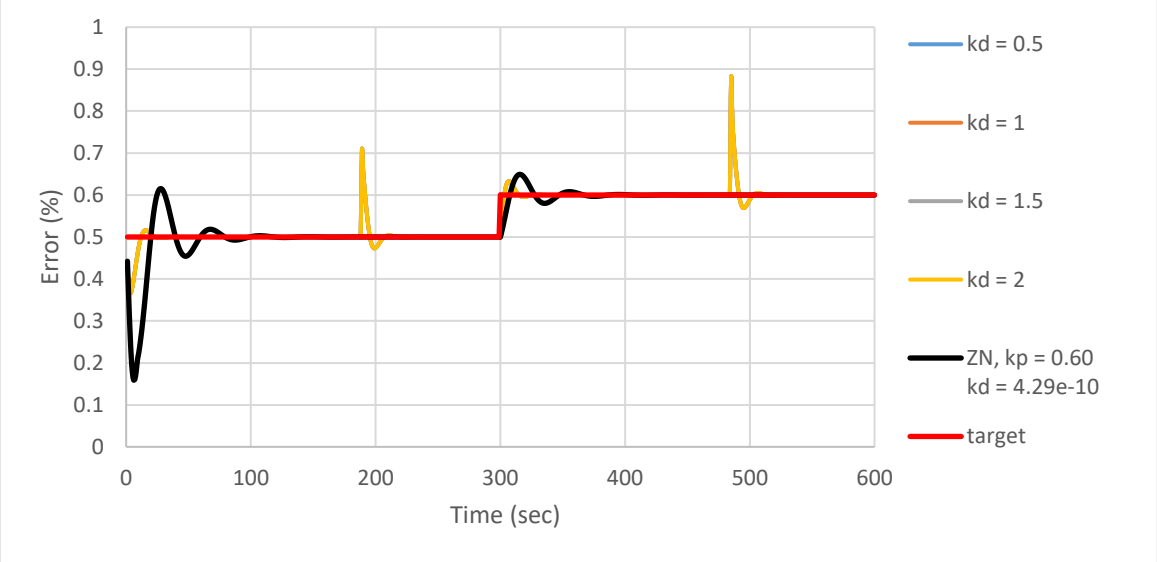

Figure (4): Manual and Auto-Tuning results for a Proportional-Differential (PD) controller for a DDMR following a sine wave path. Note that P = 2 and I = 0

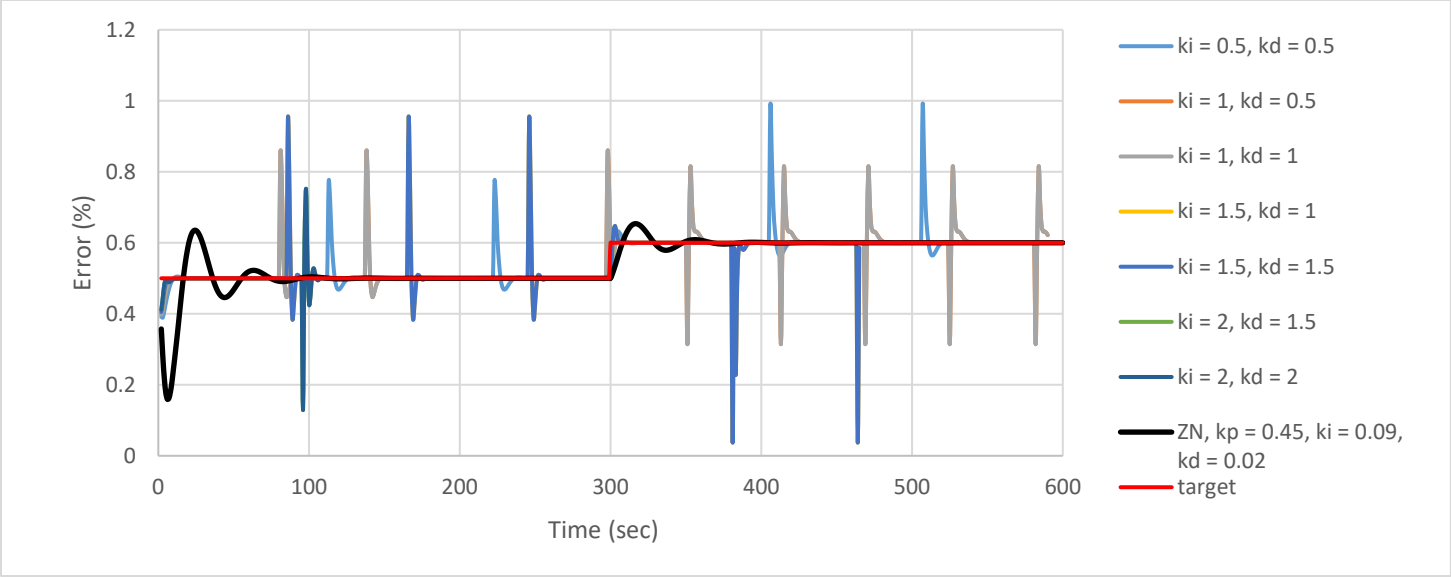

Figure (5): Manual and Auto-Tuning results for a Proportional-Differential-Integral (PID) controller for a DDMR following a sine wave path.

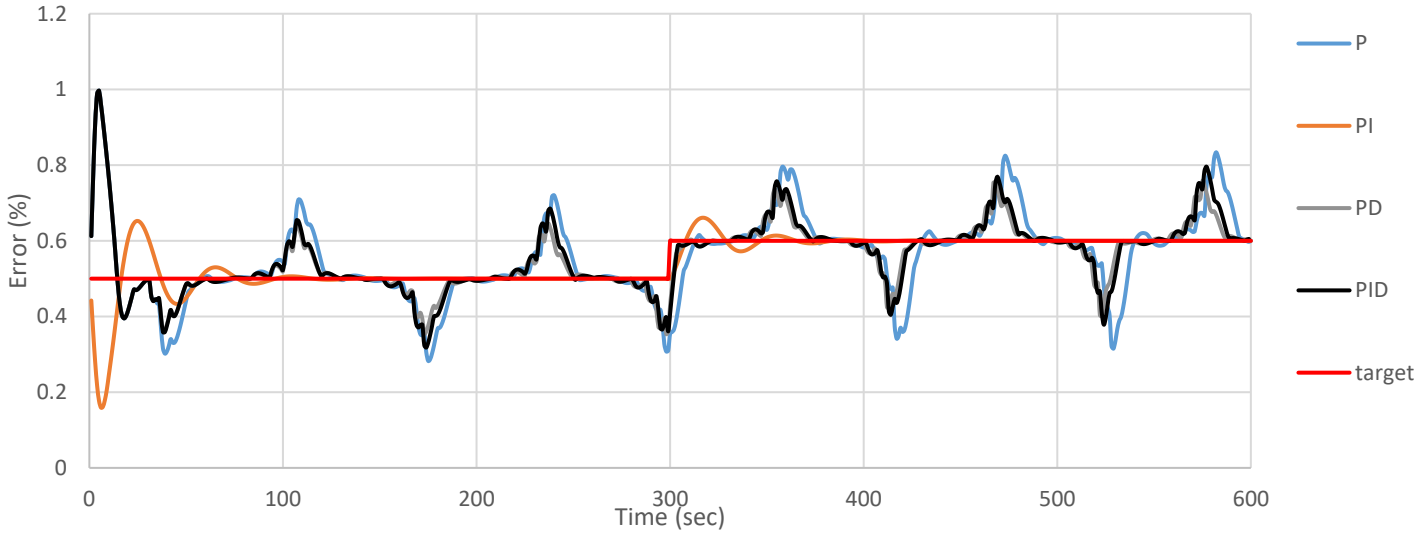

Figure (6): Ziegler-Nichols results for P, PI, PD, and PID controllers for a simulation where vehicle speed increased over time.

Table (2): kp, ki, and kd values generated for the Ziegler- Nichols for the P, PI, PD, PID

	kp	ki	kd	ku	Max Speed (in/sec)
P	1.4	0.00	0.00	2.8	5.2
PI	1.3	0.21	0.00	2.8	5.2
PD	2.3	0.00	0.30	2.8	5.2
PID	1.7	0.17	0.004	2.8	5.2

VI. Conclusion

Presented in this report are the results of how effective the Ziegler-Nichols method is at estimating controller gains for a controlled system. To do so, the kinematics of a Differential Drive Mobile Robot were modeled and used to follow a plotted path controlled using closed-loop feedback. The model and controller were both tested in a simulated environment using MATLAB. The robot model was first controlled using a manual tuning method for the controller gains. This method was then used to evaluate how effective the Ziegler-Nichols method is at estimating controller gains for the robot. The Zeigler-Nichols method was compared to the manual method for a P, PI, PD, and PID controller and the results show that the Ziegler-Nichols method was successful at predicting acceptable controller gains and successfully follow the plotted line.

VII. References

1. Dhaouadi, R., & Hatab, A. (2013). Dynamic Modeling of Differential-Drive Robots using Lagrange and Newton-Euler Methodologies: A Unified Framework. *Advance in Robotics & Automation.* Autom 2:107 doi: 10.4172/2168-9695.1000107
2. Bennett, J., & Bhasin, A., & Grant, J., & Wen, C. (2006, October 19). *PIDTuningClassical*. Retrieved from https://web.archive.org/web/20080616062648/http://controls.engin.umich.edu:80/wiki/index.php/PIDTuningClassical#Ziegler-Nichols_Method
3. (2018, April 22). *Ziegler-Nichols method.* Retrieved from https://en.wikipedia.org/wiki/Ziegler%E2%80%93Nichols_method