Anti-windup PI controller, SIPIC for Motor Position Control

Citation for published version:
Wong, KK, Hoo, CL & Mohyi, MHH 2018, 'Anti-windup PI controller, SIPIC for Motor Position Control', MATEC Web of Conferences, vol. 152, 02022. https://doi.org/10.1051/matecconf/201815202022

Digital Object Identifier (DOI):
10.1051/matecconf/201815202022

Link:
Link to publication record in Heriot-Watt Research Portal

Document Version:
Publisher's PDF, also known as Version of record

Published In:
MATEC Web of Conferences

General rights
Copyright for the publications made accessible via Heriot-Watt Research Portal is retained by the author(s) and / or other copyright owners and it is a condition of accessing these publications that users recognise and abide by the legal requirements associated with these rights.

Take down policy
Heriot-Watt University has made every reasonable effort to ensure that the content in Heriot-Watt Research Portal complies with UK legislation. If you believe that the public display of this file breaches copyright please contact open.access@hw.ac.uk providing details, and we will remove access to the work immediately and investigate your claim.
Anti-windup PI controller, SIPIC For Motor Position Control

Kah Kit Wong¹, Choon Lih Hoo², Mohd Hardie Hidayat Mohyi ¹,∗

¹School of Engineering, Taylor’s University, Malaysia
²School of Engineering and Physical Sciences, Heriot-Watt University, Malaysia

Abstract. Control system plays a major role in the industry nowadays as it simplifies workload and reduce manpower. Among all the controlled applicable field, control system is heavily used in motor speed and motor position controls. Although there are various types of controllers available in the market, PID controller remains as one of the most used controller due to its simplicity. Unfortunately, PID controller experiences windup phenomenon which affects the controller’s performance. This paper proposes a new type of anti-windup PI controller, SIPIC for motor position control application and aims to validate the performance of this controller as compared to conventional PI controller. To test the ability of the controllers, both controllers were experimented using hardware testing. The settings conditions of with and without loadings were used under two different inputs of 0˚ to 90˚ and 270˚ to 90˚. The results obtained show that under without loadings, both controller showed favourable performances. Though, SIPIC controller slightly outperforms PI controller by having lower overshoot and shorter settling time for a wider range of gains. The rise time of both controllers are similar as it is the lowest possible rise time due to hardware limitations. Experiment results with loading condition, for both inputs and when Kp is 1 and Ki is 15, PI controller shows unstable performance by having large amount of oscillations and overshoots. The settling time was unable to be determined as the controller did not settle within the given step time. On the other hand, at the same gain, SIPIC controller still shows acceptable performance. This shows that SIPIC controller is more favourable by having better stable performance for a wider range of gains while PI controller needs to be finely tuned to a specific gain to obtain desired results.

1. Introduction

In the 21st century where manual manpower is increasingly being replaced by automatic robotic system, control system plays a major role in ensuring the quality of product and efficiency of the automated system.

* Corresponding author: hardie.hidayat@taylors.edu.my

© The Authors, published by EDP Sciences. This is an open access article distributed under the terms of the Creative Commons Attribution License 4.0 (http://creativecommons.org/licenses/by/4.0/).
The involvement of control system in the industry can mainly be found in speed control and position control. In position control system, mechanical position usually is the system output for an arm or a pointer. The controller performance highly affects the accuracy of arm/pointer system output. The position control system applications can be commonly found in automated systems such as solar tracking system and robotic system. Servomotor is usually used as the actuator for position control system. Servomotor is a type of direct current (DC) motor that is capable of precise control for velocity and acceleration as well as angular and linear positions. Many types of controllers have been used in the application of position control systems. These controllers usually include the use of fuzzy logic, fractional order, finite control with model predictive controllers as well as a proposed controller that combines the characteristics of \( \mu \)-synthesis and neural network [1].

The proposed controller is a modified crisp fuzzy logic controller where modifications were done on both the input and the output of the controller. The input of the controller uses crisp rectangular adjacent non-overlapping membership function which is distributed along the discourse universe whereas modification for the output was done on the computation method [1]. The controller showed acceptable performance and shorter processing time than conventional fuzzy logic controller [2].

Another proposed controller is based on finite control set model predictive control for the application of permanent magnet synchronous motor (PMSM). This controller predicts the future behavior of the actuator and perform action that is chosen from a model set [3]. Other than that, many also applied fractional order calculus in motor position control which uses a fractional order PI controller and compared to integer order PI controller. The fractional order controller contains two extra variables which allow more flexible and precise tuning in the applications. It can be concluded that fractional order PI controller experience less overshoot and lower maximum percentage of error than conventional PI controller [4]. Further expanded application of fractional order added with differential evolution algorithm is also introduced in PI controllers. This method adds a self-tuning ability to the controller for the application of voice coil motor (VCM). Result shows that the proposed controller experience lower settling time, tracking error as well as lower maximum overshoot as compared to conventional PI controller and standalone fractional order PI controller [5].

In this paper, Proportional-integral-derivatives (PID) controller is chosen due to its simplicity in applications [6]. By tuning the controller’s three individual components, overshoots, steady-state errors, rise times and settling times of the system can be altered easily. The limitation of a conventional PID controller is often caused by the limitation of the actuator being used instead of the controller itself. When the output of the controller exceeds the limit of the actuator, the controller begins to lose the ability to control the system. Example can be shown through the flushing mechanism. The system controls the opening of the valve to limit the flow rate of the water depending on the tank level. However, if the error continues to grow even after the valve is fully open, the flow rate of the water will not continue to increase due to the limitation of the valve. This condition is known as the integral windup phenomenon, where the integral component continues to offset the error even when the output remains the same.

Over the years, several anti-windup schemes had been proposed to eliminate the integral windup phenomenon and to increase the efficiency of PID controller. These schemes include the Conditional Integration (CI), Tracking Back Calculation (TBC), Integral State Prediction (ISP), pole placement method and gain scheduling method.
Conditional Integration is one of the most common schemes and it function by switching off the integral component during saturation state then turns it back on when the controller enters a linear state again. Tracking Back Calculation measures the difference between the output signal and the non-saturation state signal before feeding the error back to obtain higher accuracy. Another scheme is the Integral State Prediction (ISP). This scheme predicts the steady-state value and feed the value back to the integral state when the controller re-enters linear state [7].

Recently, both pole placement and gain scheduling methods were introduced in PID. The desired plant is first transformed into a type-1 plant as type-0 plant will always have error. Pole placement method is then used for the derivation of the controller gain [8]. Lastly, gain scheduling proposed the use of various gains and constants for different controller states for higher accuracy [9].

This paper uses a new type of controller known as Steady-state Integral Proportional Integral Controller (SIPIC) scheme for the application of position control system. Steady-state Integral Proportional Integral Controller (SIPIC) scheme is also another proposed controller that can overcome integral windup. It does that by having a separate closed loop feedback with the steady state integral value as an input. This constantly leads the integral control towards the steady state integral value[7]. It was mentioned earlier that different component of PID controller will have different effect on the performance and for PI controller, it will often have a short settling time with overshoot performance. SIPIC scheme made it possible to have a controller with short settling time with no overshoot performance due to its tuning decoupling ability. This ability allows the proportional gain ($K_p$) and integral gain ($K_i$) to be tune separately.

SIPIC has shown ideal performance with little overshoot while maintaining short settling time on a DC motor speed control application[7]. However, no work had been done on applying the SIPIC scheme on a DC motor position control application. Therefore, this research aims to validate the usage of proposed SIPIC controller in the application of motor position control. The objectives of this research are as below:

1. To develop a new anti-windup PI controller that is suitable for motor position control.
2. To validate the stability and performance of the new PI controller by studying the overshoot, steady-state error and settling time of the controller.
3. To investigate and determine the range of gain that will maximize the performance of the PI controller for motor position control.

2. **Research Methodology**

2.1 Experimental Setup

Hardware testing was done to verify the proposed SIPIC controller. Comparison was done between SIPIC controller and conventional PI controller. The performance of the controllers is evaluated based on theirs steady-state error, overshoot, rise time as well as settling time. SCILAB and SCICOSLAB software were used to carry out experiment and data analysis for both the controllers. Figure below shows the equipment used for experimental testing purpose.
Fig 1. Experimental setup

2.2 SIPIC

This section focuses on the SIPIC controller and its characteristics. Based on previous work done by [7], Eq. 1 shows the generic template where SIPIC was developed from. $Q(s)$, $q_{ss}$, $f(s)$, $s$ and $n$ represents Laplace form of integral state, integral during steady state, function in Laplace form, Laplace domain and non-negative integer respectively. $A$ and $B$ are constants. Eq. 2 below shows integral component of SIPIC after various substitutions and derivations while Eq. 3 shows the error equation of SIPIC.

$$ \frac{q_{ss}}{s} - Q(S) = As^n f(s) + B \quad (1) $$

$$ k_1 q_{ss} = v - \frac{b}{f k_T} j y + \frac{a}{f k_T} e \quad (2) $$

$$ E(S) = \frac{\alpha(0) [s + k_i] + k_i k_T [q_{ss} - q(0)]}{\left(\frac{1}{f} + \frac{\alpha}{f} + k_p k_T\right)(s + k_i) - k_i k_T} \quad (3) $$

2.3 Input Reference

Based on the previous work done on SIPIC controller, the input reference was modified for the application of motor position control. Fig. 2 shows the block diagram for motor position control. From this, Eq. 4 was derived, and Eq. 5 shows the final input reference for motor position control for SIPIC controller. Fig. 3 shows the SIPIC controller block diagram for experimental setup whereas Table 1 shows the parameter used to obtain the input reference.
This section focuses on the SIPIC controller and its characteristics. Based on previous work done by [7], Eq. 1 shows the generic template where SIPIC was developed from.

\[
X(s) = \frac{v_e(s) - \theta(s)k_m \ast \eta k_T}{Ls + R}
\]

\[
k_iq_{ss} = v_{ss} = v_\theta + \frac{LI}{\eta k_T} \dot{e} + \frac{BR}{\eta k_T} \dot{e} + \frac{\eta k_T k_m}{\eta k_T} e
\]

Fig. 2. Motor position control block diagram

Fig. 3. SIPIC experimental setup block diagram
Table 1. Parameters for experiment [7]

| Characteristics                        | Values                      |
|----------------------------------------|-----------------------------|
| Viscous damping coefficient, $B$       | $2.12 \times 10^{-4}$ kg m$^2$/s |
| Inductance, $L$                        | 0.005 H                     |
| Moment of inertia of motor, $J$        | $2.14 \times 10^{-5}$ kg m$^2$ |
| Torque constant, $K_T$                 | 0.09 Nm/A                   |
| Back-emf constant, $K_m$               | 0.09 Nm/A                   |
| Efficiency, $\eta$                    | 0.8                         |
| Resistance, $R$                       | 7.8 $\Omega$                |
| Moment of inertia of mild steel black plating | $8.63 \times 10^{-5}$ kg m$^2$ |

3. Results and Discussions

This section shows the results obtained from hardware testing. Necessary information was obtained from the extracted data and analysis was done to verify the performance of each controller. Comparison was done between the two controllers to verify the proposed SIPIC controller for motor position control under no load condition and under load condition.

Two different step inputs were used for testing. For the first step input, the controller travels from 0° to 90° and for the second input, the pointer travels from 270° to 90°. To put this into real life application, the first input is similar as to when changing steering direction of a vehicle while travelling in straight line while the second input is similar as to changing the direction of a vehicle from left directly to right. The stability in each case is different.

All the data obtained was tabulated below from table 2 to table 4. The table indicates

3.1 No Load

3.1.1 0 to 90

Table 2. Results for no load condition first input

| $K_p$ | $K_i$ | Overshoot | Rise Time | Settling Time |
|-------|-------|-----------|-----------|---------------|
|       |       | Pie       | SIPIC     | Pie           | SIPIC     | Pie | SIPIC |
| 1     | 1     | 0.132     | 0.05      | 0.078         | 0.086     | 1.737 | 2.34 |
| 1     | 5     | 0.452     | 0.21      | 0.064         | 0.091     | 0.504 | 0.671|
| 1     | 10    | 0.722     | 0.374     | 0.056         | 0.095     | 0.576 | 0.981|
| 1     | 15    | 0.949     | 0.487     | 0.05          | 0.094     | 0.93  | 0.762|
| 1     | 20    | 1.09      | 0.543     | 0.046         | 0.095     | 1.228 | 1.047|
| 2     | 1     | 0.251     | 0.204     | 0.04          | 0.042     | 0.156 | 0.632|
| 2     | 5     | 0.371     | 0.188     | 0.037         | 0.048     | 0.688 | 0.434|
| 2     | 10    | 0.521     | 0.295     | 0.035         | 0.055     | 0.175 | 0.333|
| 2     | 15    | 0.634     | 0.411     | 0.034         | 0.059     | 0.242 | 0.541|
| 2     | 20    | 0.663     | 0.502     | 0.034         | 0.061     | 0.239 | 0.486|
| 3     | 1     | 0.389     | 0.358     | 0.027         | 0.029     | 0.165 | 0.167|
| 3     | 5     | 0.452     | 0.242     | 0.027         | 0.033     | 0.161 | 0.352|
| 3     | 10    | 0.502     | 0.305     | 0.026         | 0.038     | 0.496 | 0.251|
| 3     | 15    | 0.524     | 0.367     | 0.026         | 0.043     | 0.396 | 0.233|
3.1. Results and Discussions

This section shows the results obtained from hardware testing. Necessary information was obtained from the extracted data and analysis was done to verify the controller for motor position control under no load condition and under each case is different.

To put this into real life application, the first input is similar as to when changing steering direction of a vehicle while travelling in straight line while the second input is 90˚. All the data obtained was tabulated below from table 2 to table 4.

| Parameter                  | Value 1 | Value 2 | Value 3 |
|----------------------------|---------|---------|---------|
| Torque constant, K_t       | 0.502   | 0.575   | 0.537   |
| Inductance, L             | 0.502   | 0.502   | 0.502   |
| Emf constant, K_e         | 0.502   | 0.502   | 0.502   |
| Moment of inertia of motor, J_m | 0.005 H | 0.005 H | 0.005 H |
| Viscous damping coefficient, B_v | 0.027 kg m² | 0.027 kg m² | 0.027 kg m² |
| Back EMF constant, K_b     | 0.502   | 0.502   | 0.502   |
| Characteristics            |         |         |         |
| PI                         | 0.367   | 0.502   | 0.502   |
| Overshoot                  | 0.019   | 0.019   | 0.019   |
| Rise Time                  | 0.019   | 0.019   | 0.019   |
| Settling Time              | 0.019   | 0.019   | 0.019   |
| Performance graph when K_p = 1, K_i = 1

![Fig. 4. Performance graph when K_p = 1, K_i = 1](image_url)
Fig 5. Performance graph when $K_p = 1, K_i = 20$

Fig. 6. Performance graph when $K_p = 20, K_i = 1$
Table 2 above shows the performance of both controllers when $K_p$ is 1 and $K_i$ is 1, 5, 10 and 15 respectively. As the integral gain increases, both controllers experience an increase in overshoot but SIPIC controller have a significantly lower overshoot especially at higher $K_i$ value. Moving on to the rise time of the controllers, different pattern is seen in each controller. For PI controller, rise time decrease as the integral gain increase but for SIPIC, rise time increase alongside the integral gain. However, SIPIC shows higher rise time compared to PI controller. Lastly, the settling time of the controllers is compared. Under comparison, when both the gain is 1, SIPIC shows a higher settling time. When the integral gain is increased to 5, both controller experience a big drop in settling time but as the integral gain continue to increase, settling time gradually increase in PI controller but experience minor up and down changes in SIPIC controller.

Next, $K_i$ is kept at 1 and $K_p$ is increased from 1 to 5, it is seen from results that the overshoot for both controllers increased significantly. Rise time and settling time decreases for both controller. Both controllers now experienced similar overshoot, rise time as well as settling time. As the integral value increases, PI controller experience slight increase in overshoot but SIPIC controller experience slight decrease in overshoot. The rise time of PI controller remain constant whereas SIPIC controller has slight increase in rise time when the integral gain increases. For settling time, PI controller experience slight increase. For SIPIC controller, when the $K_i$ value is increase from 1 to 5, there is an increase in settling time but setting time experience a descent when the $K_i$ value is further increased.

The $K_p$ value was then increased to 10. The increased of $K_p$ value causes the same effect to both controllers, increase in overshoot and settling time but decrease in rise time. SIPIC controller still has a lower overshoot than PI controller, similar settling time and both controller now have the same rise time. As the integral gain increase, overshoot for PI controller has a slight increase but a slight decrease is experienced by SIPIC controller. There are no changes in rise time for both controller as this is the lowest possible rise time due to hardware limitation. PI controller continue to experience slight increase in settling
time as $K_i$ increases while settling time for SIPIC controller first increase then decreases, like where the $K_p$ is 5.

Lastly, $K_p$ was increased to 15. The rise time for both controllers is still 0.019s, same as before. The increase in $K_p$ causes a slight increase in overshoot and settling time of PI controller but a slight decrease in overshoot and settling time of SIPIC controller. Increasing $K_i$ value continue to cause slight increase in overshoot of PI controller but decrease in overshoot for SIPIC controller. Same pattern is also seen in settling time of both controller. The increase in $K_i$ value results in increase of settling time for PI controller but for SIPIC controller, the settling time first increases then start to decrease when the $K_i$ value is further increased from 5.

Under no load condition first input, both controller shows very close performance. The proposed SIPIC controller have lower overshoot at all parameter as compared to conventional PI controller. SIPIC controller experiences higher rise time at low $K_p$ value but both controller experience similar rise time when $K_p$ value is further increased. Comparing the condition when both $K_p$ and $K_i$ is 15 as shown in Fig. 6, SIPIC controllers outperform PI controller by having a lower overshoot as well as lower settling time. This is because at higher $K_i$ value, SIPIC controller experience decrease in both overshoot and settling time but PI controller experience increase in both components.

### 3.1.2 270 to 90

| $K_p$ | $K_i$ | Overshoot | Rise Time | Settling Time |
|-------|-------|-----------|-----------|---------------|
|       |       | PI | SIPIC | PI | SIPIC | PI | SIPIC |
| 1     | 1     | 0.358 | 0.213 | 0.069 | 0.076 | 2.209 | 1.942 |
| 1     | 5     | 0.986 | 0.468 | 0.061 | 0.089 | 0.584 | 0.647 |
| 1     | 10    | 1.486 | 0.785 | 0.053 | 0.091 | 0.499 | 0.868 |
| 1     | 15    | 1.756 | 0.967 | 0.049 | 0.095 | 1.223 | 1.137 |
| 1     | 20    | 2.149 | 1.087 | 0.044 | 0.094 | 1.093 | 1.377 |
| 2     | 1     | 0.433 | 0.427 | 0.036 | 0.039 | 0.191 | 1.302 |
| 2     | 5     | 0.704 | 0.44  | 0.033 | 0.046 | 0.941 | 0.523 |
| 2     | 10    | 0.983 | 0.619 | 0.034 | 0.055 | 0.612 | 0.337 |
| 2     | 15    | 1.156 | 0.801 | 0.033 | 0.058 | 0.561 | 0.523 |
| 2     | 20    | 1.285 | 1.002 | 0.032 | 0.06  | 0.448 | 0.702 |
| 3     | 1     | 0.616 | 0.54  | 0.032 | 0.033 | 0.169 | 1.288 |
| 3     | 5     | 0.769 | 0.531 | 0.031 | 0.035 | 1.023 | 0.515 |
| 3     | 10    | 0.923 | 0.644 | 0.03  | 0.037 | 0.676 | 0.309 |
| 3     | 15    | 1.055 | 0.776 | 0.03  | 0.042 | 0.528 | 0.419 |
| 3     | 20    | 1.244 | 0.936 | 0.03  | 0.043 | 0.424 | 0.391 |
| 4     | 1     | 0.669 | 0.631 | 0.031 | 0.031 | 0.216 | 1.005 |
| 4     | 5     | 0.782 | 0.587 | 0.03  | 0.032 | 1.11  | 0.501 |
| 4     | 10    | 0.901 | 0.682 | 0.03  | 0.033 | 0.857 | 0.318 |
| 4     | 15    | 1.03  | 0.851 | 0.03  | 0.034 | 0.649 | 0.217 |
| 4     | 20    | 1.15  | 0.952 | 0.03  | 0.036 | 0.552 | 0.333 |
because at higher $K_i$ value, SIPIC controller experience decrease in both overshoot and outperform PI controller by having a lower overshoot as well as lower settling time. This is for SIPIC controller, the settling time first increases then start to decrease when the $K_p$ value but both controller experience similar rise time when $K_p$ value is further increased.

Increasing $Ki$ value continue to cause slight increase in overshoot of PI controller but PI controller but a slight decrease in overshoot and settling time of SIPIC controller. Like where the $K_p$ is 5.

Comparing the condition when both $K_p$ and $Ki$ is 15 as shown in Fig. 6, SIPIC controllers settling time but PI controller experience increase in both components.

The increase in $Ki$ value results in increase of settling time for PI controller but decrease in overshoot for SIPIC controller. Same pattern is also seen in settling time of both controllers at $K_i = 5$. Over time $K_i$ increases while settling time for SIPIC controller first increase then decreases,

Under no load condition first input, both controller shows very close performance. Lastly, $K_p$ was increased to 15. The rise time for both controllers is still 0.019 s, $K_i = 1$.

### Table 3

| $K_p$ | $K_i$ | Overshoot | Rise Time | Settling Time |
|------|------|------------|-----------|--------------|
| 5    | 1    | 0.697      | 0.03      | 0.184        |
| 5    | 5    | 0.782      | 0.03      | 1.212        |
| 5    | 10   | 0.905      | 0.029     | 0.916        |
| 5    | 15   | 0.964      | 0.029     | 0.721        |
| 5    | 20   | 1.043      | 0.029     | 0.618        |
| 10   | 1    | 0.682      | 0.029     | 0.196        |
| 10   | 5    | 0.71       | 0.029     | 0.255        |
| 10   | 10   | 0.776      | 0.029     | 0.98         |
| 10   | 15   | 0.826      | 0.029     | 0.943        |
| 10   | 20   | 0.864      | 0.029     | 0.864        |
| 15   | 1    | 0.694      | 0.029     | 0.214        |
| 15   | 5    | 0.704      | 0.029     | 0.223        |
| 15   | 10   | 0.732      | 0.029     | 0.631        |
| 15   | 15   | 0.769      | 0.029     | 0.935        |
| 15   | 20   | 0.798      | 0.029     | 0.912        |
| 20   | 1    | 0.691      | 0.029     | 0.219        |
| 20   | 5    | 0.697      | 0.029     | 0.24         |
| 20   | 10   | 0.719      | 0.029     | 0.291        |
| 20   | 15   | 0.738      | 0.029     | 0.701        |
| 20   | 20   | 0.769      | 0.029     | 0.866        |

Fig. 8. Performance graph when $K_p = 1$, $K_i = 1$
Fig. 9. Performance graph when $K_p = 1$, $K_i = 20$

Fig. 10. Performance graph when $K_p = 20$, $K_i = 1$
As shown in Fig. 6, when both value is 1, SIPIC controller has lower overshoot and settling time as compared to PI controller, but higher rise time. When the $K_i$ value is increased, both controller experience an increase in overshoot but the increase in PI controller are far more significant. Rise time for PI controller decreases as the value of $Ki$ increases but SIPIC controller experience higher rise time at higher $K_i$ value. For settling time, both controller first experience a decrease before increasing again as $Ki$ value is further increased. When $K_p$ is 1 and $K_i$ is 15, SIPIC controller performs slightly better by having lower overshoot and settling time but higher rise time as compared to PI controller.

Moving on, $K_p$ value was increased from 1 to 5. Compared to when $K_p$ value is 1, both controller now have higher overshoot, lower rise time and lower settling time. As the value of $K_i$ increases, both controller experience an increase in overshoot. The changes in rise time are insignificant in both controller. For settling time, SIPIC controller goes through a steady decrease in settling time but an increase then decrease in the PI controller. As the $K_i$ value increases, the settling time of the SIPIC controller starts to be significantly lower than the PI controller.

When the $K_p$ is further increased to 10 and 15, the rise time of both controllers remain the same as it is the lowest rise time possible. Both controller also experience minor changes in overshoot with SIPIC controller still having lower overshoot than PI controller. For settling time, SIPIC controller continue to show higher settling time than PI controller when the $K_i$ value is small but SIPIC controller then outperform PI controller when the $K_i$ value increases.
Table 4. Results for loading condition first input

| \(K_p\) | \(K_i\) | Overshoot | \(K_{PI}\) | SIPIC | Rise Time | \(K_{PI}\) | SIPIC | Setting Time | \(K_{PI}\) | SIPIC |
|---|---|---|---|---|---|---|---|---|---|---|
| 1 | 1 | 0.543 | 0.521 | 0.096 | 0.1 | 0.532 | 1.18 |
| 1 | 5 | 1.121 | 0.697 | 0.083 | 0.097 | 1.314 | 1.423 |
| 1 | 10 | 1.624 | 0.876 | 0.075 | 0.095 | N/A | 1.804 |
| 1 | 15 | N/A | 0.923 | 0.068 | 0.09 | N/A | 2.012 |
| 1 | 20 | N/A | 0.942 | 0.063 | 0.09 | N/A | 1.986 |
| 2 | 1 | 0.81 | 0.697 | 0.063 | 0.064 | 1.595 | 0.667 |
| 2 | 5 | 0.917 | 0.766 | 0.058 | 0.06 | 0.967 | 0.631 |
| 2 | 10 | 1.165 | 0.854 | 0.054 | 0.061 | 1.157 | 0.957 |
| 2 | 15 | 1.354 | 0.905 | 0.051 | 0.061 | N/A | 1.038 |
| 2 | 20 | 1.476 | 0.945 | 0.049 | 0.06 | N/A | 1.449 |
| 3 | 1 | 0.813 | 0.807 | 0.046 | 0.047 | 0.586 | 0.579 |
| 3 | 5 | 0.958 | 0.848 | 0.044 | 0.047 | 0.736 | 0.609 |
| 3 | 10 | 1.043 | 0.927 | 0.043 | 0.046 | 0.874 | 0.831 |
| 3 | 15 | 1.212 | 0.952 | 0.043 | 0.046 | 1.413 | 0.963 |
| 3 | 20 | 1.26 | 0.952 | 0.041 | 0.046 | 1.722 | 1.034 |
| 4 | 1 | 0.917 | 0.886 | 0.039 | 0.04 | 0.626 | 0.615 |
| 4 | 5 | 0.967 | 0.92 | 0.039 | 0.04 | 0.709 | 0.65 |
| 4 | 10 | 1.052 | 0.955 | 0.037 | 0.039 | 0.824 | 0.642 |
| 4 | 15 | 1.134 | 0.977 | 0.038 | 0.039 | 0.951 | 0.643 |
| 4 | 20 | 1.256 | 1.005 | 0.036 | 0.039 | 1.044 | 0.912 |
| 5 | 1 | 0.895 | 0.939 | 0.037 | 0.037 | 0.593 | 0.762 |
| 5 | 5 | 0.961 | 0.971 | 0.039 | 0.037 | 0.718 | 0.65 |
| 5 | 10 | 1.037 | 1.037 | 0.036 | 0.037 | 0.73 | 0.658 |
| 5 | 15 | 1.106 | 1.065 | 0.036 | 0.037 | 0.815 | 0.667 |
| 5 | 20 | 1.168 | 1.118 | 0.036 | 0.037 | 0.929 | 0.672 |
| 10 | 1 | 0.889 | 0.92 | 0.035 | 0.035 | 0.652 | 0.664 |
| 10 | 5 | 0.933 | 1.074 | 0.036 | 0.035 | 0.717 | 0.644 |
| 10 | 10 | 0.974 | 1.209 | 0.035 | 0.035 | 0.791 | 0.68 |
| 10 | 15 | 1.002 | 1.344 | 0.035 | 0.035 | 0.802 | 0.683 |
| 10 | 20 | 1.043 | 1.426 | 0.035 | 0.035 | 0.799 | 0.858 |
| 15 | 1 | 0.908 | 0.917 | 0.035 | 0.035 | 0.724 | 0.711 |
| 15 | 5 | 0.92 | 1.046 | 0.035 | 0.035 | 0.772 | 0.73 |
| 15 | 10 | 0.942 | 1.172 | 0.035 | 0.035 | 0.791 | 0.764 |
| 15 | 15 | 0.958 | 1.25 | 0.035 | 0.035 | 0.792 | 0.789 |
3.2 With Load

3.2.1 0 to 90

| Value | Kp | Ki | PI | SIPIC | PI | SIPIC | PI | SIPIC |
|-------|----|----|----|-------|----|-------|----|-------|
| 10    | 1  | 0.543 | 0.521 | 0.096 | 0.1 | 0.532 | 1.18 |
| 10    | 5  | 1.121 | 0.697 | 0.083 | 0.097 | 1.314 | 1.423 |
| 10    | 10 | 1.624 | 0.876 | 0.075 | 0.095 | N/A   | 1.804 |
| 10    | 15 | N/A   | 0.923 | 0.068 | 0.09 | N/A   | 2.012 |
| 10    | 20 | N/A   | 0.942 | 0.063 | 0.09 | N/A   | 1.986 |
| 20    | 1  | 0.81  | 0.697 | 0.063 | 0.064 | 1.595 | 0.667 |
| 20    | 5  | 0.917 | 0.766 | 0.058 | 0.06 | 0.967 | 0.631 |
| 20    | 10 | 1.165 | 0.854 | 0.054 | 0.061 | 1.157 | 0.957 |
| 20    | 15 | 1.354 | 0.905 | 0.051 | 0.061 | N/A   | 1.038 |
| 20    | 20 | 1.476 | 0.945 | 0.049 | 0.06 | N/A   | 1.449 |
| 30    | 1  | 0.813 | 0.807 | 0.046 | 0.047 | 0.586 | 0.579 |
| 30    | 5  | 0.958 | 0.848 | 0.044 | 0.047 | 0.736 | 0.609 |
| 30    | 10 | 1.043 | 0.927 | 0.043 | 0.046 | 0.874 | 0.831 |
| 30    | 15 | 1.212 | 0.952 | 0.043 | 0.046 | N/A   | 1.038 |
| 30    | 20 | 1.26  | 0.952 | 0.041 | 0.046 | N/A   | 1.119 |
| 40    | 1  | 0.917 | 0.886 | 0.039 | 0.04 | 0.626 | 0.615 |
| 40    | 5  | 0.967 | 0.92 | 0.039 | 0.04 | 0.709 | 0.65 |
| 40    | 10 | 1.052 | 0.955 | 0.037 | 0.03 | 0.824 | 0.642 |
| 40    | 15 | 1.134 | 0.977 | 0.038 | 0.039 | 0.951 | 0.643 |
| 40    | 20 | 1.256 | 1.005 | 0.036 | 0.039 | 1.044 | 0.912 |
| 50    | 1  | 0.895 | 0.939 | 0.037 | 0.037 | 0.593 | 0.762 |
| 50    | 5  | 0.961 | 0.971 | 0.039 | 0.037 | 0.718 | 0.65 |
| 50    | 10 | 1.037 | 1.037 | 0.036 | 0.037 | 0.73 | 0.658 |
| 50    | 15 | 1.106 | 1.065 | 0.036 | 0.037 | 0.815 | 0.667 |
| 50    | 20 | 1.168 | 1.118 | 0.036 | 0.037 | 0.929 | 0.672 |
| 100   | 1  | 0.889 | 0.92 | 0.035 | 0.035 | 0.652 | 0.664 |
| 100   | 5  | 0.933 | 1.074 | 0.036 | 0.035 | 0.717 | 0.644 |
| 100   | 10 | 0.974 | 1.209 | 0.035 | 0.035 | 0.791 | 0.68 |
| 100   | 15 | 1.002 | 1.344 | 0.035 | 0.035 | 0.802 | 0.683 |
| 100   | 20 | 1.043 | 1.426 | 0.035 | 0.035 | 0.799 | 0.8 |
| 150   | 1  | 0.908 | 0.917 | 0.035 | 0.035 | 0.724 | 0.711 |
| 150   | 5  | 0.92 | 1.046 | 0.035 | 0.035 | 0.772 | 0.73 |
| 150   | 10 | 0.942 | 1.172 | 0.035 | 0.035 | 0.791 | 0.764 |
| 150   | 15 | 0.958 | 1.25 | 0.035 | 0.035 | 0.792 | 0.789 |
| 150   | 20 | 0.989 | 1.319 | 0.035 | 0.035 | 0.898 | 0.964 |
| 200   | 1  | 0.876 | 0.895 | 0.035 | 0.035 | 0.785 | 0.834 |
| 200   | 5  | 0.908 | 0.999 | 0.035 | 0.035 | 0.869 | 0.833 |
| 200   | 10 | 0.911 | 1.09 | 0.035 | 0.035 | 0.875 | 0.834 |
| 200   | 15 | 0.952 | 1.178 | 0.035 | 0.035 | 0.883 | 0.876 |
| 200   | 20 | 0.964 | 1.209 | 0.035 | 0.035 | 0.958 | 1.119 |

**Fig. 12.** Performance graph when $K_p = 1$, $K_i = 1$
Fig. 13. Performance graph when $K_p = 1$, $K_i = 20$

Fig. 14. Performance graph when $K_p = 20$, $K_i = 1$
Table 4 shows results of controller first input under loading condition. When both \( K_p \) and \( K_i \) is 1 as shown in Fig. 10, SIPIC controller showed lower overshoot, shorter rise time and settling time as compared to PI controller. As the \( K_i \) value increases, both controller experience an increase in overshoot and decrease in rise time. Both controller also experience an increase in the settling time of the controller but for PI, the settling time were unable to be determined because the controller have not settled in the given step time. When \( K_p \) is 1 and \( K_i \) is 15 as shown in Fig. 11, PI controller shows very high oscillation and the overshoot, rise time and settling time were all not able to be determined. SIPIC controller on the other hand, still showed a very stable performance. This proves that SIPIC controller is more stable than conventional PI controller.

As the \( K_p \) value is increases from 1 to 5, both controller experience higher overshoot but lower rise time. PI controller experienced slightly longer slightly time but SIPIC controller experienced shorter settling time. As the value of \( K_i \) increases, the overshoot for both controller increases and both controller showed similar overshoot. The difference in rise time of both controllers is insignificant while SIPIC controller shows shorter settling time as \( K_i \) increases.

The \( K_p \) value is increased to 10 with \( K_i \) value being 1, 5, 10 and 15 respectively. Compared to when \( K_p \) was 5, both controller experience slight decrease in overshoot and rise time. PI controller has a longer settling time than before while SIPIC controller now has shorter settling time. As \( K_i \) increases, overshoot for both controllers increases while PI controller outperform SIPIC controller by having lower overshoot. Both controller shows similar rise time as this is the lowest possible value. Both controllers also experience longer settling time as \( K_i \) increases but SIPIC performs slightly better by having slightly shorter settling time.
Lastly, $K_p$ value was increased to 15. Rise time remains the same for both controllers. The increase in $K_p$ value causes minor changes to overshoot in both controller while increasing the settling time for both controller. As $K_i$ increases, similar pattern as before can be seen. Both overshoot and settling time increases alongside the $K_i$ value and PI controller still have lower overshoot while settling time are similar for both controller.

3.2.2 270 to 90

### Table 5. Results for loading condition second input

| $K_p$ | $K_i$ | Overshoot  | Rise Time  | Settling Time |
|-------|-------|------------|------------|---------------|
|       |       | PI         | SIPIC      | PI            | SIPIC         |
| 1     | 1     | 1.209      | 1.033      | 0.09          | 0.098         |
| 1     | 5     | 2.18       | 1.429      | 0.101         | 0.092         |
| 1     | 10    | 3.251      | 1.586      | 0.052         | 0.094         |
| 1     | 15    | N/A        | 1.778      | 0.052         | 0.091         |
| 1     | 20    | N/A        | 1.765      | 0.052         | 0.091         |
| 2     | 1     | 1.47       | 1.316      | 0.059         | 0.06          |
| 2     | 5     | 1.765      | 1.586      | 0.057         | 0.058         |
| 2     | 10    | 2.224      | 1.674      | 0.054         | 0.059         |
| 2     | 15    | 2.673      | 1.759      | 0.054         | 0.059         |
| 2     | 20    | 3.349      | 1.85       | 0.052         | 0.058         |
| 3     | 1     | 1.467      | 1.495      | 0.053         | 0.053         |
| 3     | 5     | 1.75       | 1.743      | 0.053         | 0.053         |
| 3     | 10    | 1.988      | 1.982      | 0.053         | 0.053         |
| 3     | 15    | 2.321      | 2.139      | 0.052         | 0.052         |
| 3     | 20    | 2.557      | 2.287      | 0.052         | 0.052         |
| 4     | 1     | 1.436      | 1.514      | 0.052         | 0.052         |
| 4     | 5     | 1.674      | 1.825      | 0.052         | 0.053         |
| 4     | 10    | 1.875      | 2.152      | 0.052         | 0.053         |
| 4     | 15    | 2.067      | 2.447      | 0.052         | 0.052         |
| 4     | 20    | 2.347      | 2.598      | 0.051         | 0.052         |
| 5     | 1     | 1.414      | 1.527      | 0.052         | 0.052         |
| 5     | 5     | 1.599      | 1.9        | 0.051         | 0.052         |
| 5     | 10    | 1.803      | 2.281      | 0.052         | 0.051         |
| 5     | 15    | 1.985      | 2.516      | 0.052         | 0.052         |
| 5     | 20    | 2.18       | 2.651      | 0.052         | 0.051         |
| 10    | 1     | 1.388      | 1.495      | 0.052         | 0.051         |
| 10    | 5     | 1.508      | 1.74       | 0.052         | 0.051         |
| 10    | 10    | 1.624      | 1.951      | 0.052         | 0.051         |
| 10    | 15    | 1.712      | 2.108      | 0.052         | 0.051         |
| 10    | 20    | 1.806      | 2.19       | 0.052         | 0.051         |
| 15    | 1     | 1.42       | 1.442      | 0.052         | 0.051         |
Lastly, Kp value was increased to 15. Rise time remains the same for both controllers. The increase in Kp value causes minor changes to overshoot in both controller while increasing the settling time for both controller. As Ki increases, similar pattern as before can be seen. Both overshoot and settling time increases alongside the Ki value and PI controller still have lower overshoot while settling time are similar for both controller.

3.2.2 270 to 90

| Kp | 5   | 1.47 | 1.65 | 0.052 | 0.051 | 0.95 | 0.85 |
|----|-----|------|------|-------|-------|------|------|
| 10 | 1.52 | 1.8 | 0.052 | 0.051 | 0.97 | 0.89 |
| 15 | 1.58 | 1.91 | 0.052 | 0.051 | 1.11 | 0.99 |
| 20 | 1.63 | 1.97 | 0.052 | 0.051 | 1.09 | 1.19 |

| Ki | 1   | 1.38 | 1.52 | 0.052 | 0.051 | 0.83 | 0.84 |
|----|-----|------|------|-------|-------|------|------|
| 5  | 1.47 | 1.65 | 0.052 | 0.051 | 0.94 | 0.95 |
| 10 | 1.52 | 1.80 | 0.052 | 0.051 | 0.97 | 0.98 |
| 15 | 1.58 | 1.92 | 0.052 | 0.051 | 1.11 | 1.01 |

| PI | 1   | 1.47 | 1.65 | 0.052 | 0.051 | 0.83 | 0.84 |
|----|-----|------|------|-------|-------|------|------|
| 5  | 1.47 | 1.65 | 0.052 | 0.051 | 0.94 | 0.95 |
| 10 | 1.52 | 1.80 | 0.052 | 0.051 | 0.97 | 0.98 |
| 15 | 1.58 | 1.92 | 0.052 | 0.051 | 1.11 | 1.01 |

**Fig. 16.** Performance graph when Kp = 1, Ki = 1
Fig. 17. Performance graph when $K_p = 1$, $K_i = 20$

Fig. 18. Performance graph when $K_p = 20$, $K_i = 1$
Table 5 show the results obtained for second input under loading condition. When both $K_p$ and $K_i$ is 1 as shown in Fig. 13, PI controller shows higher overshoot, longer settling time but slightly shorter rise time as compared to SIPIC controller. As $K_i$ value increases, a major increase in overshoot can be seen in the PI controller and while SIPIC controller also experience an increase in overshoot, the increase is subtle as compared to PI controller. A very minimal spike was seen in the PI controller when $K_i$ value was increased from 1 to 5 but the rise time then decreases when the value was further increased to 10. For settling time, a steady increase can be seen in the SIPIC controller as the $K_i$ value increases. However, for the PI controller, it was unable to settle inside the desired range within the step time given. Fig. 14 shows the controller performance when $K_p$ is 1 and $K_i$ is 15, the overshoot and settling time experienced by the PI controller is far higher than the SIPIC controller.

Moving on to $K_p = 5$, the increase in $K_p$ value causes the overshoot of both controller to increase and an increase in rise time can be seen in both controller. A more stable performance is seen in the PI controller as the controller can settle in the given step time while the SIPIC controller has a slightly longer settling time due to the increase in $K_p$ value. As the $K_i$ value increases, overshoot starts to increase in both controller and PI controller has a lower overshoot as compared to SIPIC controller. Rise time of both controllers are similar while PI controller shows slightly shorter settling time as compared to SIPIC controller.

When the $K_p$ value is increased to 10, both controller experienced a decrease in overshoot but slight decrease in the settling time. Rise time remains the same as it is the lowest possible rise time due to limitation of hardware. The pattern for overshoot is similar as before where overshoot increase alongside the $K_i$ value with PI controller having lower overshoot than SIPIC controller. Similar trend can also be found in settling time of both controller. As $K_i$ increases, both controller starts to have longer settling time and SIPIC controller perform slightly better in this aspect.
As the $K_p$ value further increases to 15, PI controller continue to see an increase in overshoot while SIPIC controller experience decrease in overshoot. Rise time remains the same for both controller and settling time increased for both controller. The same trend is seen for both overshoot and settling time of both controllers as $K_i$ value increases. When both $K_p$ and $K_i$ is 15 as seen in Fig. 15, PI controller performs better in terms of overshoot while SIPIC controller shows more ideal performance in terms of settling time.

4. Conclusion

To conclude this paper, a new type of anti-windup PI controller was used for the application of motor position control and the performance of the proposed controller was used to compare with the performance of a conventional PI controller. The controllers were tested under no load and under loading condition while two different inputs were used to test the controllers: 0˚ to 90˚ and 270˚ to 90˚. Results show that under no load condition, SIPIC controller performs better than PI controller by having lower overshoot and lower settling time across various gains. Under loading condition, PI controller showed to be unstable under some of the gains as it experience large amount of oscillations and overshoot as well as very long settling time whereas under the same gain, the proposed SIPIC controller still shows acceptable performance. As a conclusion, the proposed SIPIC controller shows a more reliable performance as compared to PI controller when experiment was conducted. A wider range of gain can be used for the SIPIC controller whereas PI controller only performs well under a small range of gain.

As this paper only compares the performance between the proposed SIPIC controller and conventional PI controller, the future work includes further validating the performance of SIPIC controller in motor position control by comparing it with other anti-windup PI controller. SIPIC controller can also be further refined to achieve lower overshoot, shorter rise time and settling time. Wider range of gain can also be used to test the performance of SIPIC controller to further test the stability of the controller.

References

1. A. Benmakhlouf, A. Louchene, and D. Djarah, “Fuzzy logic and modified crisp logic applied to a DC motor position control,” Control Intell. Syst., 38, no. 3, 173–180 (2010).
2. N. Bianchi and M. Dai Pre, “Active power filter control using neural network technologies,” IEE Proceedings-Electric Power Appl., 150, no. 2, 139–145, (2003).
3. V. Slapak, K. Kyslan, and F. Durovsky, “Position Controller for PMSM Based on Finite Control Set Model Predictive Control,” no. 1, 17–22 (2016).
4. K. Orman, A. Basci, A. Derdiyok, E. Engineering, and E. Engineering, “Speed and Direction Angle Control of Four Wheel Drive Skid-Steered Mobile Robot by Using Fractional Order PI Controller,” 14–20 (2016).
5. S.-Y. Chen and C.-S. Chia, “Precision Position Control of a Voice Coil Motor Using Self-Tuning Fractional Order Proportional-Integral-Derivative Control,” Micromachines, 7, no. 11, 207 (2016).
6. K. Kawamura and Y. Ishida, “Approach to New Model Recovery Anti-windup Scheme with PID Controller,” 2015 3rd Int. Conf. Artif. Intell. Model. Simul., 208–212 (2015).
7. C. T. Ling, “A New Anti-windup PI control for Motor Speed Application,” no. June (2016).
8. K. Sakai and Y. Ishida, “A Design of an Improved Anti-Windup Control Using a PI
As the Kp value further increases to 15, PI controller continue to see an increase in overshoot while SIPIC controller experience decrease in overshoot. Rise time remains the same for both controller and settling time increased for both controller. The same trend is seen for both overshoot and settling time of both controllers as K_i value increases. When both Kp and K_i is 15 as seen in Fig. 15, PI controller performs better in terms of overshoot while SIPIC controller shows more ideal performance in terms of settling time.

4. Conclusion

To conclude this paper, a new type of anti-windup PI controller was used for the application of motor position control and the performance of the proposed controller was used to compare with the performance of a conventional PI controller. The controllers were tested under no load and under loading condition while two different inputs were used to test the controllers: 0˚ to 90˚ and 270˚ to 90˚. Results show that under no load condition, SIPIC controller performs better than PI controller by having lower overshoot and lower settling time across various gains. Under loading condition, PI controller showed to be unstable under some of the gains as it experienced large amount of oscillations and overshoot as well as very long settling time whereas under the same gain, the proposed SIPIC controller still shows acceptable performance. As a conclusion, the proposed SIPIC controller shows a more reliable performance as compared to PI controller when experiment was conducted. A wider range of gain can be used for the SIPIC controller whereas PI controller only performs well under a small range of gain.

As this paper only compares the performance between the proposed SIPIC controller and conventional PI controller, the future work includes further validating the performance of SIPIC controller in motor position control by comparing it with other anti-windup PI controller. SIPIC controller can also be further refined to achieve lower overshoot, shorter rise time and settling time. Wider range of gain can also be used to test the performance of SIPIC controller to further test the stability of the controller.

References

1. A. Benmakhlouf, A. Louchene, and D. Djarah, "Fuzzy logic and modified crisp logic applied to a DC motor position control," Control Intell. Syst., 38, no. 3, 173–180 (2010).
2. N. Bianchi and M. Dai Pre, "Active power filter control using neural network technologies," IEE Proceedings-Electric Power Appl., 150, no. 2, 139–145, (2003).
3. V. Slapak, K. Kyslan, and F. Durovsky, "Position Controller for PMSM Based on Finite Control Set Model Predictive Control," no. 1, 17–22 (2016).
4. K. Orman, A. Basci, A. Derdiyok, E. Engineering, and E. Engineering, "Speed and Direction Angle Control of Four Wheel Drive Skid-Steered Mobile Robot by Using Fractional Order PI Controller," 14–20 (2016).
5. S.-Y. Chen and C.-S. Chia, "Precision Position Control of a Voice Coil Motor Using Self-Tuning Fractional Order Proportional-Integral-Derivative Control," Micromachines, 7, no. 11, 207 (2016).
6. K. Kawamura and Y. Ishida, "Approach to New Model Recovery Anti-windup Scheme with PID Controller," 2015 3rd Int. Conf. Artif. Intell. Model. Simul., 208–212 (2015).
7. C. T. Ling, "A New Anti-windup PI control for Motor Speed Application," no. June (2016).
8. K. Sakai and Y. Ishida, "A Design of an Improved Anti-Windup Control Using a PI Controller Based on a Pole Placement Method," no. 2, 1–7 (2016).
9. S. C. Pratama, E. Susanto, and A. S. Wibowo, “Design and implementation of water level control using gain scheduling PID back calculation integrator Anti Windup,” ICCEREC 2016 - Int. Conf. Control. Electron. Renew. Energy, Commun., Conf. Proc., 101–104 (2017).