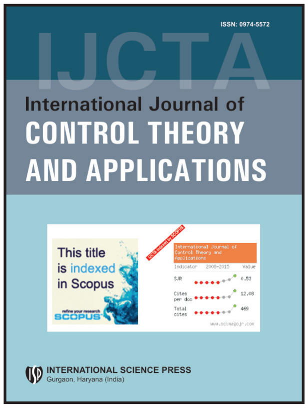

International Journal of Control Theory and Applications

ISSN : 0974-5572

© International Science Press

Volume 10 • Number 16 • 2017

A Novel Gain Tuning of Anti-Windup PID Controller using Ant Lion Optimization

Ayyarao SLV Tummala[a], Ravi Kiran Inapakurthi[b] and P.V. Ramanarao[c]

[a,b]Department of Electrical and Electronics Engineering, GMR Institute of Technology Rajam, Srikakulam, AP, India. Email: ayyarao.tslv@gmrit.org, ravikiran.i@gmrit.org
[c]Department of Electrical and Electronics Engineering, Nagarjuna University, Guntur, AP, India

Abstract: PID controller is superior in terms of fast response, immune to parameter variations, disturbance rejection etc. But when control input to the plant saturates, the response of the system is significantly affected and in some cases this may lead to instability. PID controller with anti-windup is proposed in the literature that can handle saturation in control input. The challenging task in implementation is the tuning the parameters of anti-windup PID. To address this issue, a simplified method to design the parameters of PID with anti-windup is proposed in this paper. The parameters of anti-windup are tuned by Ant Lion optimization technique (ALO). The proposed method is validated for different case studies including basic second order system, speed control of DC motor and induction motor with direct torque control. The proposed design is found to be superior in dynamic response in the cases studied. A Novel Gain Tuning of Anti-Windup PID Controller is proposed using Ant Lion Optimization.

Keywords: Anti-windup PID, PI control, Ant Lion optimization technique, Gain tuning.

1. INTRODUCTION

PI and PID controllers are the most popular linear controllers due to the salient features like zero steady state error, good dynamic response, high disturbance rejection. Because of these features, the applications of PI controller are vast and are not only limited to variable speed drives, process control, wind energy conversion system, solar power plant, converters. PID controller is widely used in several industrial applications. Most of the practical systems possess non-linear characteristics and these systems are linearized around an operating point. Hence the control input to the plant is limited to certain range. In some cases the control input generated may not be realized in practical cases. Considering the case of speed control of chopper fed DC motor, the control voltage cannot exceed input battery voltage. For most of the cases, the control input is limited by plant functioning equipment. Consider another example of inverter fed induction motor, current input to the motor is restricted by device ratings. Therefore a saturation block is connected in simulation studies to limit the control input to the plant. When saturation is set in, non-linearity is enforced in plant dynamics. Thus the functionality of PID controller is deviated from the desired. To address this issue, PID controller with anti-wind up is proposed in the literature.

The parameters of the PID controller must be designed to improve the stability, transient response of the system and to reduce steady state error. The tuned system must also be immune to large disturbances, modeling errors and parameter variations[1]–[4]. There are many techniques proposed in the literature to tune the parameters. The tunable parameters for PID controller are K_P, K_I and K_D. Other than conventional methods like Ziegler–Nichols method, Pole placement technique, Nyquist method, controller can also be tuned by nature inspired optimization algorithms. The most recent, superior and easiest one is tuning the parameters using Ant Lion Optimization.

The transfer function of PI controller is given as

$$T_{PI}(s) = K_P + \frac{K_i}{s} \tag{1.1}$$

The transfer function of PID controller is given as

$$T_{PID}(s) = K_P + \frac{K_I}{s} + K_D s \tag{1.2}$$

The control output of PID controller with anti-windup is

$$v = K_P e + K_I \int \left(e + \frac{1}{T_t} e_t \right) dt + K_D \frac{de}{dt} \tag{1.3}$$

where, $e_t = v - u$

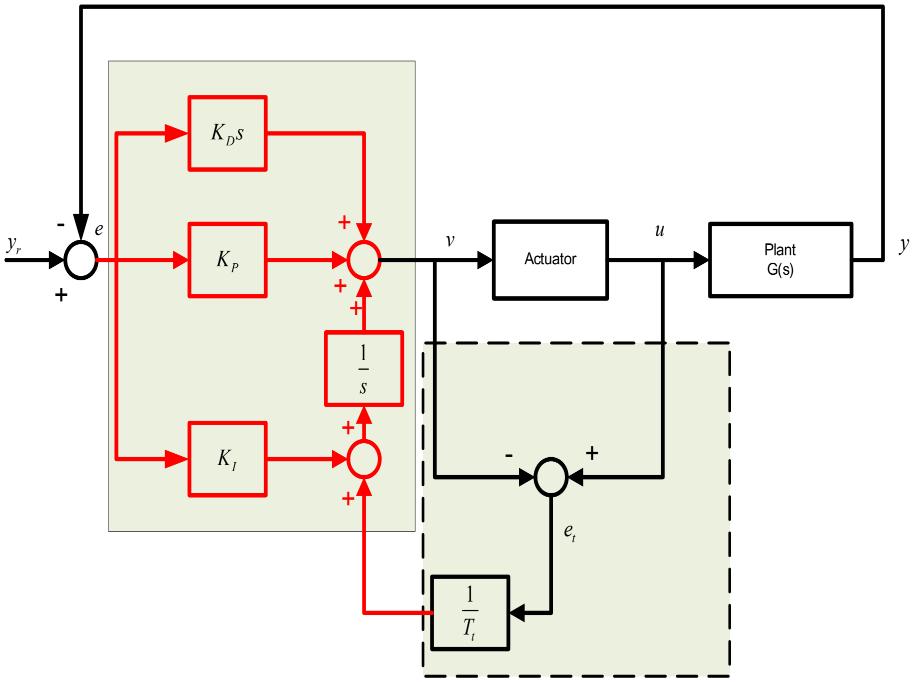

Figure 1: Anti-windup PID controller

If controller output v is equal to plant input u i.e. under non-saturation condition, this controller works like an ordinary PID controller and Tt is called as correction factor[5]–[9].

This paper details the design of anti-windup PID controller by tuning the parameters using ALO algorithm[10]. The ALO tuned anti-windup PID controller is compared with PID controller for second order system. Later anti-windup PI is tuned for speed control of DC motor. The performance of anti-windup PI controller is compared with tuned PID controller. In the latter case, ALO tuned anti-windup PI controller is compared with PID controller for speed control of induction motor with direct torque control.

2. ANT LION OPTIMIZATION ALGORITHM

ALO optimization is a bio-inspired heuristic algorithm inspired from the antlions trapping the ants in the process of hunting them. Antlions make inverted cone shaped cavities on sands with their jaws. These cavities are used for trapping the ants. Antlions will hide into that tips of the cones and waits for the ants to fall in that cavity. If the ants trap in those cavities then the antlion try to catch them. After hunting, antlions will make the channels clean for another hunt.

The ALO algorithm imitates the behavior of ant lions and ants in a trap. In such modeling ants are made to crawl over the search space and ant lions are made to catch them using traps. A random walk is initiated for this model as ants wander around for food randomly. Apart from ants, ant lions are also considered to occupy some space and hiding.

For the application of ALO, several considerations like different random walks chosen by ants in the search space, traps of ant lions and size of the cavity are made. The traps built by the ant lions are based on their fitness and are also affected by their ability to catch. In every iteration each antlion hunts an ant and the range of random walk will be diminished so as to model the sliding of the ants. The relocation of the antlion depends on the fitness of the ant hunt by it.

The random walk of ants is such that they update their positions in each step of optimization matching with the positions of ant lions. For building a trap, ALO algorithm uses a roulette wheel for selection of fittest ant lion during optimization which ensures a higher probability for ant lions in catching the prey. Once the ant falls in the trap, ant lion drags the ant to the epicenter. Once the ant reaches to bottom of the trap, the ant lion consumes its body and then it moves to a new location to enhance its trap to catch the other prey. In this process the best of all ant lions from each iteration is taken as elite.

In ALO algorithm, different solutions (ants and ant lions) are randomly generated. For every iteration, the position of the ant changes according to the ant lion selected by roulette wheel operator. Initially the boundary of position is according to the current iteration and then it will depend on the selected ant lion and elite. Fitness of all the ants will be evaluated and the new location of the ant lion is matched to the highest fit ant lion in the previous iteration. The best ant lion obtained in current iteration is compared with the elite and the fittest of the two is considered as the new elite.

The Antlion algorithm is given below [10]

I. Initialize the maximum iteration, population of antlions and ants separately for a given dimension *d*.

II. Generate the fitness values of ants and antlions from the objective function.

$$\Theta_{\mathrm{OA}} = \begin{bmatrix} \Psi\left(\Lambda_{1,1}, \Lambda_{1,2}, \ldots, \Lambda_{1,d}\right) \\ \Psi\left(\Lambda_{2,1}, \Lambda_{2,2}, \ldots, \Lambda_{2,d}\right) \\ \vdots \\ \vdots \\ \Psi\left(\Lambda_{n,1}, \Lambda_{n,2}, \ldots, \Lambda_{n,d}\right) \end{bmatrix} \tag{2.1}$$

$$\Theta_{\text{OAL}} = \begin{bmatrix} \Psi\left(\Lambda\text{L}_{1,1}, \Lambda\text{L}_{1,2}, ..., \Lambda\text{L}_{1,d}\right) \\ \Psi\left(\Lambda\text{L}_{2,1}, \Lambda\text{L}_{2,2}, ..., \Lambda\text{L}_{2,d}\right) \\ : \\ : \\ \Psi\left(\Lambda\text{L}_{n,1}, \Lambda\text{L}_{n,2}, ..., \Lambda\text{L}_{n,d}\right) \end{bmatrix} \tag{2.2}$$

$\Lambda\text{L}_{n,d}$ represents d^{th} variable of n^{th} antlion.

$\Lambda_{n,d}$ represents d^{th} variable of n^{th} ant.

$\Psi(\Lambda_{n,1}, \Lambda_{n,2}, ..., \Lambda_{n,d})$ represents fitness value of n^{th} ant. $\Psi(\Lambda\text{L}_{n,1}, \Lambda\text{L}_{n,2}, ..., \Lambda\text{L}_{n,d})$ represents fitness value of n^{th} antlion.

III. The antlion with optimum fitness value is treated as elite.

while $i < \text{Max}_{\text{ite}}$

for $j < n$ (maximum # ants)

IV. Choose antlion using Roulette wheel

V. Update min-max of variables as given in (2.3) and (2.4)

$$\theta^t = \frac{\theta^t}{\text{I}} \tag{2.3}$$

$$\pi^t = \frac{\pi^t}{\text{I}} \tag{2.4}$$

VI. Generate a random walk of ants using (2.5)

$$\text{X}(t) = \left[0, \Sigma\left(2r(t_1) - 1\right), \Sigma\left(2r(t_2) - 1\right), ..., \Sigma\left(2r(t_n) - 1\right)\right] \tag{2.5}$$

where, r is a random value.

VII. Update antlion position using (2.6)

$$\text{Ant}_i^t = \frac{\text{R}_\text{A}^t + \text{R}_\text{E}^t}{2} \tag{2.6}$$

end for loop

VIII. Generate the fitness values of ants

IX. Replace ant with antlion if $f\left(\text{Ant}_i^t\right) > f\left(\text{Antlion}_j^t\right)$

X. Modify elite if fitness value is optimum.

end while loop

Display elite

3. CASE STUDIES AND RESULT ANALYSIS

Case study 1: A Second order system:

Let us consider second order linear system

$$G_1(s) = \frac{\omega_0^2}{s^2 + 2\xi\omega_0 s + \omega_0^2} \qquad \xi = 0.8 \text{ and } \omega_0 = 1 \tag{3.1}$$

Here the objective is to tune the parameters K_P, K_I and K_D for PID controller.

Constraints: $0 \le K_P \le 10 \quad 0 \le K_I \le 10 \quad 0 \le K_D \le 10$

To tune the parameters of PID controller, an objective function is formulated in such a way that steady state error, settling time, rise time and peak overshoot is reduced.

$$F = \min\left(\int \left| y_r - y \right| t\, dt + y_p - y_r\right) \tag{3.2}$$

where, y_r is a reference output,

y is the measured output,

y_p is the peak value of the output.

PI and PID control parameters are tuned using ALO algorithm. The parameters obtained for PID controller after tuning are given below

$$K_P = 9.90,\ K_I = 2.49 \text{ and } K_D = 3.66$$

The transient response of the plant is shown in Figure (2).When compared to PI controller, PID controller is superior in terms of peak overshoot, settling time, Figure (3) shows the control input to the plant without saturation (actuator). The control effort is very high which may not be possible in real-time conditions. Hence there is a need to limit the control input to the plant. To address this issue a saturation block is added to simulated system to limit the control input to ±5.

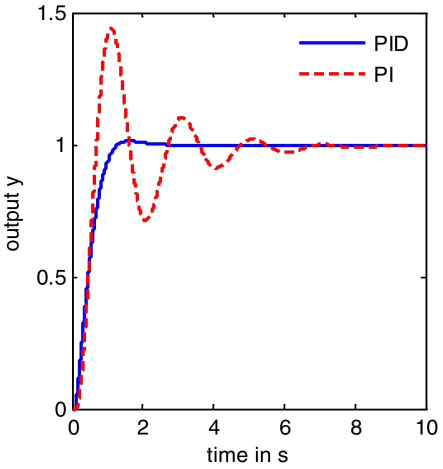

Figure 2: Output of second order system

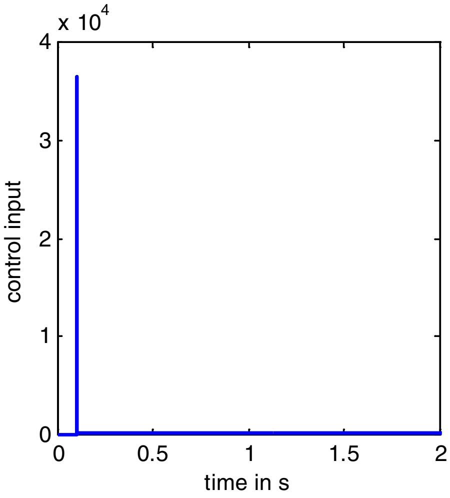

Figure 3: Control input

The response shows that peak overshoot, rise time and settling time are increased under saturated conditions. This is because when the control input to the plant cannot exceed saturation limit and this introduces nonlinearity in to the system. In some cases this may lead to instability. To address this issue, PID controller is replaced with anti-windup PID controller. There are many designs of PID controller with anti-windup. Out of these the conventional method, Tracking Back calculation scheme is adopted here. The transfer function of anti-windup PID controller is given in section-I. The design of anti-windup PID controller includes selection of K_P, K_I, K_D and T_t.

The limits of PID controller are selected same as before. The range of correction factor is selected as [0.001 1]. Thus the number of parameters to be tuned is four. Following the AntLion optimization algorithm given in section 2, the objective function (3.2) is minimized and the corresponding tuned parameters are given below.

$$K_P = 9.90,\ K_I = 2.49,\ K_D = 3.66 \text{ and } T_t = 0.1432$$

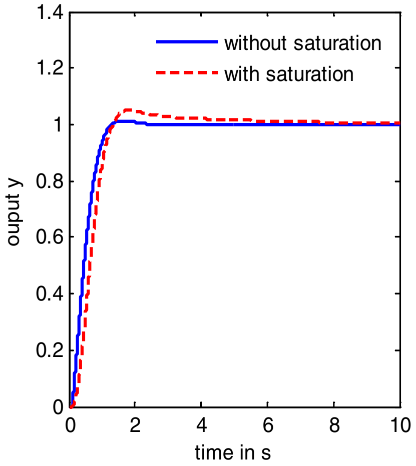

Figure 4: Output of the system

Figure (5) shows the performance of the system with anti-windup controller. The output settles faster with less peak overshoot for PID with anti-windup.

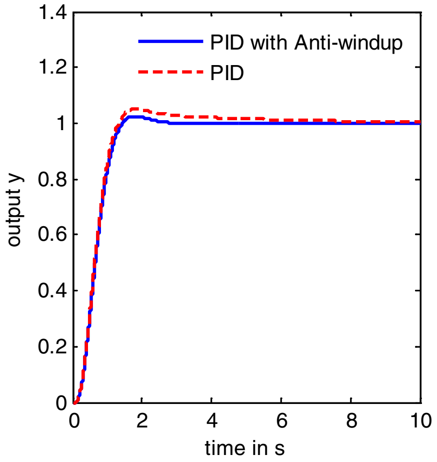

Figure 5: Output with anti-windup controller

Case Study-2: Speed Control of DC motor:

Here the objective is to tune the parameters K_P, K_I and K_D for PID controller.

Constraints: $0 \le K_P \le 2 \quad 0 \le K_I \le 20 \quad 0 \le K_D \le 1$

To tune the parameters of PID controller, an objective function is formulated in Eq.(3.3) to mitigate steady state error, settling time, rise time and peak overshoot.

$$F = F = \min\left(\int \left| \omega^* - \omega \right| t\, dt + \frac{\omega_{max} - \omega^*}{\omega^*} \right) \tag{3.3}$$

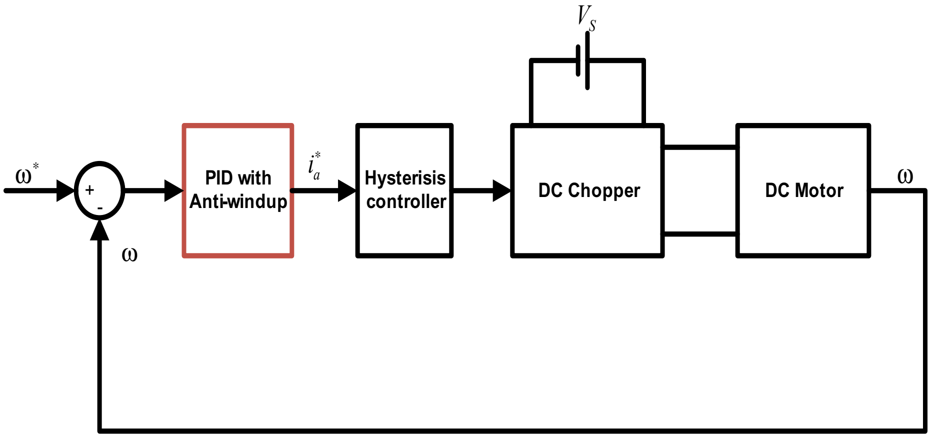

Figure 6: Speed control of DC motor

The open loop transfer function of DC motor is:

$$T(s) = \frac{K_a}{(Js + B)(L_a s + R_a) + K_a^2} \tag{3.4}$$

In this section, we want to analyze the performance of DC motor with anti-windup PID controller. The objective of this section is to design a speed controller for DC motor. The DC motor operates under closed loop conditions operating with variable loads and its speed is regulated using duty ratio of step-down chopper which is derived from the controller. Actual speed ω is sensed and is compared with its reference value ω^*. This speed error is given to a speed regulator which generates reference armature current i_a^* and this current reference is given to hysteresis current controller with hysteresis band of ±2 A. Initially PID controller is selected as speed controller. With PID controller, peak overshoot is 20.83%. This is because at the time of starting the starting current is limited to 30A. To enhance the performance of speed controller, PID controller is replaced with PI controller with anti-wind up. The major challenge in the design of anti-windup is selection of T_t. Figure (7) shows the response of the system with different values of T_t.

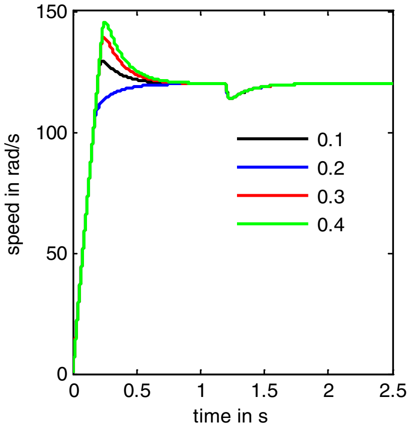

Figure 7: Variation of speed with correction factor

Thus there is a need to select a proper value of T_t. The value of correction factor is tuned using ALO algorithm. Figure (8) shows the performance of the controller with anti-windup. The speed overshoot is drastically reduced at the time of starting. Under running conditions since the control current is less than 30A, the controller is similar to PI controller. A load torque of 30Nm is added at 1.2s and because of this speed is reduced and again

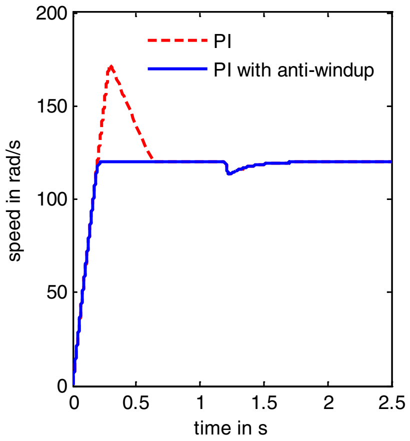

Figure 8: Speed output with anti-windup controller

regained to reference speed within 0.5 s. But there is no effect of correction factor on this because the control current required under load change conditions is less than 30A.

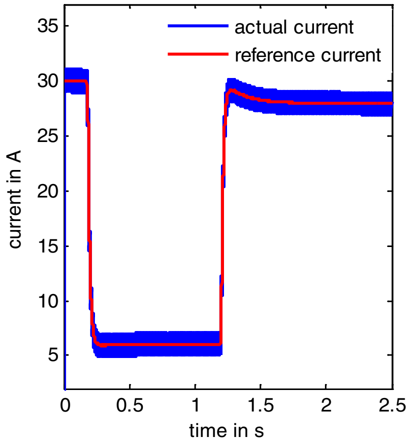

Figure 9: Armature current

Case study-3: Speed Regulation of Induction motor:

The dynamics of induction motor are given by[11].The notations are same as given in literature.

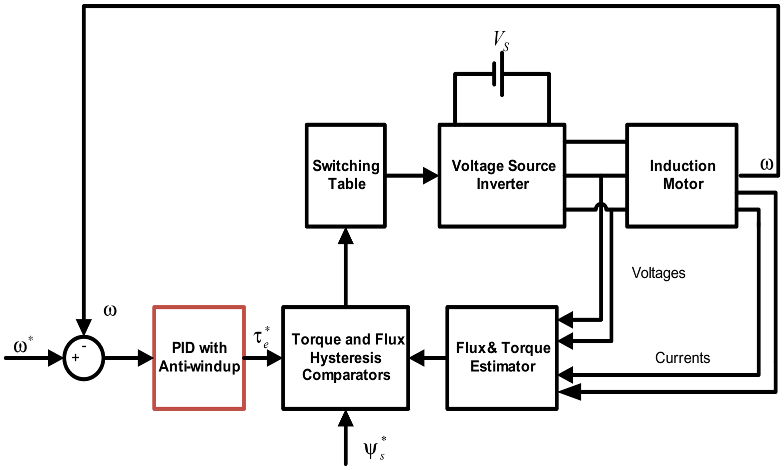

Figure 10: Speed control of induction motor

$$\frac{d\omega}{dt} = \frac{n_p \mathrm{M}}{\mathrm{J}\mathrm{L}_r}\left(\psi_{ra} i_{sb} - \psi_{ra} i_{sa}\right) - \frac{\mathrm{T_L}}{\mathrm{J}} \tag{3.5}$$

$$\frac{d\psi_{ra}}{dt} = -\frac{\mathrm{R}_r}{\mathrm{L}_r}\psi_{ra} + \frac{\mathrm{R}_r}{\mathrm{L}_r}\mathrm{M} i_{sa} - n_p \omega \psi_{rb} \tag{3.6}$$

$$\frac{d\psi_{rb}}{dt} = -\frac{\mathrm{R}_r}{\mathrm{L}_r}\psi_{rb} + \frac{\mathrm{R}_r}{\mathrm{L}_r}\mathrm{M} i_{sb} + n_p \omega \psi_{ra} \tag{3.7}$$

$$\frac{di_{sa}}{dt} = \frac{\mathrm{M}\mathrm{R}_r}{\sigma \mathrm{L}_s \mathrm{L}_r^2}\psi_{ra} + \frac{n_p \mathrm{M}}{\sigma \mathrm{L}_s \mathrm{L}_r} w \psi_{rb} - \left(\frac{\mathrm{M}^2 \mathrm{R}_r + \mathrm{L}_r^2 \mathrm{R}_s}{\sigma \mathrm{L}_s \mathrm{L}_r^2}\right) i_{sa} + \frac{1}{\sigma \mathrm{L}_s} u_{sa} \tag{3.8}$$

$$\frac{di_{sb}}{dt} = \frac{\mathrm{M}\mathrm{R}_r}{\sigma \mathrm{L}_s \mathrm{L}_r^2}\psi_{rb} - \frac{n_p \mathrm{M}}{\sigma \mathrm{L}_s \mathrm{L}_r} w \psi_{ra} - \left(\frac{\mathrm{M}^2 \mathrm{R}_r + \mathrm{L}_r^2 \mathrm{R}_s}{\sigma \mathrm{L}_s \mathrm{L}_r^2}\right) i_{sb} + \frac{1}{\sigma \mathrm{L}_s} u_{sb} \tag{3.9}$$

The measured speed ω is compared with the reference value ω^* and speed error is fed to speed regulator. The output of the speed regulator is a reference torque τ_e^*. Flux and Torque estimator estimates flux and torque from stator voltages and currents. These reference torque and reference fluxes are given to Torque and hysteresis comparators. Figure (11) shows the speed response with PID controller and PI controller with anti-windup. Results show that the speed change under sudden change in load torque conditions is low with anti-windup controller.

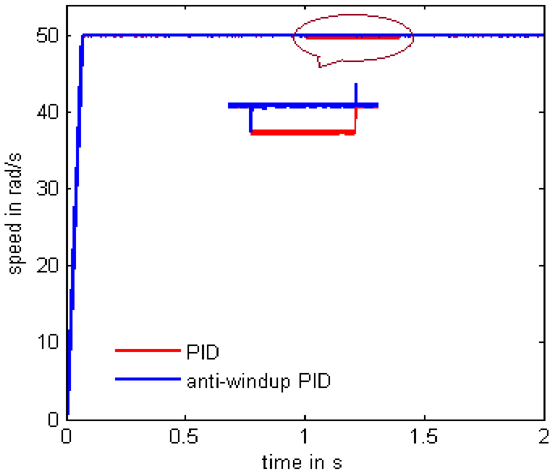

Figure 11: Speed response of Induction motor

4. CONCLUSIONS

The functionality of PID controller is deviated from the desired when control input to the plant saturates. This leads to introduction of controller with anti-windup. In traditional methods, correction factor is selected as a random value which effects the performance of the controller. As it is difficult to tune the anti-windup controller parameters, we proposed the design of anti-windup PID controller parameters using Ant Lion optimization technique. The performance of ALO tuned anti-windup controller is evaluated on different systems. In case of second order system, ALO tuned anti-windup PID controller outperforms anti-windup PID controller. In the second case ALO tuned anti-windup PI controller outperforms PID controller. In the third case, ALO tuned anti-windup PID controller showed superior performance compared to PID controller.

REFERENCES

[1] K. H. Ang, G. Chong, and Y. Li, “PID control system analysis, design, and technology,” IEEE Trans. Control Syst. Technol., Vol. 13, No. 4, pp. 559–576, 2005.

[2] Y. Li, K. H. Ang, and G. C. Y. Chong, “PID control system analysis and design,” IEEE Control Syst., Vol. 26, No. 1, pp. 32 – 41, 2006.

[3] K. J. Åström and T. Hägglund, “The future of PID control,” Control Eng. Pract., Vol. 9, No. 11, pp. 1163–1175, Nov. 2001.

[4] G. P. Liu and S. Daley, “Optimal-tuning PID control for industrial systems,” Control Eng. Pract., Vol. 9, No. 11, pp. 1185–1194, 2001.

[5] J. W. Choi and S. C. Lee, “Antiwindup strategy for PI-type speed controller,” IEEE Trans. Ind. Electron., Vol. 56, No. 6, pp. 2039–2046, 2009.

[6] H.-B. Shin and J.-G. Park, “Anti-Windup PID Controller With Integral State Predictor for Variable-Speed Motor Drives,” IEEE Trans. Ind. Electron., Vol. 59, No. 3, pp. 1509–1516, 2012.

[7] F. Cupertino, E. Mininno, D. Naso, B. Turchiano, and L. Salvatore, “On-line genetic design of anti-windup unstructured controllers for electric drives with variable load,” IEEE Trans. Evol. Comput., Vol. 8, No. 4, pp. 347–364, 2004.

[8] J. M. Gomes da Silva Jr. and S. Tarbouriech, “Anti-windup design with guaranteed regions of stability: an {LMI}-based approach,” Vol. 5, No. 1, pp. 4451–4456, 2003.

[9] L. Zaccarian and A. R. Teel, “A common framework for anti-windup, bumpless transfer and reliable designs,” Automatica, Vol. 38, No. 10, pp. 1735–1744, 2002.

[10] S. Mirjalili, “The Ant Lion Optimizer,” Adv. Eng. Softw., Vol. 83, pp. 80–98, 2015.

[11] T. S. L. V Ayyarao, G. I. Kishore, and M. Rambabu, “Adaptive Control of Saturated Induction Motor with Uncertain Load Torque,” Int. J. Recent Trends Eng. Technol., Vol. 3, No. 4, pp. 2–6, 2010.