# A Predictive Hill Climbing Algorithm for Real Valued multi-Variable Optimization Problem like PID Tuning

Karthikeyan Nagarajan

***Abstract*—This paper proposes a novel method of applying Hill Climbing algorithm for optimizing a problem which has more than one dependent variable and a very large search space. Tuning of PID controller for complete Black-Box plant model is an example of one such problem, where the search algorithm is applied to find PID gains which satisfy the desired optimization criteria. The traditional Hill Climbing algorithm cannot be directly applied to tune PID since the PID controller has three parameters to be tuned and search space is a large collection of real numbers. Searching the entire 3D infinite space without knowing the step size is not easy for this search method. Hence, the traditional Hill Climbing algorithm is modified to adjust the step size and the direction of search dynamically to make the search faster. The algorithm maintains the direction of search as long as it is approaching the optimal point with increase in step size. When it fails in its prediction, the search is randomized in all directions with decrease in step size. This predictive search method reduces the search time and is effective for a very large search space. This paper explains how the algorithm is successful in tuning PID controller and the performance of the algorithm is analyzed.**

***Index Terms*—Hill climbing, PID tuning, search algorithm.**

## I. Introduction

Search algorithms are used in many applications to find a solution in the search space based on the defined constraints. There are many algorithms like Hill Climbing, Genetic Algorithm, Simulated Annealing, etc. which are used depending on the type of application. Although Hill Climbing suffers from a drawback of getting stuck at the local maxima/minima, it is used in applications such as wind power system [1], evacuation route planning [2], modelling of soil behavior [3], forward planning [4] and many other applications because it is simple and easy to implement. There are different variants of Hill Climbing algorithm and these variants have evolved based on the need to make the search effective in addressing a particular problem. For example, Variable Step Hill Climbing algorithm is already being used to track Maximum Power Point of Wind Power System [1]. In this case, the searching time is reduced by varying the step based on a certain formula. This idea is extended to a problem where multiple variables are to be altered. One such example is PID tuning.

PID controller is a very popular control strategy used in many industrial applications. Though PID is simple in its operation, selecting the proper PID gains is difficult and time-consuming. In some cases, the mathematical model of the plant is available and PID gains can be derived based on few methods like Pole Placement or some PID tuning rules. If the plant model is not known, tuning the PID manually by trial and error method is very tedious. PID values can be of any order such as $10^{-3}$ or even $10^{3}$. With the help of thumb rules, manual tuning can be made easier but it is not very helpful when the system is complex.

However, many technical papers have described how different search or optimization techniques like Genetic Algorithm [5], [6], Neural network [7], Particle Swarm [8], etc. are used to tune PID. In [5] and [6], the Genetic Algorithm's performance in PID tuning depends on the following factors: size of the initial population, if the initial population has sufficient variety of good candidate solutions resulting in good crossover, and the method of crossover. Sometimes, it is difficult to create the initial population for an application like PID tuning because the search space is very large and determination of initial population with good PID values is not practically feasible. In [7], it is mentioned that the training periods are too long to identify the weights in the network because it uses supervised learning algorithm. As explained in [8], a modified Particle Swarm Optimization (PSO) is used to eliminate the problem of getting stuck at the local maxima/minima. Optimization methods such as PSO require more computational time and space. There are a few applications where such complex techniques are not needed to solve the problem. For such applications, Hill Climbing is used to tune PID since it is simple and easy to implement. This paper primarily focusses on the development of a generic Hill Climbing variant to solve a problem like PID tuning where the plant model is a complete black box. As described in [9], the PID tuning can be made more efficient by including prior knowledge of the system to help the algorithm find solutions faster. But, in order to make the algorithm generic, no thumb rules are used in tuning the controller heuristically. This paper demonstrates a method that not only solves the PID tuning problem but also other similar problems.

Another optimization problem similar to PID tuning is model parameter estimation in the field of system identification. As already mentioned in [3], there are a few instances where the exact values of model parameters are not derived, instead they are found in the heuristically defined search space. By this method, the model is fine-tuned to simulate accurate behavior of the system. In this case, the dimension of the search space depends on the number of model parameters to be estimated. One more example is the determination of optimal input-output scaling factors for a fuzzy system. In [10], different algorithms are used to find the optimal values of the scaling factors and their performance is compared. The method illustrated in this

Manuscript received November 7, 2017; revised January 18, 2018. This work was supported in part by the Department of RD I/CEP, Mercedes-Benz R&D India Pvt. Ltd.

Karthikeyan Nagarajan is with the Modeling & Simulation Team, RD I/CEP, Mercedes-Benz R&D India Pvt. Ltd., Bangalore, Karnataka, India (e-mail: nk_karthi@ymail.com).

doi: 10.18178/ijmlc.2018.8.1.656

paper can also be applied to the types of problems mentioned above.

In [11], Modified Hill Climbing uses an approach to alter one variable at a time and continues the search in both directions (increasing and decreasing values). Variables are changed one at a time, independent of other variables. But the step size is not modified and remains constant for each variable. There is no prediction involved while searching for the solution. Another variant of Hill Climbing is mentioned in [1] which uses a heuristic method of altering the step size to find the solution specifically for wind power systems. In view of the above mentioned Hill Climbing variants, a predictive method of Hill Climbing is proposed in this paper. It does not involve a specific heuristic approach to solve the problem and so it is a generic method that can be applied to any problem. Unlike [11], all the variables are changed simultaneously but the total magnitude of those changes is equal to the step size which is not constant. So this algorithm can be considered as an extension of [1].

The rest of this paper is organized as follows. Section II describes the problem statement and formulation of optimization function for PID tuning. Section III elaborates the concept of the proposed algorithm and the analysis of its results. Section IV demonstrates how the algorithm is applied to tune PID for a fixed wing airplane model. Section V is the conclusion of this work.

## II. Problem Description

The algorithm is developed in MATLAB script and the closed loop system comprising PID and the plant is modeled in MATLAB Simulink. The plant model defined by the transfer function (1) is randomly selected first and the algorithm is developed using that plant model in the simulation environment. Then the algorithm is applied to tune PID for a fixed wing airplane model [12] defined by the transfer function (2).

$$E(s) = \frac{1}{s^4+4s^3+14s^2+6s+1} \quad (1)$$

$$P(s) = \frac{\theta(s)}{\Delta(s)} = \frac{1.151s+0.1774}{s^3+0.739s^2+0.921s} \quad (2)$$

The optimization function is formulated based on the performance evaluation criteria comprising Overshoot, IAE, ISE, ITAE, Settling time and Oscillations. Each criterion has a weightage and these are adjusted based on the requirements of closed loop performance. The function output is normalized and ranges between 0-100 percent. The drawback of Hill Climbing Algorithm is that it gets stuck at local maxima/minima and this can be overcome by formulating a proper optimization function. It is important to select the weightages in such a way that the function is monotonically increasing and has only one maximum/minimum. The function should also output non-zero values for a major portion of the search space. Instead, if the function has a sudden hill arising abruptly covering a small area, then finding the initial non-zero value (foothills) by random search is time-consuming. In this case, the search is not effective when the appropriate initial values to tune PID are not known. For example, if the weightages are chosen only to avoid overshoot, this will result in the function having 0 percent value covering a large search space. This will lead to a randomized search instead of Hill Climbing in the beginning of the search. In this case, it is advisable to have non-zero weightage for IAE since it will contribute to non-zero value for most of the PID gains in search space. But, considering only IAE will result in a state where the function will have many local maxima/minima because a particular IAE can be achieved by multiple set of PID gains in the space. So it is very important to formulate the optimization function by selecting proper weightages in case of PID tuning.

## III. Predictive Hill Climbing Algorithm

### A. Overview

The proposed Hill Climbing algorithm is basically a best-first search method where the neighbor is selected randomly and the algorithm moves to the neighbor where it finds an improvement. Since it has infinite number of neighbors in this case, it is not possible to evaluate all the neighbors and choose the best among them. During the start, if the start point lies on a flat surface, the algorithm does a randomized search in the entire space till it finds the foothills of the optimization function. For example, this can happen if the optimization function has constant value covering a large search space and a sudden peak in a small region as mentioned previously.

The algorithm is developed to search in 3D space and each set of PID gains is denoted by spherical coordinate system (3). The representation is defined by the step size ($r$), which is also referred to radius of search, and two angles ($\theta_1$, $\theta_2$). The current point is considered as the center of the sphere. The region of search is bounded by the upper and lower limit of radius [$r_{LL}$, $r_{UL}$] and two angles [$\theta_{1LL}$, $\theta_{1UL}$, $\theta_{2LL}$, $\theta_{2UL}$] and a neighbor is randomly selected in that region. This actually defines a small portion of a solid spherical surface. The main reason for using limits is to control the region of search by altering the limits individually as explained below.

$$\begin{aligned} Kp_{x+1} &= Kp_x + (r * \cos\theta_1 * \sin\theta_2) \\ Ki_{x+1} &= Ki_x + (r * \sin\theta_1 * \sin\theta_2) \\ Kd_{x+1} &= Kd_x + (r * \cos\theta_2) \end{aligned} \quad (3)$$

where $r$, $\theta_1$ and $\theta_2$ are randomly selected in the ranges [$r_{LL}$, $r_{UL}$], [$\theta_{1LL}$, $\theta_{1UL}$] and [$\theta_{2LL}$, $\theta_{2UL}$] respectively.

The logic of the algorithm is shown in Fig. 1. It is explained in the following sections in detail. The algorithm differs from the traditional method in terms of:

- ✓ Step Size Adjustment
- ✓ Focused Directional Search

### B. Step Size Adjustment

If the algorithm notices successive improvements during the search, the lower limit of radius ($r_{LL}$) is increased by some factor, say 2. A constant radius width ($r_{width}$) is considered and therefore the upper limit ($r_{UL}$) is also increased accordingly.

Successive improvements in search may indicate that the solution can be found faster if the search progresses in same direction with increased step size. The threshold that defines the minimum number of successive improvements required

to increase the step size is called 'Positive Trial Threshold' (α). This is an important parameter to be defined according to the given problem in order to obtain efficient search results. However, the $r_{LL}$ can increase till it reaches the maximum value ($r_{LL\_max}$) since it cannot be too large to step out of the entire space in few steps.

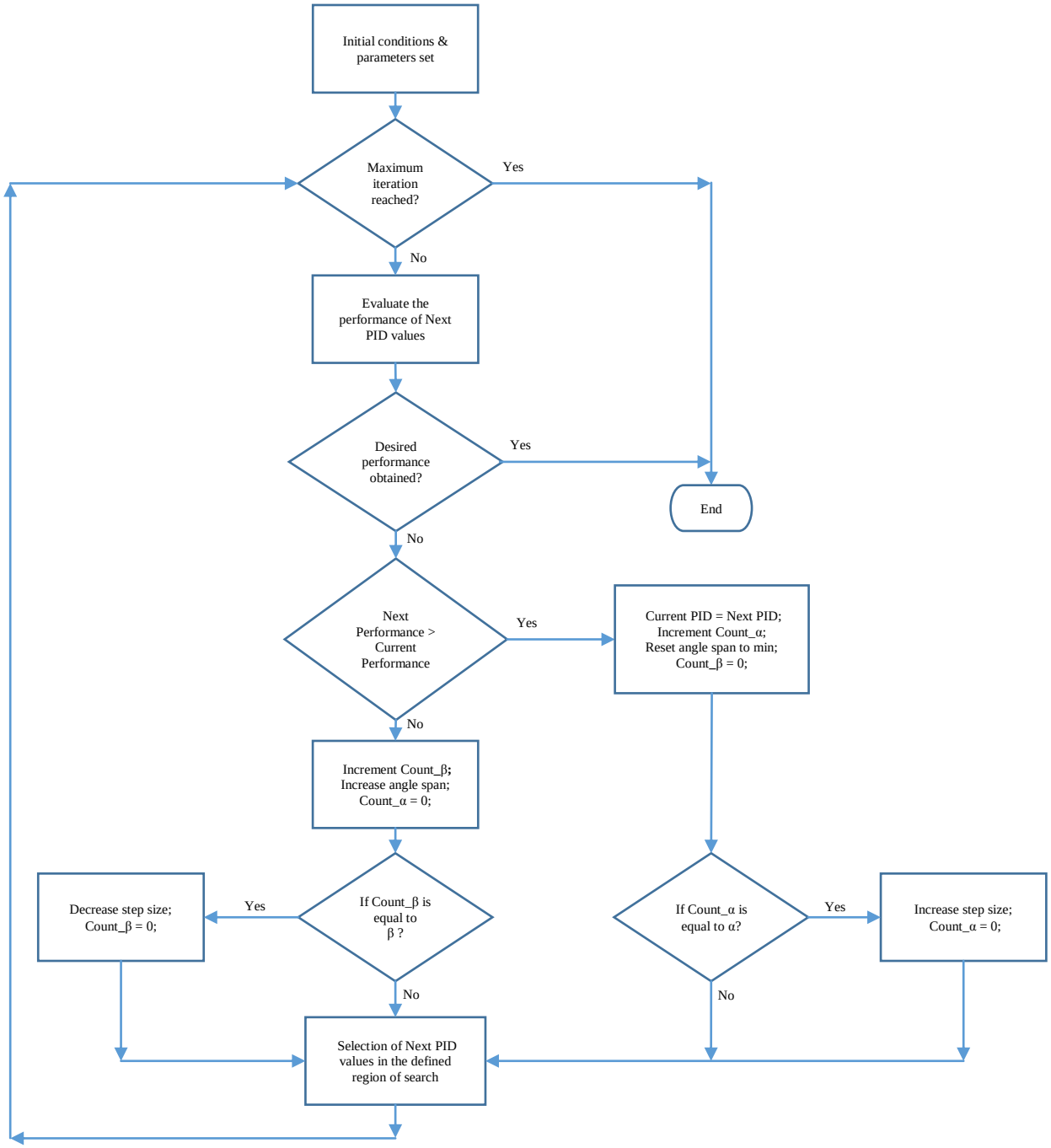


Fig. 1. Flowchart of the algorithm.

If the algorithm observes no successive improvements, the lower limit of radius ($r_{LL}$) is decreased by a factor, say 2. Anticipating an improvement, the neighbors are chosen very close to the current point. As the algorithm reaches the top of the hill, the radius has to be reduced to search nearby points to find the peak and not to fall off the other side of the hill by leaping too much. There is a minimum radius limit ($r_{LL\_min}$) defined for an optimization problem to stop the reduction of radius after that limit. If $r_{LL\_min}$ threshold is not defined then the algorithm keeps reducing the radius and continues the search where such small perturbations will not have any impact. It is assumed that if the algorithm does not find any improvement with the minimum radius, then it has reached maximum/minimum. The threshold that defines when the radius is to be reduced is called 'Negative Trial Threshold' (β). This is also one of the parameters that defines the performance of the algorithm.

### C. *Focused Directional Search*

As mentioned above, the two angles ($\theta_1$, $\theta_2$) define the direction of the search. By fixing the limits of these two parameters, the search can be localized or it can cover the entire spherical surface defined by the radius. When the algorithm detects an improvement, both the limits of the angles are brought close together and a focused search is performed in that direction. Using (4), the window of search is narrowed down by reducing the span ($\theta_{1width}$, $\theta_{2width}$) of $\theta_1$

and $\theta_2$ to a minimum value ($\theta_{1\text{width\_min}}$, $\theta_{2\text{width\_min}}$) with current $\theta_1$ and $\theta_2$ as the center.

$$\theta_{LL} = \theta - \left(\theta_{width}/2\right);\ \theta_{UL} = \theta + \left(\theta_{width}/2\right) \quad (4)$$

The algorithm increases the width ($\theta_{1\text{width}}$, $\theta_{2\text{width}}$) by a factor ($\theta_{1\Delta\text{width}}$, $\theta_{2\Delta\text{width}}$) for each 'no improvement' condition. According to (5), the ($\theta_{1\Delta\text{width}}$, $\theta_{2\Delta\text{width}}$) are chosen in such a way that the entire spherical surface is searched before the radius is reduced by 'Negative Trial Threshold' parameter.

$$\theta_{width_{x+1}} = \theta_{width_x} + \theta_{\Delta width}$$

$$\theta_{\Delta width} = \theta_{width\_max}/(0.5 * \beta) \quad (5)$$

where

$$\theta_{width} \leq \theta_{width\_max}$$

$$\theta_{1width\_max} = 360\ ;\ \theta_{2width\_max} = 180\ ;$$

$$\theta_{width\_min} = \theta_{\Delta\text{width}}$$

The equation (5) states that the full surface search will start at iteration equal to half of $\beta$ after the first occurrence of no-improvement case. This means, randomized search of the entire surface is performed for $\beta$/2 iterations before the radius is reduced. This is to ensure that the algorithm searches thoroughly with a given radius limits even if the prediction of focused search fails.

D. *Analysis*

For analysis purpose, the weightages chosen for tuning PID for the plant (1) are 50% for Overshoot and 50% for Oscillation and initial condition is *Kp* = 12.95; *Ki* = 8.44; *Kd* = 31.07. The maximum number of iterations allowed to find the desired solution is 300. The parameters of the algorithm used for this model are shown in Table I.

TABLE I: ALGORITHM PARAMETERS

| ***Parameter*** | ***Value*** |
|---|---|
| $\beta$ | 10 |
| $\alpha$ | 3 |
| $r_{\text{LL\_min}}$ | 0.01 |
| $r_{\text{LL\_max}}$ | 10 |
| $r_{\text{width}}$ | 10 |
| Search Space for *Kp*, *Ki*, *Kd* | [0.01,10] |

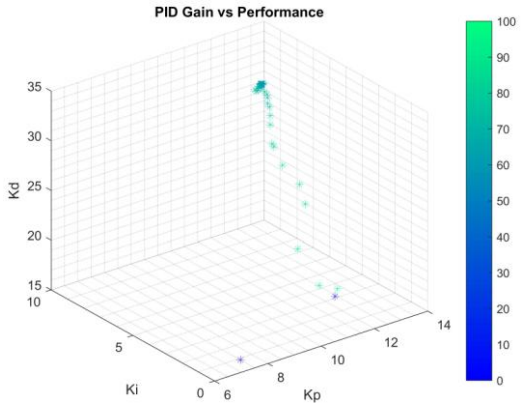


Fig. 2. Search path traced before finding the desired solution.

The following graphs are obtained when the algorithm is used with the above setting to tune PID for the plant (1). Fig. 2 shows how the search progressed and the solution found is *Kp* = 10.99; *Ki* = 0.61; *Kd* = 19.32. Fig. 3, Fig. 4 and Fig. 5 explains the adjustment of limits of radius and the angles of search ($\theta_1$, $\theta_2$) respectively. The red and blue markers in the plots denote upper and lower limits respectively and black markers denote the random selection of a value between those limits.

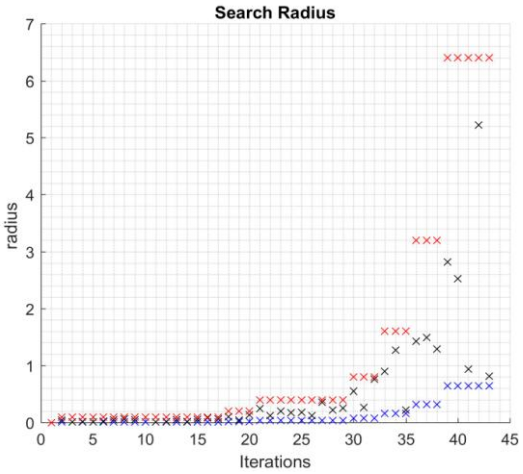


Fig. 3. Step size limit adjustment.

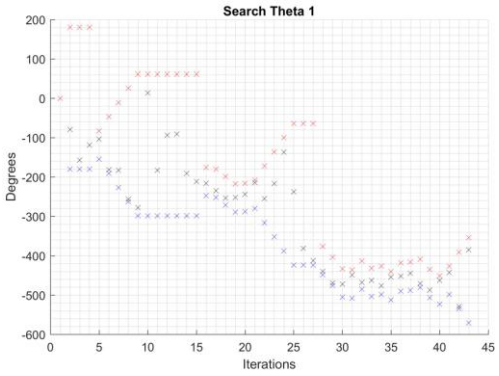


Fig. 4. $\theta_1$ Limit changes.

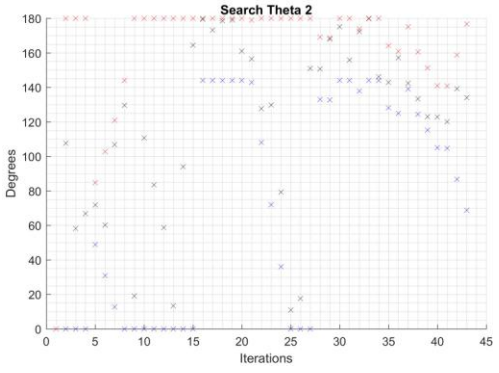


Fig. 5. $\theta_2$ Limit changes.

It is important to select $\alpha$ and $\beta$ carefully because it determines the rate at which the step size increases and decreases respectively. A very low $\alpha$ will increase the step size unnecessarily and result in jumping over the peaks. A high value of $\alpha$ will result in a sluggish search. Let's assume that $\beta$ is set to a very high value and the algorithm has reached close to the desired solution. In this case, the algorithm will waste too much time evaluating solutions in the region which is far away from the peak.

Conversely, if β is set to a very low value and the search direction is predicted incorrectly, this will result in a slow search since the step size will be reduced within a few iterations before exploring better candidate solutions. For better understanding, the impact of different α values is studied. Fig. 6 shows the average number of iterations required to find the solution for different α values. The optimum value is around 3.

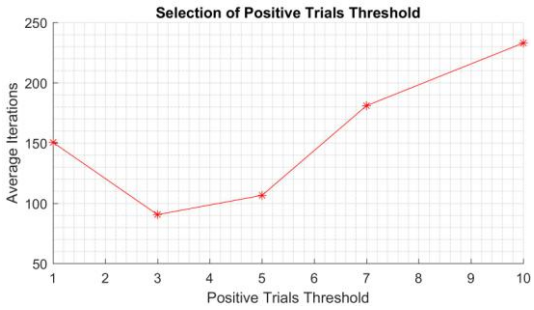


Fig. 6. Selection of Positive Trial Threshold.

The effect of the focused search on the performance is studied and explained below. The comparison of total number of iterations needed to find the solution with and without focused search for 10 trials is shown in Table II. Considering 99% and above as the desired solution and 300 as the maximum iteration limit, the probability of achieving the desired result with focused search is 1. The mean and standard deviation is 90.5 and 59.28 respectively. Meanwhile, the probability of achieving the desired result without focused search is 0.3. Although seven out of ten instances are unsuccessful in finding the desired solution, the mean is approximated as 272.7. It is clear that the focused search helps in converging to the desired solution faster since the average number of iterations is reduced and the probability is quite high for a given maximum iteration. The prediction works very well when the optimization function has a curve as specified in the previous section. Even though the search becomes faster, the standard deviation of 59.28 indicates that the search does not always follow the optimal path. The algorithm predicts the direction of optimal path and continues its search even when the prediction is wrong.

TABLE II: SEARCH IMPROVEMENT

| ***Without Focused Search*** | | ***With Focused Search*** | |
|---|---|---|---|
| ***Best solution*** | ***Iterations*** | ***Best solution*** | ***Iterations*** |
| 100 | 229 | 100 | 226 |
| 82.20 | 300[a] | 100 | 91 |
| 85.21 | 300[a] | 99.68 | 30 |
| 85.34 | 300[a] | 100 | 44 |
| 85.11 | 300[a] | 99.75 | 98 |
| 85.00 | 300[a] | 100 | 43 |
| 95.93 | 300[a] | 100 | 70 |
| 100 | 119 | 100 | 142 |
| 100 | 279 | 100 | 111 |
| 82.21 | 300[a] | 99.75 | 50 |
| ***Average*** | ***272.7*** | ***Average*** | ***90.5*** |

a. maximum iteration reached.

To summarize the efficiency of the algorithm, the space requirement is low and constant since it does not keep track of the search path traced. The algorithm is not always complete as it gets stuck at local maxima/minima. It is also not optimal since it looks for first-best neighbor instead of finding the best of all neighbors. Theoretically, time complexity is O (∞). The time efficiency can be measured in terms of the time taken to reach the global maximum/minimum. So, the algorithm can be made complete and efficient only if the optimization function is chosen properly. In addition to this, the choice of α and β determines time efficiency.

## IV. PID TUNING OF FIXED WING AIRPLANE MODEL

As stated earlier, a few experiments of PID tuning by Modified Hill Climbing (MHC) algorithms have been conducted. One such work where MHC is used for tuning PID for the plant model (2) is described in [11]. In this paper, the closed loop requirement is to have zero overshoot and settling time less than 25s with steady state tolerance band of ±0.01. The weights defined in the proposed algorithm are 50% for overshoot and 50% for settling time. The parameter values mentioned in Table I are also used for configuring the algorithm to solve this problem. The results of the experiment and performance analysis are tabulated in Table III and Table IV respectively. The experiments suggest that the optimization function for the plant (2) has more than one maximum. Like the traditional Hill Climbing, the proposed algorithm also yields different solutions when started with different initial values. The step responses of the closed loop system obtained with different PID solutions are shown in Fig. 7. Compared to the results shown in [11], the algorithm has shown better results and could find the following solutions within 300 iterations.

TABLE III: PID GAINS

| *Initial Values* | | | *Final values* | | | *Solution* |
|---|---|---|---|---|---|---|
| *Kp* | *Ki* | *Kd* | *Kp* | *Ki* | *Kd* | *Name* |
| 1 | 1 | 1 | 2.12 | 0.90 | 1.61 | Solution 1 |
| 5 | 5 | 5 | 6.92 | 2.09 | 3.45 | Solution 2 |
| 10 | 10 | 10 | 23.95 | 4.74 | 7.79 | Solution 3 |

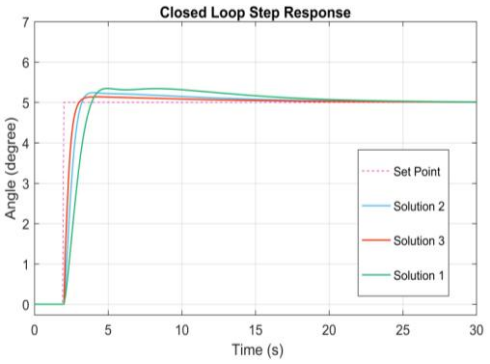


Fig. 7. Closed loop responses with the PID tuned by the algorithm.

TABLE IV: PID PERFORMANCE ANALYSIS

| ***Solution Name*** | ***Overshoot (%)*** | ***Settling Time (s)*** |
|---|---|---|
| Solution 1 | 6.8 | 27.8 |
| Solution 2 | 4.8 | 25.5 |
| Solution 3 | 2.6 | 23.9 |

## V. CONCLUSION

In this paper, traditional Hill Climbing algorithm is

modified by including a predictive logic which improves the search. The importance of formulating the optimization function for PID tuning is also explained. The implementation of the algorithm is described and experiments are conducted to demonstrate the effect of the important algorithm parameters. The algorithm is applied to tune PID which operates on fixed wing airplane model. The results suggest that the proposed algorithm is very effective when there are multiple real numbered variables to be tuned and the search space is very large. The algorithm is generic, hence, it can be applied to other similar problems like PID tuning. Although the algorithm is efficient, it has a drawback that it gets stuck at local maxima/minima like the traditional Hill Climbing algorithm. This drawback can be overcome by extending this algorithm to Iterative Predictive Hill Climbing where multiple searches are performed with randomly selected start points.

## ACKNOWLEDGMENT

Thanks to Daimler and MBRDI members for making this paper possible.

## REFERENCES

[1] F. Lan, L. Tao, J. Li, and Z. Yao, "An improved variable step hill-climbing searching algorithm for tracking maximum power point of wind power system," in *Proc. 2016 China International Conference on Electricity Distribution (CICED)*, Xi'an, 2016, pp. 1-6. d

[2] T. P. Jiang, G. Ren, and X. Zhao, "Evacuation route optimization based on Tabu search algorithm and hill-climbing algorithm," *Procedia - Social and Behavioral Sciences*, vol. 96, pp. 865-872, 2013.

[3] D. M. G. Taborda and L. Zdravkovic, "Application of a Hill-Climbing technique to the formulation of a new cyclic nonlinear elastic constitutive model," *Computers and Geotechnics*, vol. 43, 2012, pp. 80-91.

[4] S. A. Akramifar and G. Ghassem-Sani, "Fast forward planning by guided enforced hill climbing," *Engineering Applications of Artificial Intelligence*, vol. 23, issue 8, pp. 1327-1339, 2010.

[5] R. W. Wies, E. Chukkapalli, and M. Mueller-Stoffels, "Improved frequency regulation in mini-grids with high wind contribution using online genetic algorithm for PID tuning," in *Proc. 2014 IEEE PES General Meeting | Conference & Exposition, National Harbor*, MD, 2014, pp. 1-5.

[6] N. F. Mohammed, E. Song, X. Ma, and Q. Hayat, "Tuning of PID controller of synchronous generators using genetic algorithm," in *Proc. 2014 IEEE International Conference on Mechatronics and Automation*, Tianjin, 2014, pp. 1544-1548.

[7] R. Hernandez-Alvarado, L. G. Garcia-Valdovinos, T. Salgado-Jimenez, A. Gómez-Espinosa, and F. F. Navarro, "Self-tuned PID control based on backpropagation Neural Networks for underwater vehicles," *OCEANS 2016 MTS/IEEE Monterey*, Monterey, CA, pp. 1-5, 2016.

[8] K. Singh and G. Shankar, "PID parameters tuning using modified particle swarm optimization and its application in load frequency control," in *Proc. 2016 IEEE 6th International Conference on Power Systems (ICPS)*, New Delhi, 2016, pp. 1-6.

[9] J. Chen, M. N. Omidvar, M. Azad, and X. Yao, "Knowledge-based particle swarm optimization for PID controller tuning," in *Proc. 2017 IEEE Congress on Evolutionary Computation (CEC)*, San Sebastian, 2017, pp. 1819-1826.

[10] D. K. Sambariya, R. Gupta, and R. Prasad, "Design of optimal input–output scaling factors based fuzzy PSS using bat algorithm," *Engineering Science and Technology, an International Journal*, vol. 19, issue 2, pp. 991-1002, 2016.

[11] A. Abdulelah, A. C. Soh, N. A. Abdullah, M. K. Hassan, and S. B. M. Noor, "Simulated real time controller using modified hill climbing algorithm on fixed wing airplane," in *Proc. 2015 10th Asian Control Conference (ASCC)*, Kota Kinabalu, 2015, pp. 1-5.

[12] Control tutorials for MATLAB and Simulink - aircraft Pitch: System Modeling. Published with MATLAB® 7.14. (2012). [Online]. Available: http://ctms.engin.umich.edu/CTMS/index.php?example=AircraftPitch&section=SystemModeling

**Karthikeyan Nagarajan** is currently working as a modeling & simulation engineer in Mercedes Benz R&D, India. He received his bachelor of engineering degree from the Instrumentation Department, Madras Institute of Technology, Chennai, India in 2013. He has worked at Delphi Technical Center, Bangalore as a software engineer from 2013-2015.

His research interests include control system, artificial intelligence and autonomous driving.