Disturbance Mitigation PID Speed Control System Using Compact Differential Evolution Algorithms

Sunday Iliya[1], Olurotimi Olakunle Awodiji [2], Geraldine Rangmoen Rimven[3]

sundayiliyagoteng@yahoo.com, awodijiio@unijos.edu.ng, daloengg@unijos.edu.ng

Abstract

This research paper present an efficient and robust way of tuning PID controller using five variants of compact Differential Evolution (cDE) algorithms for speed control of a dynamic system, with major focus on external disturbance mitigation. Control systems should not only be designed to track the command input (desired speed), but they should also be evaluated on the basis of their ability to mitigate the effects of external disturbance resulting from both natural sources (winds, thunder strike, etc), and man oriented events (e.g. explosive, etc), and also the effects of internal disturbance due to system parameters deviation from the optimal. Many catastrophic effects resulting from failure of control systems is often traced to inability of the system to maintain stability due to the effects of external disturbance or system parameter deviation. Out of the five variants implemented, cDE/rand/2/exp using exponential crossover appear to be more promising in addressing this problem with root mean square error (RMSE) of **16.45 RPM,** maximum disturbance unit step amplitude of 0.00133 that rapidly decayed to zero within 1 second. The second best algorithm is cDE/rand/2/bin using binomial crossover with RMSE of **16.45 RPM,** and maximum disturbance unit step amplitude of 0.00136**.** One of the major advantage of using compact differential evolution in this research as compared with the population based variants is that cDE can be implemented using single chip microcontrollers with limited memory space to effectively fine-tuned PID gains for real time control of complex systems. The controller implemented in this research depicted a robust performance not only in tracking the command input (desired speed), but also in mitigating the effect of external disturbance.

Index Terms: — Compact Differential Evolution algorithms, PID controller, Step response, Ziegler–Nichols tuning method, optimization, objective fitness function, External disturbance.

1. Introduction

ONe of the major challenge of any speed control system, is the ability to coup with unpredictable changes resulting from within the system or its environment or target speed. Speed control systems are face with different challenges depending on their application and the environment they are designed to be used, among these are disturbance from natural events such as wind, unpredictable change of their target speed for non-stationary target, changes resulting from system model parameters, etc. To mitigate the chances of the system missing its target, we proposed a generalised intelligent control schemes for speed control systems that uses compact Differential Evolution (cDE) optimization algorithm to tune the PID gains of a speed controlled system in a dynamically changing environment.

2. OPTIMIZATION OR TUNING ALGORITHMS

A brief description of the optimization algorithms implemented are presented in this section. The advantages of global search capability of compact Differential Evolutionary (cDE) algorithms variants were explored to evolve the gains of the PID controller. The complexity of many heuristic controllers becomes increasingly complicated due to meta parameters (free parameters) in the model or controller frame work that govern their behaviour, and efficiency in optimizing a given problem. How best a given controller can solve a given problem, depends on the correct choice of the meta parameters. The values of those parameters are problem dependent, thus for each problem, those parameters need to be fined tune to get the optimum or near optimum. The tuning imposed another optimization problem. The PID gains of the speed control system depicted in this paper were optimized using compact differential evolution (cDE) algorithms.

3. Compact Differential Evolution (cDE)

The population based DE algorithm variants are not memory efficient even though they are often more robust and accurate in solving many optimization problems than the compact variants. The compact version of DE which can easily be implemented in embedded systems with limited memory constrained is used in this study. Compact differential evolution (cDE) algorithm is achieved by incorporating the update logic of real values compact genetic algorithm (rcGA) within DE framework [9]. The steps involves in cDE is as follows: A (2 x n) probability vector **PV** consisting of the mean **μ** and standard deviation **σ** is generated. Where n is the dimensionality of the problem. At initialization, μ was set to 0 while σ was set to a very large value 10, in order to simulate a uniform distribution. A solution called the elite is sampled from the **PV**. At each generation (step), other candidate solutions are also sampled from the **PV** according to the mutation schemes adopted e.g. for DE/rand/1 three potential candidate solutions X_{r1}, X_{r2} and X_{r3} are sampled. Without lost of generality, each designed variable $X_{r1}[i]$ belonging to a candidate solution X_{r1}, is obtained from the **PV** as follows: For each dimension indexed by i, a truncated Gaussian probability density function (PDF) with mean μ and standard deviation σ is assigned. The truncated PDF is defined by Eq. (1).

The cumulative density function (CDF) of the truncated PDF is obtained. A random number rand(0,1) is sampled from a uniform distribution. $X_{r1}[i]$ is obtained by applying the random number rand(0,1) generated to the inverse function of the CDF. Since both the PDF and CDF are truncated or normalized within the range [-1, 1]; the actual value of $X_{r1}[i]$ within the true decision space of [a, b] is obtain as $(X_{r1}[i]+1)\frac{(b-a)}{2}+a$. The mutant (provisional offspring) is now generated using the mutation schemes. The offspring is evolved by performing a crossover operation between the elite and the provisional offspring as described in Section 3.2. The fitness value of the offspring is computed and compare with that of the elite. If the offspring outperform the elite, it replaces the elite and declared the winner while the elite the loser; otherwise the elite is maintained and declared the winner while the offspring the loser. In this study, the fitness function is the weighted sum of the peak overshot, rise time and settling time obtain using step input command as depicted by Eq (7). The **PV** is updated using Eq. (1). Hence in cDE, instead of having a population of individuals (candidate solutions) for every generation as in normal DE, the population are represented by their probability distribution function (i.e. their statistics), thus minimizing the computational complexity, amount of memory needed, and the optimization time. cDE consist of only one candidate solution called the elite, which is either maintained or replaced by its offspring in the next generation subject to its fitness.

$$PDF(\mu[i],\sigma[i]) = \frac{e^{\frac{-(x-\mu[i])^2}{2\sigma[i]^2}}\sqrt{\frac{2}{\pi}}}{\sigma[i](\mathrm{erf}\left(\frac{\mu[i]+1}{\sqrt{2}\sigma[i]}\right)-\mathrm{erf}\left(\frac{\mu[i]-1}{\sqrt{2}\sigma[i]}\right))}$$

(1)

$$\mu^{t+1}[i] = \mu^t[i] + \frac{winner[i]-loser[i]}{Np}$$

$$\sigma^{t+1}[i] = \sqrt{(\sigma^t[i])^2 + \delta[i]^2 + \frac{winner[i]^2 - loser[i]^2}{Np}}$$

Where $\delta[i]^2 = (\mu^t[i])^2 - (\mu^{t+1}[i])^2$, *t*=generation, *Np* is virtual population while *erf* is the error function

3.1 Mutation

For every individuals (target vectors) $X_{i,G}$ at generation G, a mutant vector $V_{i,G}$ called the provisional or trial offspring is generated via certain mutation schemes [3][6][7]. Some of the mutation strategies commonly implemented in DE are given by Eqs. (2) to (5). while the ones used in this research are depicted by Eqs. (2), (3) and (5).

DE/rand/1

$$V_{i,G} = X_{r1,G} + F.(X_{r3,G} - X_{r2,G}) \qquad (2)$$

DE/best/1:

$$V_{i,G} = X_{best,G} + F.(X_{r2,,G} - X_{r1,G}) \qquad (3)$$

DE/rand-to-best/1:

$$V_{i,G} = X_{i,G} + F.\left(X_{best,G} - X_{ri,G}\right) + F.(X_{r2,G} - X_{r1,G}) \qquad (4)$$

DE/rand/2:

$$V_{i,G} = X_{r1,G} + F.\left(X_{r3,,G} - X_{r2,G}\right) + F.(X_{r5,G} - X_{r4,G}) \qquad (5)$$

Where the indexes r1, r2, r3, r4 and r5 are mutually exclusive positive integers and distinct from i. These indexes are generated at random within the range [1 - PN]. $X_{best,G}$ is the individual with the best fitness at generation G while F is the mutation constant.

3.2 Cross Over

After the mutants were generated, the offspring $U_{i,G}$ are produced by performing a crossover operation between the target vector $X_{i,G}$ and its corresponding provisional offspring $V_{i,G}$. The two crossover schemes i.e. exponential and Binomial crossover are used in this study for all the cDE algorithms implemented. The Binomial crossover copied the jth gene of the mutant vector $V_{i,G}$ to the corresponding gene (element) in the offspring $U_{i,G}$ if rand(0,1) ≤ CR or $j=j_{rand}$. Otherwise it is copied from the target vector $X_{i,G}$ (parent). The crossover rate CR is the probability of selecting the offspring genes from the mutant while j_{rand} is a random integer in the range [1 - D], this ensure that at least one of the offspring gene is copied from the mutant. D is the dimension of the problem i.e. number of genes (elements) in each candidate solution. If CR is small it will result in exploratory moves parallel to a small number of axes of the decision space .i.e. many of the genes of the offspring will come from its parent than from the mutants, consequently the offspring will resemble its parent. In this way, the DE will serve as a local searcher as it bear strong exploitative capabilities than being explorative. On the other hand, large values of CR will lead to moves at angles to the search space's axes as the genes of the offspring are more likely to come from the provisional offspring (mutant vector) than its parent. This will favour explorative moves. The Binomial crossover is represented by Eq. (6).

$$U_{iG}^{j} = \begin{cases} V_{iG}^{j} \; if \; \; rand(0,1) \leq CR \; or \; j = j_{rand} \\ X_{iG}^{j} \; Otherwise \end{cases} \quad (6)$$

For exponential crossover, the genes of the offspring are inherited from the mutant vector $V_{i,G}$ starting from a randomly selected index j in the range [1 - D] until the first time rand(0,1) > CR after which all the other genes are inherited from the parent $X_{i,G}$ [6][7][3].The exponential crossover is as shown in Algorithm 1.

```
$U_{j,G} = X_{j,G}$
$Generation\ j = randi(1\ ,D)$
$U_{i,G}^{j} = V_{i,G}^{j}$
$K = 1$
$\boldsymbol{while}\ rand(0,1) \leq Cr\ AND\ K < D\ \boldsymbol{do}$
    $U_{i,G}^{j} = V_{i,G}^{j}$
    $j = j + 1$
    $\boldsymbol{if}\ j = D$
        $j = 1$
    $\boldsymbol{end\ if}$
    $K = K + 1$
$\boldsymbol{end\ while}$
```

Algorithm 1: Exponential Crossover Pseudo code

Fitness Function Evaluation

The optimization problem presented in this paper is a multi-objectives optimization problem since there are three cost functions to be minimised i.e. the maximum overshot (M_o), rise time (T_r) and settling time (T_s). In order to get a robust controller gains, the problem is converted to single objective problem with one cost function consisting of the weighted sum of the three objective functions, Eq. (7). The weights depends on the important or cost of risk resulting from that particular performance index. This approach is robust because different models can be evolved by just changing the weight to meet up with setting system performance specifications.

$$\gamma = \alpha_o M_o + \alpha_r T_r + \alpha_s T_s \quad (7)$$

Where: γ is the combined or overall fitness function, M_o is the maximum overshot, T_r is the rise time and T_s settling time, while α_o, α_r, and α_s are their weights respectively. For this research, after a manual tuning, the following values were used with M_o having the highest priority, α_o =0.8, α_r=0.1, and α_s=0.1. Note, the maximum value the weight can take for this application is 1. This can differ for other models or application.

4. Proportional Plus Integral Plus Derivative (PID) Controller

It is interesting to know that nearly half of the industrial controllers used today are PID or modified PID or derivatives of PID controllers. Some intelligent controllers e.g. Fuzzy logic or adaptive fuzzy logic are derivatives of basic PID i.e. they make use of the error and its derivative (rate of change of the error). There are different variant of the PID controller, the one used in this research is given by Eq. (8) while the transfer function $G_c(s)$ of the controller is depicted by Eq. (9) [2][1][4][5]. A proportional controller will have the effect of reducing the rise time, but will not eliminate the steady-state error. Because of the present of pole at the origin introduced by the integral controller, the integral controller will have the capability of eliminating the steady-state error, but it may make the transient response worse. The derivative controller will have the effect of increasing the stability of the system, reducing the overshoot, and improving the transient response. The derivative controller predict future error using the rate at which the error is changing while the integral captured the cumulative effects of past errors to improve future system performance.

$$PID = K_p\left(e(t) + \frac{1}{T_i}\int_{t_0}^{t} e(t)\,dt + T_d\frac{de(t)}{dt}\right) \tag{8}$$

$$G_c(s) = K_p\left(1 + \frac{1}{T_i s} + T_d s\right) \tag{9}$$

Where: t is time, e(t) is present error at time t, K_p is the proportional gain while T_i and T_d are integral and derivative time constants respectively, s is Laplace complex notation.

Tuning of the PID gains (K_p, T_i and T_d) using Ziegler–Nichols tuning method

The process of selecting the controller parameters K_p, T_i and T_d to meet a given performance specifications is known as controller tuning. Different variants of compact Differential Evolution (cDE) algorithms were used to evolve the PID gains. One of the major challenge is to define the decision search space i.e. the range within which each of the meta parameters (K_p, T_i and T_d) of the controller should be searched. To address this problem, Ziegler–Nichols tuning method was used to obtain the centroid of the radius of the search space. The Ziegler–Nichols reference gains were obtained using the mathematical model of the speed controlled system shown in Fig. (1). The centre of the radius for search of the gains K_p, T_i and T_d are given by equations (10), (11) and (12) respectively [2].

$$K_p = 0.6K_{cr} \tag{10}$$
$$T_i = 0.5P_{cr} \tag{11}$$
$$T_d = 0.125P_{cr} \tag{12}$$

Where K_{cr} and P_{cr} are the critical gain and critical frequency for self-sustained oscillation of the system.

The decision search space for each of the gains were obtained as follows:

$$K_{p(space)} = [\alpha_{min}K_p,\ \alpha_{max}K_p] \quad (11)$$

$$T_{i(space)} = [\beta_{min}T_i,\ \beta_{max}T_i] \quad (12)$$

$$T_{d(space)} = [\mu_{min}T_d,\ \mu_{max}T_d] \quad (13)$$

K_p, T_i and T_d are given by equations (8), (9) and (10) respectively while after a manual tuning, the minimum and maximum values of α, β and μ were obtained as follows:

$\alpha_{min} = 0.4,\quad \beta_{min} = 0.2,\ \mu_{min} = 0.2,\ \alpha_{max} = 5,\quad \beta_{max} = 4,\ \mu_{max} = 4$

5. Mathematical model of the speed controlled system

The rotation of the speed controlled system to meet up with a given target specifications is achieved using DC motor.

$$V = R_aI_a + L_a\frac{dI_a}{dt} + E_b \quad (13)$$

$$T = J\frac{dw}{dt} + Fw \quad (14)$$

$$E_b = K_bw \quad (15)$$

$$T = K_tI_a \quad (16)$$

$$w = \frac{d\Theta}{dt} \quad (17)$$

Where V is motor terminal supply voltage, R_a armature resistance, L_a is armature inductance, I_a is armature current, E_b is back emf (electromotive force), T is the torque, w is the angular speed in rad/s, J is the inertia constant while F is the viscose constant, K_b is the back emf constant, t is time and Θ is angular position in rad.

The block diagram shown in Fig. 1 was obtain using equations (13) to (17) along with the controller, where W_R is the command reference input speed while W is the actual controller output.

Disturbance T_D

One of the expected effect of the PID controller other than tracking the command input, is to mitigate the effect of disturbance due to external sources e.g. wind or other sources. The response of the output with respect to the disturbance T_D as input should die or decayed to zero within a very short time. The transfer function of the output W with respect to the disturbance $T_{D,}$ at command input $W_R = 0$ is given by Eq (18)

$$\frac{W}{T_D} = \frac{sL_a+R_a}{(sL_a+R_a)(sJ+F)+sK_t(G_C(s)+sK_b)} \quad (18)$$

Without PID, $G_C(s)$ =1 but with PID controller $G_C(s)$ is given by:

$$G_c(s) = K_p(1 + \frac{1}{sT_i} + sT_d)$$

All the parameters are the same as stated in Section 3.1.

6. Results

Each of the cDE variants were run for 50 generations consisting with a virtual population of 10. At the end of the generation, the must fitted (best) candidate is used to set the PID gains. The fitness function used during the tuning is the weighted sum of the maximum overshot, rise time and settling time, Eq. (7). The evolved best candidate was used to control the speed of the system. The system was tested using three different approaches, i.e. the system was tested using standard ram and parabolic input command. Thirdly a real world scenario was modelled as a command input to see how the output of the system can track the target input. The performance index used to evaluate the accuracy of the system in tracking the command input is the root mean square error (RMSE) given by Eq. (19) [8]. RMSE is used as the performance metric because it is expressed in the same unit as the measured variable. In addition to this, it is less sensitive to errors due to outlier data points as compare with Mean Square Error (MSE). Hence the RMSE is a good metric for model selection where the risk resulting from the error is proportional to the error [8]. It is interesting to note that, the fact that the system depicted good performance for standard ram and parabolic input with low RMSE does not necessarily mean that the system will perform well when subjected to real world scenario with unpredictable change in command input and system (model) parameters. This is revealed when the untuned controller obtain directly using Ziegler–Nichols method was used. The RMSE of ram and parabolic command using untuned PID shown in Fig. 4 and Fig. 5 are 0.0238 and 0.0040 respectively while for the tuned PID are 0.0236 and 0.0022 respectively. But when the tuned and the untuned PIDs were tested using real world command input scenario, the untuned PID perform poorly while the tuned PID followed the command input closely as shown in Fig. 3. This research also validate that PID gains obtained using Ziegler–Nichols method may not be the optimum but is a useful tool for obtaining the radius (domain) of the search space within which the optimum or near optimum are likely to be found. The details of the numeric results obtained from the five cDE variants implemented in this research are shown in table 1. The speed controlled system in this research is designed and simulated using MATLAB as shown in Fig. 8. The parameter of the model, and the meta parameters of the optimnization algorithms used are depicted in Fig. 8.

The controller implemented in this research depicted a robust performance not only in tracking the command input, but also in mitigating the effect of external disturbance with a maximum unit step response of 1.59 x 10^{-3}, and decayed rapidly to zero within 1 second as shown in Fig. 6. The normalised generational fitness function of the various cDE variants implemented are as shown in Fig. 7.

$$RMSE = \sqrt{\frac{1}{N}\sum_{i=1}^{N}(\mathrm{W}_{Ri} - \mathrm{W}_{i})^2} \qquad (19)$$

Where: RMSE is the root mean square error, N is the number of simulation time steps, W_{Ri} and W_{i} are the command input and the actual output at time index i respectively.

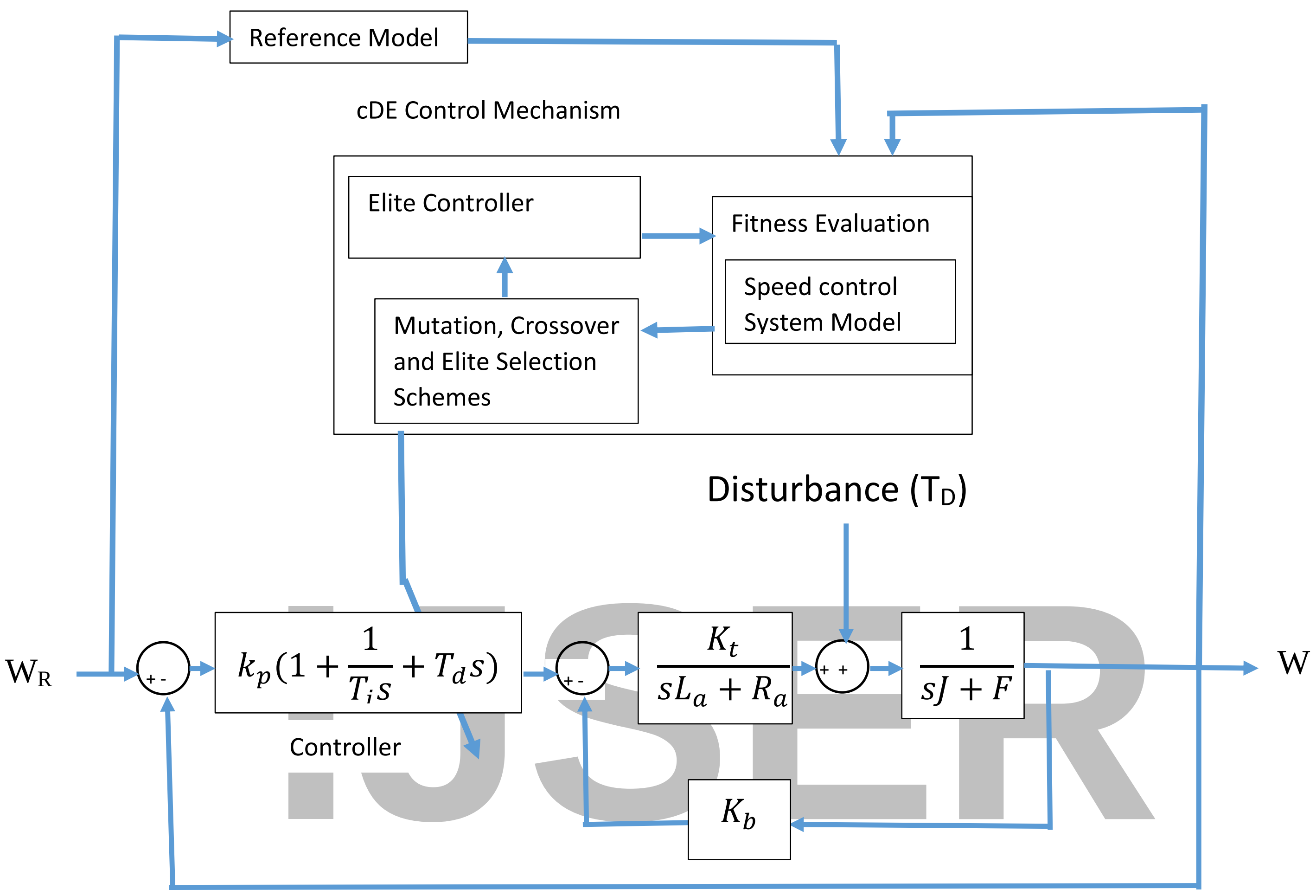


Fig. 1: Block diagram of the speed control system

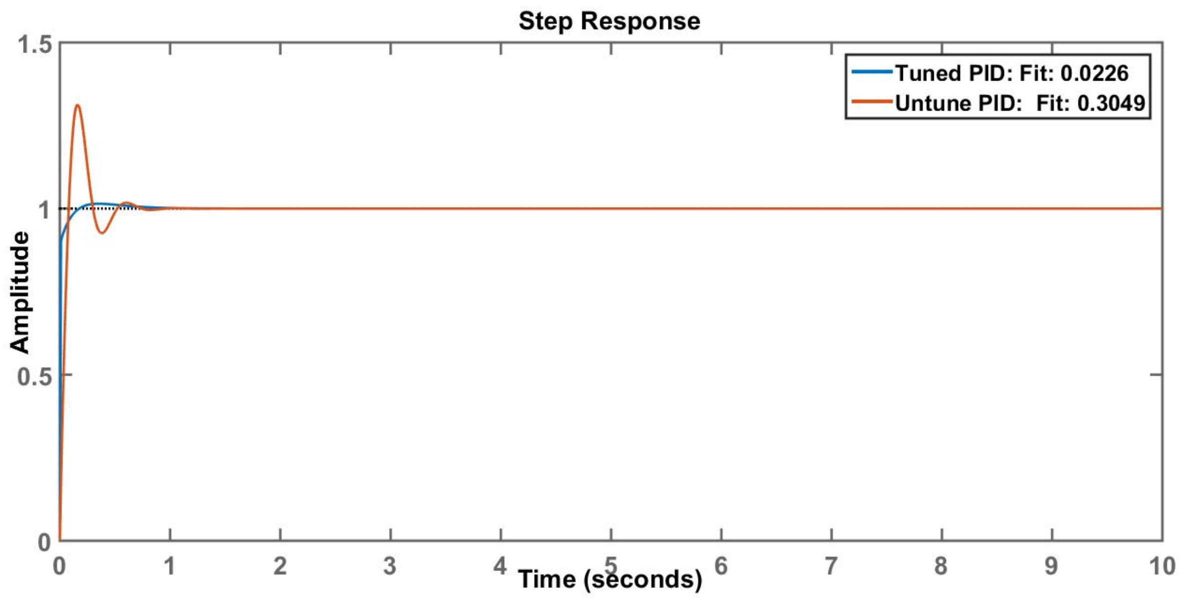


Fig. 2: Unit step response using: tuned PID and untune PID

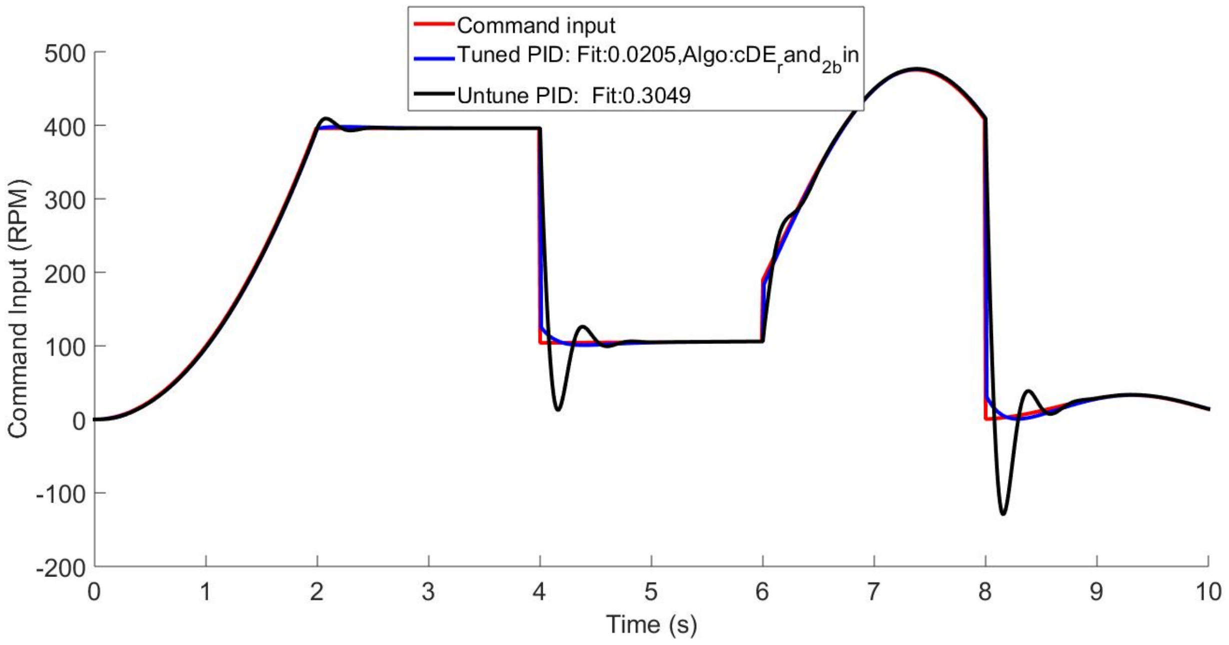


Fig 3: Real world command input using tuned and untune PID

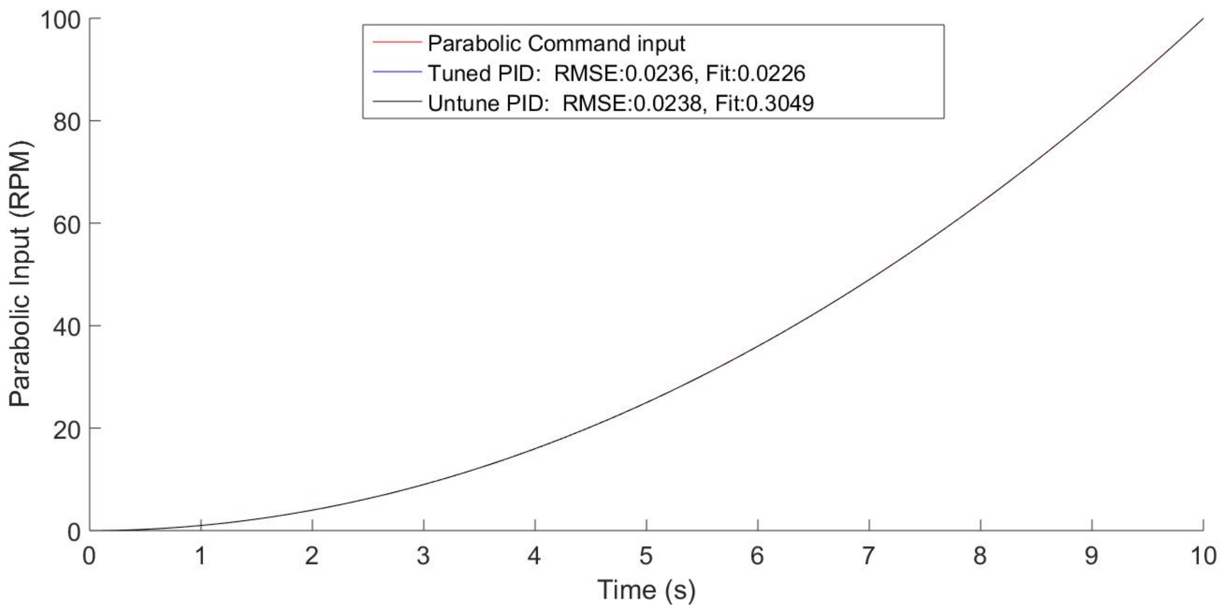


Fig. 4: Parabolic input command

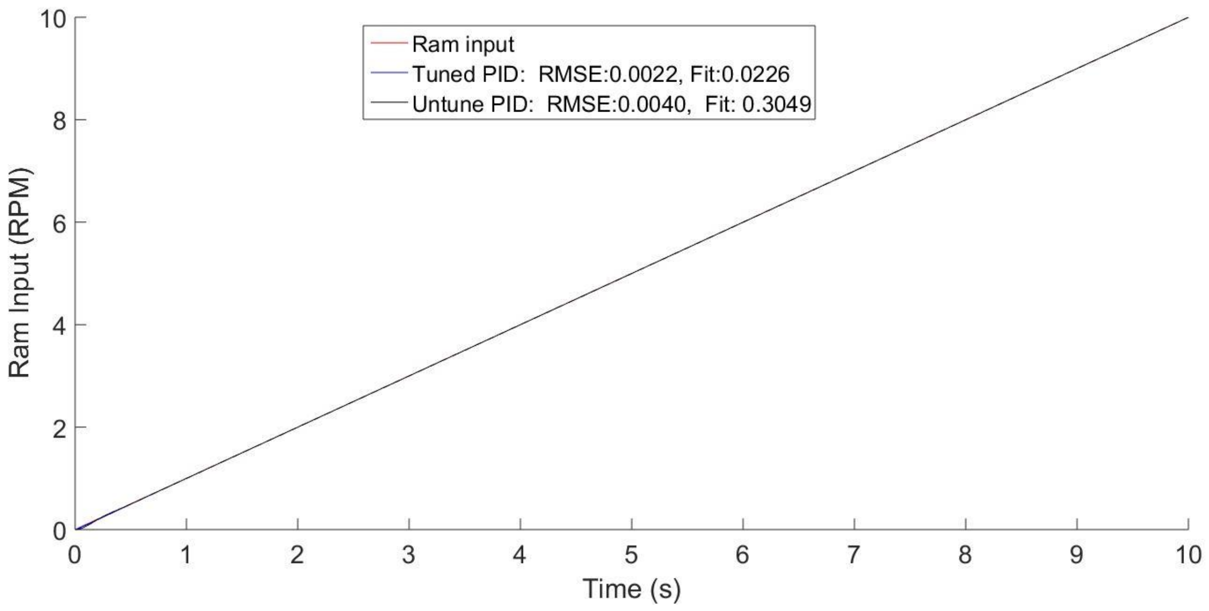


Fig. 5: Ram input command

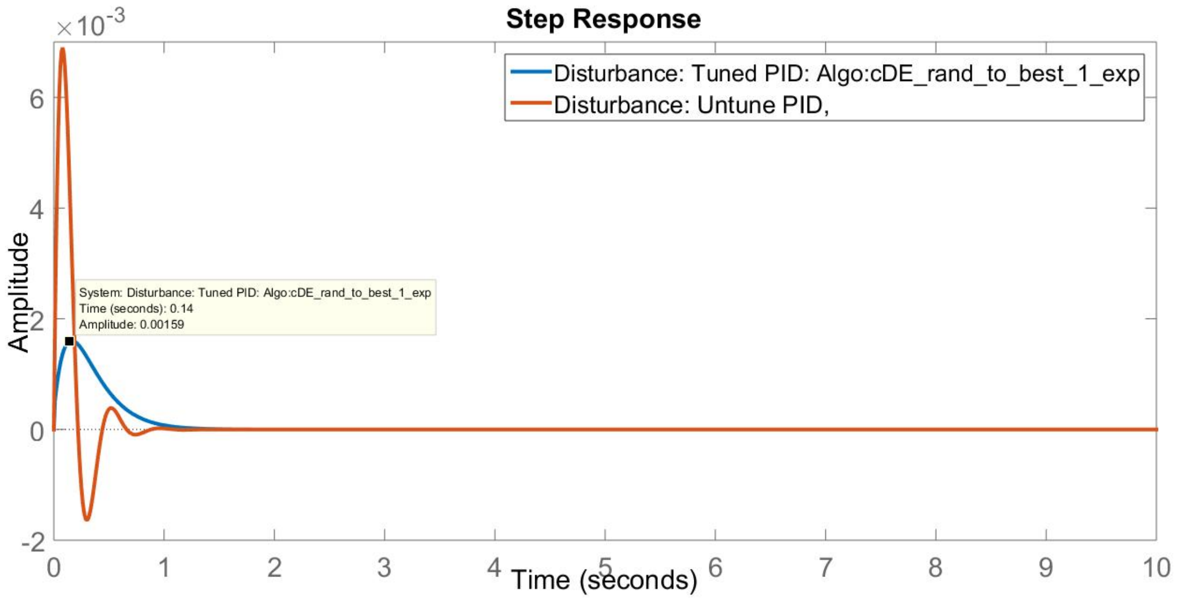


Fig. 6: Disturbance step response

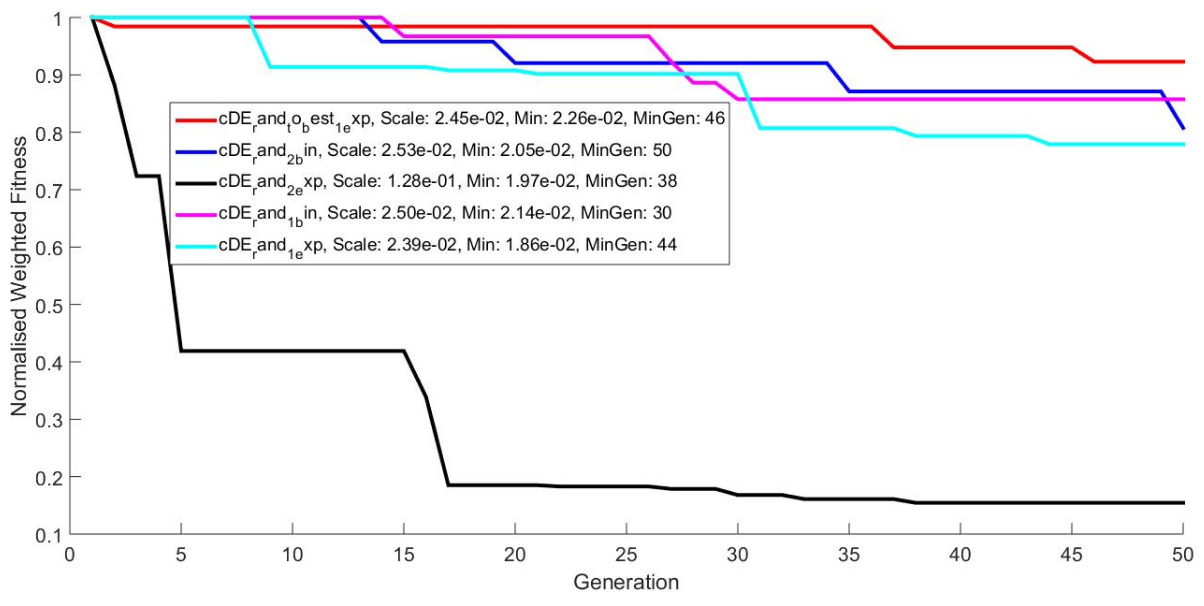
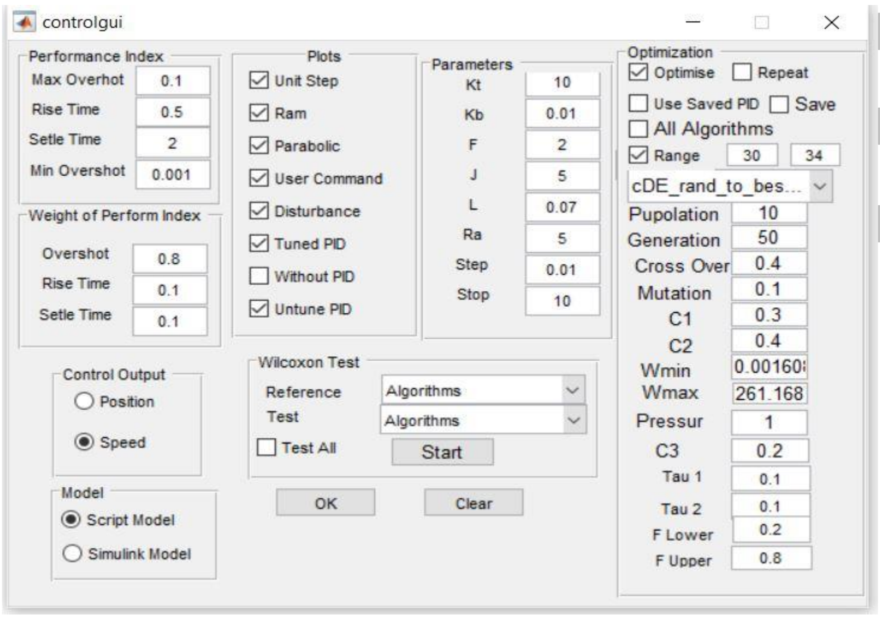


Fig. 7: Generational fitness function of the cDE variants used



controlgui

Performance Index
Max Overhot 0.1
Rise Time 0.5
Setle Time 2
Min Overshot 0.001

Weight of Perform Index
Overshot 0.8
Rise Time 0.1
Setle Time 0.1

Plots
☑ Unit Step
☑ Ram
☑ Parabolic
☑ User Command
☑ Disturbance
☑ Tuned PID
☐ Without PID
☑ Untune PID

Parameters
Kt 10
Kb 0.01
F 2
J 5
L 0.07
Ra 5
Step 0.01
Stop 10

Optimization
☑ Optimise ☐ Repeat
☐ Use Saved PID ☐ Save
☐ All Algorithms
☑ Range 30 34
cDE_rand_to_bes...
Pupolation 10
Generation 50
Cross Over 0.4
Mutation 0.1
C1 0.3
C2 0.4
Wmin 0.00160
Wmax 261.168
Pressur 1
C3 0.2
Tau 1 0.1
Tau 2 0.1
F Lower 0.2
F Upper 0.8

Control Output
○ Position
● Speed

Model
● Script Model
○ Simulink Model

Wilcoxon Test
Reference Algorithms
Test Algorithms
☐ Test All

Fig. 8: Control GUI

Table 1: Performance of the cDE variants implemented for speed control

Algorithms	RMSE for Three Inputs			Max Disturbance Amplitude	Max Overshot	Rise Time	Settling Time	Fitness
	Real world	Ram	Parabolic					
cDE_rand_to_best_1_exp	16.58	0.0022	0.0236	0.00159	0.0145	0.00001	0.11	0.0226
cDE_rand_2_bin	16.45	0.0019	0.0208	0.00136	0.0118	0.00001	0.11	0.0204
cDE_rand_2_exp	16.45	0.0017	0.0179	**0.00133**	0.0134	0.00001	0.09	0.0197
cDE_rand_1_bin	16.53	0.0022	0.0237	0.00152	0.0130	0.00001	0.11	0.0214
cDE_rand_1_exp	16.57	0.0019	0.0207	0.00151	0.0145	0.01	0.06	0.0186

7. Conclusion

The cDE algorithms variants proved to be an efficient optimizer for tuning the PID controller gains with cDE/rand/2/exp algorithm emerging as the best for addressing this particular control problem with maximum disturbance step unit amplitude **0.00133,** followed by cDE/best/1/bin with maximum disturbance step unit amplitude of **0.00136**. The controller implemented in this research depicted a robust performance not only in tracking the command input, but also in mitigating the effect of external disturbance.

References

[1] A.R. Laware1, V.S. Banda and D.B. Talange. Real Time Temperature Control System Using PID Controller and Supervisory Control and Data Acquisition System (SCADA), *International Journal of Application or Innovation in Engineering & Management, Volume 2, Issue 2, 2013*

[2] I J Narath and M. Gopal: Control system Engineering, fourth Edition, 2006

[3] J. J. Liang, A. K. Qin, P. N. Suganthan, and S. Baskar. Comprehensive learning particle swarm optimizer for global optimization of multimodal functions. IEEE Transactions on Evolutionary Computation, 10(3):281–295, 2006.

[4] Katsuhiko Ogata Modern Control Engineering *Fifth Edition* Upper Saddle River 3010

[5] K Smriti Rao, Ravi Mishra Comparative study of P, PI and PID controller for speed control of VSI-fed induction motor, International Journal of Engineering Development and Research, Volume 2, Issue 2, 2014

[6] S. Iliya. Differential Evolution Based PID Antenna Position Control System. *International Journal of Scientific and Engineering Research, July 2017.*

[7] Y. Shi and R. Eberhart. A modified particle swarm optimizer. In Proceedings of the IEEE Congress on Evolutionary Computation, 1998.

[8] S. Iliya. Application of computational Intelligence in Cognitive Radio Network for Effective Spectrum Utilization and Speech Therapy. PhD Thesis, 2017.