CHAPTER 5

CONTROLLER DESIGN FOR DFIG BY USING FIREFLY ALGORITHM TECHNIQUE

5.1 Introduction

The wind energy has proven to be a potentially clean, free and renewable source for generation of electricity with minimal environmental impact. Renewable energy sources are the type of energy sources which are plenty in quantity and are derived from the earth. Wind energy, solar energy, geothermal and biomass are different types of renewable energy sources. These resources are inexhaustible [1]. However, the consistency capacity and financial evaluation of offshore wind farm along with well-known recompense of renewable energy sources are its clean nature and eco-friendly, unlike non-renewable energy sources. Nowadays more research is going on the enhancement of technology which can efficiently convert the renewable energy sources into useful electrical energy sources. The technology is developed and improvised on a daily basis to improve its efficiency above 90%. The Wind power capacity is growing at the rate of 20% annually on the average in the world, but its cost has decreased 50% in the last ten years [140]. On the other hand, the conventional sources of energy are causative towards pollution. For the reason that of several disadvantages, renewable energy sources can be used as alternatives to it. Among different types of renewable sources of energy, wind energy is the cleanest and the efficient cause of power [1]. The significant advantages of wind energy are wind-generated electricity doesn't pollute the water, air or soil. It doesn't contribute towards global warming. It doesn't consume a significant amount of water required by other energy sources. All day solar radiation causes it. Its supply is abundant, unlike solar power during adverse weather condition and night time. The price of electricity generation by wind power plant is comparatively lesser than other modes of production. It contributes towards the economy of middle class and low-class communities. It also creates employment opportunities for highly skilled workers. It's very fast and easy to install. In a year many sizeable utility-scale wind power plants are established. A wind turbine uses the DFIG generator, and we use PID controller, to optimize the control parameter of controller we use the evolutionary algorithms [141].

The establishment of this chapter is as follows. The basic theory related to non-conventional sources and wind energy conversion system is described in section-1. The types of generator used in wind energy for WECs have been explained in section-2. A technological overview as well as power flow with Mathematical Model of DFIG system as a transfer function is illustrated in section-3. In section-4&5, the evolutionary technique descriptions with their advantages along with the firefly algorithms with fitness function for the design of a PID controller using FFA-method are described. However, the Differential evolution (DE) technique along with algorithm has been explained for controller gains in section-6. The simulation response to DFIG system is described in section-7. The experimental comparison between FFA and DE-based PID controller has been presented successfully in this section. Finally, the conclusion and brief discussions on research work are illustrated in the last section-8.

5.2 Technological outline of Doubly Fed Induction Generator for Wind Turbine

The method shown below is used for variable speed operation about thirty percent of synchronous speed power electronic converter controls 20-30% of the whole power flowing through the network. The power factor of the overall system will be one [142]. PWM Pulse generators control the converter. Figure (5.1) shows the symbolic representation of Doubly-Fed Induction Generator.

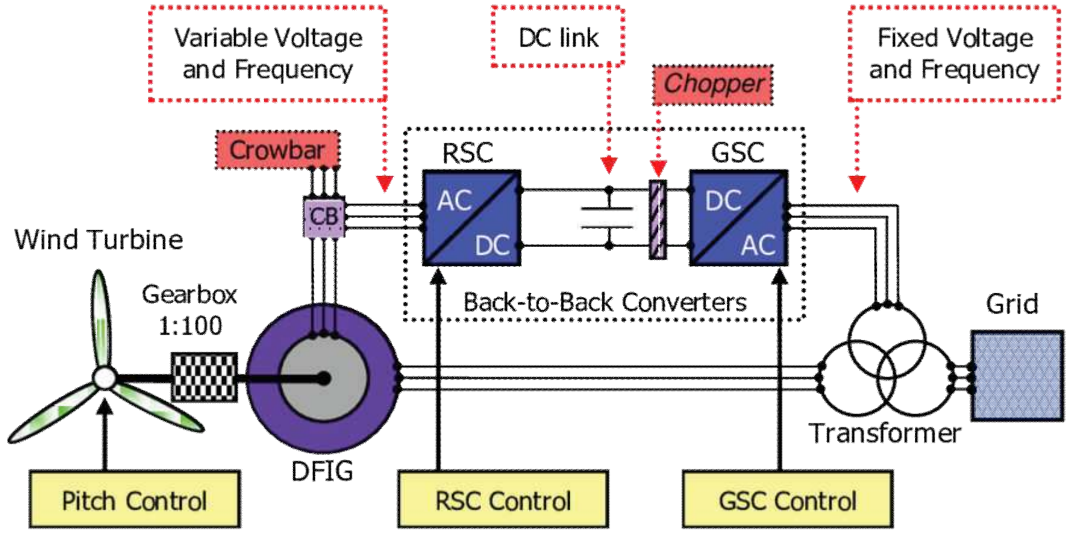


Figure 5.1: Symbolic representation of Doubly-Fed Induction Generator

The converters are linked in back to back manner through a transformer to the grid. Between two converters, a dc capacitor is associated, to keep the voltage variances in the link voltage to a minimum quantity[143].

5.3 Power flow in the DFIG system

To compute power flow of the DFIG scheme the apparent power is fed to the DFIG through the stator. And rotor circuit must be resolved. DFIG can run in two methods for operation, for example, a sub-synchronous and super-synchronous mode given the rotor speed under or more the synchronous speed[22]. The flow of power in the rotor of a doubly fed induction machine has three parts. These are

- The electromagnetic power trade between the stator and the rotor throughout the air gap and can be characterized as the air gap power Pg.
- The mechanical force Pm traded between the rotor and shaft.

➢ P_r (slip power) exchange between the rotor and load through the rotor slip rings. These three segments of rotor forces are secondary, in sub- and super-synchronous methods of operation, as indicated in Figure (5.2&5.3).

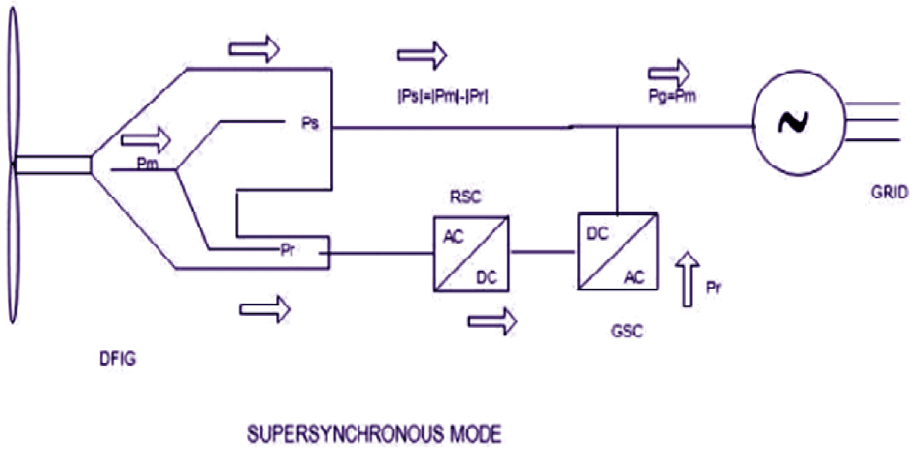


Figure 5.2: Power flow in DFIG during Super-synchronous speed

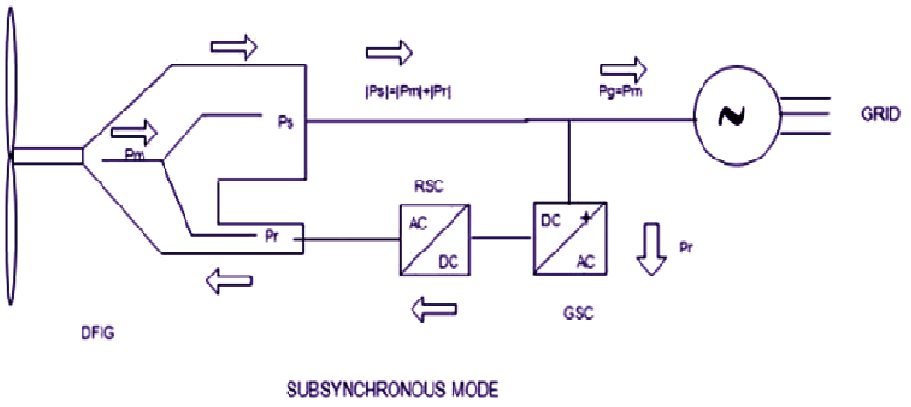


Figure 5.3: Power flow in DFIG during Sub-Synchronous speed

On the other hand the Numerical form of DFIG system in this thesis, as a transfer function of 6th order model is given [46] as follows;

$$G(s) = \frac{0.000324\, s^6 - 1.75\, s^5 - 2366\, s^4 + 7.9e06\, s^3 + 7.5e09\, s^2 + 5e12\, s + 2.18e14}{S^6 + 2340\, s^5 + 8.67e06\, s^4 + 4.79e09\, s^3 + 2.7e12\, s^2 + 1.27e14\, s + 9.6e14} \quad (1)$$

The transfer function of the taken as a whole system is obtained either via simple modeling or using the numerical differentiation [44].

5.4 Description of Evolutionary Techniques

Intelligence is the capability to obtain, understand and apply knowledge or the ability to implement thought and reason. It embodies all the knowledge both aware and unconscious, which we learn during learn and understanding, highly refined sight and sound perception, thought, imagination, ability to converse, read, write and recall facts, express and feel emotions, and much more[141]. Artificial Intelligence deals with the study and creation of computer systems that exhibit some form of intelligence systems. The motivation of AI skill is to make computers behave more like humans in solving problems. It is different from general programming. On the other hand, soft computing is a tool of artificial intelligence which differ from hard computing, In effect, the role model of soft computing is the human mind. The work is based on exploiting the two efficient intelligence based evolutionary soft computational technique, Firefly Algorithm (FFA) and Differential Evolution (DE) algorithm to design a PID controller for a low damping plant.

5.4.1 The advantage of Evolutionary techniques [144]

1. Evolutionary methods work with a coding for the parameter set, not the parameters themselves. The solution doesn`t depend on the parameter irrespective to the unit of the variable, for example, the parameter will be converted to the binary string, and all operation will be performed on that solution string irrespective of nature of solution and termination of iteration it turns back a binary answer to the original unit.

2. Evolutionary technique search from a population of points, not a single point. So to search solution in different dimension simultaneously, by doing so to reach the answer in a fast way and also discard the unimproved solutions, gradually our solution is going converging at a point (global solution).

3. Evolutionary techniques use objective function information, not derivatives or other auxiliary knowledge. So primary focus is the primary function, and other features are not needed, but sometimes there are some restrictions for a solution called parametric Constraints which have to keep in mind.

4. Evolutionary techniques use probabilistic transition rules, not deterministic rules so that the solution will consider all possibilities.

5. Evolutionary methods are robust concerning local minima, maxima. So the solution will not stick into the local minima or maxima, and it will search for global maxima or minimal if the solution stuck into the local maxima or minima then by doing crossover and mutation solution get overcome from the local minima and maxima.

6. Evolutionary techniques work well on continuous and on discrete or mixed discrete/continuous problems.

7. Evolutionary methods can operate on various representations and various dimensions also.

5.4.2 Firefly Algorithm (FFA)

The firefly algorithm is a meta-heuristic proposed by Xin-She Yang and inspired by the flashing behavior of fireflies. Firefly Algorithm is a nature-inspired algorithm, which is based on the flashing light of fireflies. In fact, the algorithm has three particular idealized rules which are found in reality on some significant flashing characteristics of real fireflies [145].

These are subsequent:

1. All fireflies are unisex and will move towards more attractive and brighter ones regardless their sex.
2. The degree of attraction of a firefly is proportional to its brightness which decreases as the distances from the other fireflies' increases.
3. If there is more attractive firefly then a particular one, then it will move randomly.

For an optimization problem, the flashing light is associated with the fitness function to obtain efficient optimal solutions [145].

In this algorithm, when searching for solutions, the fireflies use two primary procedures: attractiveness and movement, which are defined as follows:

Attractiveness

The form of the attractiveness function of a firefly is the following monotonically decreasing function:

$$\beta(r) = \beta_0 \exp(-\gamma r^m), \text{ With } m \geq 1$$

here r is the distance between any two fireflies, β_0 is the initial attractiveness at $r = 0$ and γ is the absorption parameter which controls the decrease of the light intensity.

The distance r between any two fireflies i and j, at position x_i and x_j, correspondingly, can be determined as a Cartesian or Euclidean as follows:

$$r_{ij} = \sqrt{\sum_{k=1}^{d} \left(x_{i,k} - x_{j,k} \right)^2}$$

Where, x_i, k, is the k_{th} component of the spatial coordinate x_i of the i_{th} firefly and d is the dimension number.

Movement

The following equation gives the movement of a firefly i is attracted by a brighter firefly j:

$$x_i = x_i + \beta_o * \exp(-\gamma r_{ij}^2) * \left(x_j - x_i\right) + \alpha * \left(rand - \frac{1}{2}\right)$$

Here the primary term is the current position of a firefly, and the second time is used for taking into consideration a firefly's attractiveness to light intensity seen by adjacent fireflies. Moreover, the third term is used for the random movement of a firefly [145]. The coefficient α is a randomisation parameter and resolute by the difficulty of interest, whereas *rand* is a random numeral generator uniformly distributed in the space [0, 1].

Flowchart of Firefly Algorithm

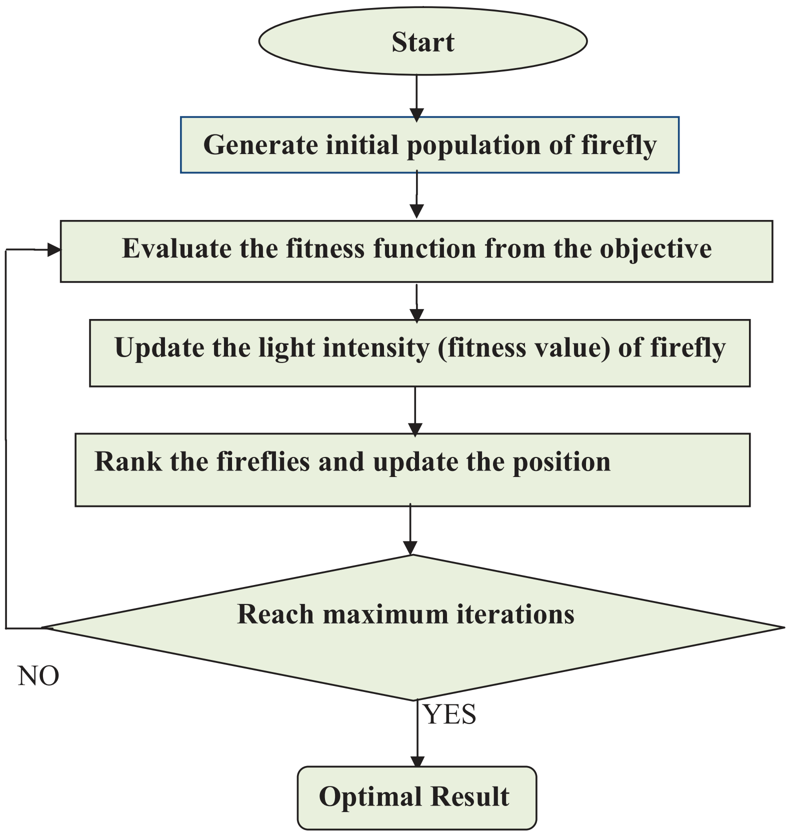

Start
Generate initial population of firefly
Evaluate the fitness function from the objective
Update the light intensity (fitness value) of firefly
Rank the fireflies and update the position
Reach maximum iterations
NO
YES
Optimal Result

Figure 5.4: Flow chart of firefly algorithm

Firefly Algorithm for controller design

```
Begin;
Initialize algorithm parameters;
To define the objective function of f(x), where $x=(x1,........, xd)^T$
To generate initial population of fireflies or $x_i$ (i=1, 2,..., n)
Determine the light intensity of $I_i$ at $x_i$ via $f(x_i)$
While (t<MaxGen)
For i = 1 to n ,(all n fireflies);
For j=1 to n, (n fireflies)
if($I_j$> $I_i$),  move firefly i towards of j;
end if
Attractiveness varies with distance r via $Exp[-\gamma r^2]$;
Evaluate new solutions and update light intensity;
End for j;
End  for  i;
Rank the fireflies and find the current best;
End while;
Post process results and visualization;
End procedure
```

5.4.3 The concept of Fitness Function for the Design

For our case of design, we have to tune all the three parameters of PID such that it gives the best output results. Here we define a three-dimensional search space in which all the three dimensions represent three different parameters of the PID. Each particular point in the search space represents a specific combination of [K_P, K_I, K_D] for which a specific response is obtained. The performance of the position or combination of PID parameters is decided by a fitness function or the cost function. This fitness function consists of several component functions which are the performance index of the design. The point in the search space is the best point for which the fitness function attains an optimal value [146]. But in such case, four component functions to define Fitness function. The fitness function is a function of steady-state error, peak overshoot, rise time as well as settling time. On the other hand, the involvement of these component functions towards the original fitness function is determined by a scale factor that depends upon the choice of the designer. For this design, the best point is the point where the fitness function has the minimal value.

The chosen fitness function:

$F = (1-\exp(-\beta))\,(M_P + E_{SS}) + (\exp(-\beta))(T_S - T_r)$

Here;

F:-Fitness function

M_P: - Peak Overshoot

T_S: - Settling Time

T_r: - Rise Time

β:-Scaling Factor(Depends upon the choice of the architect)

For such case design scaling factor $\beta = 1$.

In the Matlab library, it has to define a fitness function that is PID parameters as input values, and it returns the fitness value of the PID based controlled model as its output. It has the format:-

Function [F] = fitness (K_D, K_P, K_I)

This fitness function will take the PID parameters as input and return calculated value of fitness function for different cases. We need to minimize it as possible by providing different values of PID parameters.

5.4.4 Problem Statement

The thesis is objected to design a PID controller for a low damping plant. The low damping plants are the higher order plants which exhibit sluggish behavior. It means that the plant has substantial settling time, large peak overshoot which is undesirable for better performance.We have the 6th order transfer function model of the DFIG plant in [44,46], as follows.

$$\mathbf{G(s)} = \frac{0.000324\,s^6 - 1.75\,s^5 - 2366\,s^4 + 7.9e06\,s^3 + 7.5e09\,s^2 + 5e12\,s + 2.18e14}{s^6 + 2340\,s^5 + 8.67e06\,s^4 + 4.79e09\,s^3 + 2.7e12\,s^2 + 1.27e14\,s + 9.6e14}$$

The fitness function is:-

$F = (1-\exp(-\beta))\,(M_P + E_{ss}) + (\exp(-\beta))\,(T_s - T_r)$

Where;

M_P: Peak Overshoot

T_s: Settling Time

T_r: Rise Time

β: Scaling Factor(Depends upon the choice of the designer)

We make a function that will take value from the output of evolutionary algorithm techniques as (K_P, K_I, K_D), and give us the evaluated value of the fitness function. In this function, we make a PID controller that will use these parameters, and then we take the closed loop performance parameters like Peak Overshoot, Settling Time, Rise Time, Scaling factor and put these parameters into the fitness function equation [147].

Function f=myfitnessfcn(x)

PID parameters taken from Evolutionary algorithm

K_P = x (1);

K_i = x (2);

K_d = x (3);

Scale factor (in our case we taken it 0.5 moderate)

b=0.5;

Closed loop transfer function

s=tf ('s');

g=tf ([.000324 -1.75 -2366 7.9e6 7.5e9 5e12 2.18e14], [1 2340 8.67e6 4.79e9 2.7e12 1.27e14 9.6e14]);

c=pid (kp, ki, 0);

t=feedback(c*g, 1);

Performance parameters of closed loop transfer function

q = step info (TF);

tr = q.RiseTime;

ts=q.SettlingTime;

m_p=q.Overshoot;

ess =1/ (1+dcgain (TF));

Evaluation of Fitness function

f= (1-exp (-b))*(m_p+e_{ss}) +exp (-b)*(ts-tr);

5.4.5 Confining the search space for FFA

We limit our search space to get a better solution and reduce the time taken by redundant iterations; we run our algorithm many numbers of times within the defined search space to obtain an optimal solution (K_p, K_i, K_d). We use two evolutionary techniques in this chapter with traditionally used parameters values and observe that different method have different results and time used. Before actually running the program we do some trial and error practices for confining the search space, and we find that by using the secondary controller our closed loop system become unstable, so in this chapter, we use only a PI controller. We also find that the range of K_P will be $0<K_P< 20$ and that the scope of K_i will be $0<K_i< 200$ are good, beyond which our response is not practical and also closed loop system get unstable [144]. However, there will be further improvements by changing the Number of iteration, different constant used and size of the population.

FFA based controller parameter is as shown in Table (5.1).

Table 5.1: Gains of the FFA-based controller

Parameters	kp	ki	Kd
Gains	15	140.9225	0

5.5 Differential Evolution Algorithm

In the evolutionary computation, differential evolution (DE) is a technique that optimizes a problem by iteratively trying to improve a candidate solution about a given measure of quality. Such methods are commonly known as meta-heuristics as they make few or no assumptions about the problem being optimized and can search vast spaces of candidate solutions. However, meta-heuristics such as DE do not guarantee an optimal solution is ever found [148]. Whereas the DE is worn for multidimensional real-valued functions but does not apply to the gradient of the problem that is being optimized, that means DE does not require for the optimization problem to be differentiable as is needed by standard optimization techniques such as gradient descent and quasi-newton methods. DE can be used to optimization problems that are not even continuous, are noisy, change over time, etc. DE optimizes a problem through maintaining a population of candidate solution along with to create novel candidate solutions with active join ones. By easy formulae and keeping candidate solution to the best score or fitness on the optimization problem. Hence the optimization problem is treated as a black box that merely provides a measured quality for given candidate solution along with the gradient is therefore not needed [148].

5.5.1 Flowchart for differential evolution algorithm

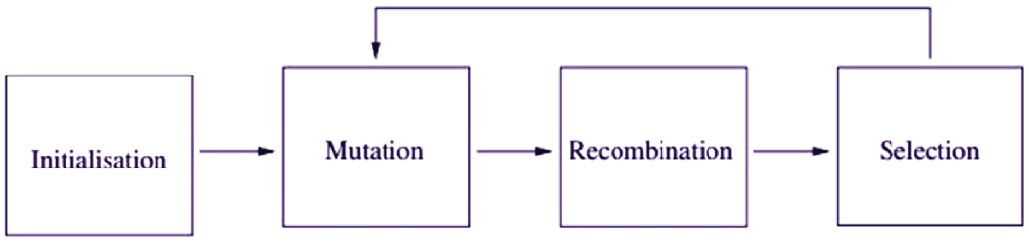


Figure 5.5: Flowchart of DE

The primary variant of the DE algorithm works by having a population of candidate solutions (called agents).These agents have moved around in the search-space by using simple mathematical formulae to combine the positions of existing agents from the population. If the new location of an agent is an improvement, it is accepted and forms part of the population. Otherwise, the new location is just redundant. On the other hand, the process is repeated and by doing, so it is hoped, but not guaranteed that an acceptable solution will eventually be discovered. Formally, let Function f (fitness function that must be optimised) be the cost function which must be minimized. Then the function takes a candidate solution as an argument in the form of a vector of real numbers and produces an actual number as output. which designates to the fitness of specified candidate solution. The gradient of *f* is not recognized. The objective is to find a solution *m* for which $f(m)<f(p)$ for all *p* in the search-space, that would mean *m* is the global minimum. Maximization can be performed by considering the function $g=-f$ instead. Let $x \epsilon Rn$ assign a candidate solution (agent) in the population. $C_R \epsilon$ [0, 1] is called the ***crossover probability***. Let $F \epsilon$ [0, 2] be called the ***differential weight***. The practitioner chooses both these parameters along with the population size less than 4.

The first DE algorithm can then be described as follows:

- To Initialize all agents X with random positions in the search-space.
- Until a termination criterion meet (e.g.number of iterations performed) replicate as follows:
    - For each agent X in the population do:
        - To choose three agents **a**, **b** and **c** from the population at random that have to be distinctive from each other as well as from agent X.
        - To select a random index $R \epsilon \{1,2\ldots..,n\}$(*n* is dimensionality problem that is to be optimized).
        - To compute the agent's potentially new position Y=[y1,….yn] as follows:
            - For each $\mathbf{i} \epsilon \{1,2\ldots,n\}$,pick consistently dispersed number $\mathbf{r_i=U(0,1)}$
            - If $\mathbf{r_i<CR}$ or $\mathbf{i = R}$ then set $\mathbf{Y_i= a_i + F \times (b_i - C_i)}$ otherwise set $\mathbf{Y_i = X_i}$
            - Then (novel position is outcome to binary crossover of agent X with the intermediate agent $\mathbf{z= a+F\times (b-c)}$).

- If $f(y)<f(x)$ then substitute the agent in the population with the improved candidate solution, that is, return X with Y in the population.
- Pick the agent from the population that has the highest fitness or lowest cost and return it as the best-found candidate solution.

5.5.2 Termination

Termination of the algorithm ideally takes place after the global optimum is achieved, but this may not always be the case. Frequently, completion of the algorithm is a user-defined input, and the user can limit the number of iterations of the algorithm. It is a trial-and-error approach, in that a sufficient number of iterations are required to ensure the best-known results are returned. Another method for termination is when the objective has been met. In some objective functions, the optimal value can already be known. For example, some features such as benchmark functions may have a known minimum terminate. Additionally, feedback provided by the objective function can determine that no further optimization is possible. For example, if the optimization stalls and thus many following objective function values are the same, the algorithm may be terminated. Also, personal monitoring can determine when optimization is over [144].

5.5.3 Variants

Variants of the DE algorithm are continually being developed to advance optimization performance. Many different schemes to performing crossover and mutation of agents are possible in the underlying algorithm given above. Supplementary advanced DE variants are also being developed with a popular research trend being to perturb or adapt the DE parameters during optimization; There is also various work in construction a mixture optimization technique using DE combined with other optimizers.

5.5.4 Setting Control Parameters

The values of population size (N_P), crossover constant (C_R) and weighing factor or mutation scale factor (F) are fixed empirically, following specific heuristics. Appropriate tuning of these parameters is essential for the reliable performance of the algorithm. Trying to tune these three main control parameters and finding bounds for their values has been a topic of intensive research. Mutation scale factor F controls the speed and robustness of the search. A lower cost for F gains the convergence rate, although it does so at the risk of getting stuck into a local optimum along with failing to find the real global solution. Parameters C_R and N_P have a similar effect on the convergence rate as F. High values of C_R favor a higher mutated element crossover to current elements; as a result, the mutation factor F has a more significant impact on the search. As well, an increased value of N_P increases the diversity of the population and with it the potential to find the exact optimal solution from the more excellent search space but at the cost of longer computation time. The control parameter selection is a difficult task due to their interdependence with each other and the fact that some objective functions are sensitive to proper settings (Liu and Lampinen, 2002). Traditionally, the control parameters have been held fixed during the whole execution of the algorithm. The **rule-of-thumb** values to the control parameters given by Storn and Price (1997) for F is generally between 0.5 and 1.0 and C_R between 0.8 and 1.0. These authors have proposed that the population size N_P should be between $5\times D$ and $10\times D$ and not less than 4 to ensure that the mutation operation can be carried out. If miss-convergence occurs, N_P should be increased; however, beyond an absolute limit, it is not used to improve the population size anymore. The suggestions by Storn and Price to the control parameters are applicable for many practical purposes but still lack generality. It means that, in practice, many time-consuming trial runs are required to find optimal parameters for each problem setting [146].

5.5.5 The concept of Fitness Function for the Design

For our case of design, we have to tune all the three parameters of PID such that it gives the best output results. Here we define a three-dimensional search space in which all the three dimensions represent three different parameters of the PID. Each particular point in the search space represents a specific combination of [K_P K_I K_D] for which a specific response is obtained. The performance of the position or the combination of PID parameters is determined by a fitness function or the cost function. This fitness function consists of several component functions which are the performance index of the design. The point in the search space is the best point for which the fitness function attains an optimal value [146]. For such case of design, it has been taken four component functions to define Fitness function. The fitness function is a function of steady-state error, peak overshoot, rise time and settling time. On the other hand, the contribution of these component functions towards the original fitness function is determined by a scale factor that depends upon the choice of the designer. For this design, the best point is the point where the fitness function has the minimal value.

The chosen fitness function is given as follows: $F = (1-\exp(-\beta))\,(M_P + E_{SS}) + (\exp(-\beta))(T_S - T_r)$

Here; F:-Fitness function

M_P: - Peak Overshoot

T_S: - Settling Time

T_r: - Rise Time

β:-Scaling Factor(Depends upon the choice of the designer)

For our case of design, we have taken the scaling factor $\beta = 1$.

In the Matlab library, we have defined a fitness function which has PID parameters as input values, and it returns the fitness value of the PID based controlled model as its output. It has the format:-

Function [F] = fitness (K_D, K_P, K_I)

This fitness function will take the PID parameters as input and return calculated value of fitness function for different cases. We need to minimize it as possible by providing different values of PID parameters.

5.5.6 Problem Statement

The thesis is objected to design a PID controller for a low damping plant. The low damping plants are the higher order plants which exhibit sluggish behavior. It means that the plant has considerable settling time, large peak overshoot which is undesirable for better performance. We have the 6th order transfer function model of the DFIG plant in [46], as follows.

$$\mathbf{G(s)} = \frac{\mathbf{0.000324\ s^6 - 1.75\ s^5 - 2366\ s^4 + 7.9e06\ s^3 + 7.5e09\ s^2 + 5e12\ s + 2.18e14}}{\mathbf{S^6 + 2340\ s^5 + 8.67e06\ s^4 + 4.79e09\ s^3 + 2.7e12\ s^2 + 1.27e14\ s + 9.6e14}}$$

The fitness function is:-

$F = (1-\exp(-\beta))(M_P + E_{ss}) + (\exp(-\beta))(T_s - T_r)$

Where;

M_P: Peak Overshoot

T_s: Settling Time

T_r: Rise Time

β: Scaling Factor (Depends upon the choice of the designer)

We make a function that will take value from the output of evolutionary algorithm techniques as ($K_{P,}$ K_I, K_D), and give us the evaluated value of the fitness function. In this function, we make a PID controller that will use these parameters, and then we take the closed loop performance parameters like Peak Overshoot, Settling Time, Rise Time, Scaling factor and put these parameters into the fitness function equation [147].

Function f=my fitness fcn(x)

PID parameters taken from Evolutionary algorithm

K_P = x (1);

K_i = x (2);

K_d = x (3);

Scale factor (in our case we taken it 0.5 moderate)

b=0.5;

Closed loop transfer function

s=tf ('s');

g=tf ([.000324 -1.75 -2366 7.9e6 7.5e9 5e12 2.18e14], [1 2340 8.67e6 4.79e9 2.7e12 1.27e14 9.6e14]);

c=pid (kp, ki, 0);

t=feedback(c*g, 1);

Performance parameters of closed loop transfer function

q = step info (TF);

tr = q.RiseTime;

ts=q.SettlingTime;

m_p=q.Overshoot;

ess =1/ (1+dcgain (TF));

Evaluation of Fitness function

f= (1-exp (-b))*(m_p+e_{ss}) +exp (-b)*(ts-tr);

5.5.7 Confining the search space for DE

We enclose our search space to get a better solution and reduce the time taken by redundant iterations; we run our algorithm many numbers of times within the defined search space to obtain an optimal solution (K_p, K_i, K_d). We use two evolutionary techniques in this chapter with traditionally used parameters values and observe that different method have different results and time used. Before actually running the program we do some trial and error practices for confining the search space, and we find that by using the derivative controller our closed loop system become unstable, so in this chapter, we use only a PI controller. We also find that the range of K_P will be $0<K_P< 20$ and that the scope of K_i will be $0<K_i< 200$ are good, beyond which our response is not practical and also closed loop system get unstable [147]. However, there will be further improvements by changing the Number of iteration, different constant used and size of the population.

For obtain the gains of PID controller parameters by DE algorithm, buy using the section 5.4.5, 5.4.6, 5.4.7 as shown in Table (5.2).

Table 5.2: Gains of the DE-based controller

Parameters	K_P	K_i	K_d
Gains	14.7269	137.9634	0

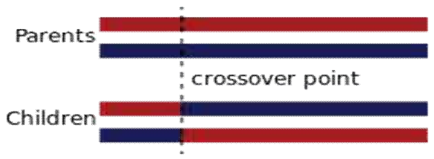
5.6 Genetic Algorithm

The genetic algorithm (GA) was developed using the principles of evolution, natural selection, and genetics from natural biological systems. For the duration of GA computing, a population of artificial individuals is modified repeatedly based on biological evolution rules that converge towards a better solution of the problem to be solved. At each step, individuals are selected randomly from the present population to be parents. These individuals are used to generate children for the next generation. Based on biological principles, the fittest individuals survive and the least appropriate die. The GA consists of three primary procedures: selection, crossover, and mutation. It starts with an initial population containing some chromosomes each of which represents a solution to the problem. However, some chromosomes are not fit, and these will be ignored while only the fittest chromosomes will be designated for reproduction. To keep the size of the first population stable some suitable chromosomes will be used more than once, so this subset is called the ‘matting pool.' The purpose of this selection operation is to obtain a mating pool with the fittest individuals according to a probabilistic rule that allows the most appropriate individuals to be mated into the new population. After the selection stage, a genetic crossover operation is then applied between parent pairs from the mating pool. Each pair will produce two children to maintain the size of the population [149]. The crossover operation forces the new chromosomes to have the properties and characteristics of both parents. In this, the same part of both parents participates and produces a new child who has both the features of parents.

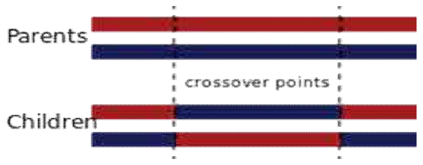
There are many types of crossover listed below.

5.6.1 Types of crossover

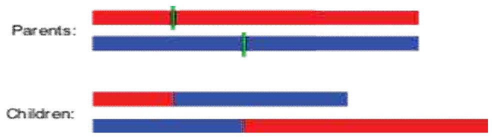
1. One Point Crossover:-



2. Two Point Crossover:-

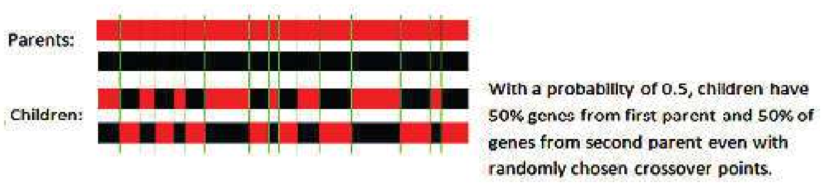


3. Cut and splice Crossover:-



In this type of crossover, the length of the children can be different.

4. Uniform crossover:-


Parents:
Children:
With a probability of 0.5, children have 50% genes from first parent and 50% of genes from second parent even with randomly chosen crossover points.

5. Three Parent Crossover:-

P_1: 110100010

P_2: 011001001

P_3: 110110101

$C_{p1p2p3:}$110100001

Figure 5.6: Types of crossover in GA

Where the P1 and P2 are same, the Child takes that digit, and where the P1 and P2 are not equal, there will be a P3 digit. Mutation is the final operation which takes place after crossover and introduces a change into the offspring bit string to produce new chromosomes. Beside this, it may represent a solution to the problem while avoiding the population falling to a local optimal point. The mutation operation is performed with a probability of mutation which is usually low. It preserves right chromosomes and mimics real life where mutations rarely happen.

Bit String Mutation: - This mutation will depend upon the mutation probability.

$$\begin{array}{ccccccc} 1 & 0 & 1 & 0 & 0 & 1 & 0 \\ & & & & \downarrow & & \\ 1 & 0 & 1 & 0 & 1 & 1 & 0 \end{array}$$

Figure 5.7: Bitwise mutation

The new individuals created by these three basic operations may be fitter than their parents. This algorithm is repeated for many generations and finally, stops when individuals are produced who provide an optimal solution to the problem or this generational process is repeated until a termination condition has been reached. Common terminating conditions are [149]:

- A solution is originated that satisfies minimum criterion
- Preset number of generations reached
- To allocated budget (computation time/money) reached
- The highest ranking solution's fitness is reaching such that successive iterations no longer produce better results
- Combinations of the above

5.6.2 Flowchart of GA optimization

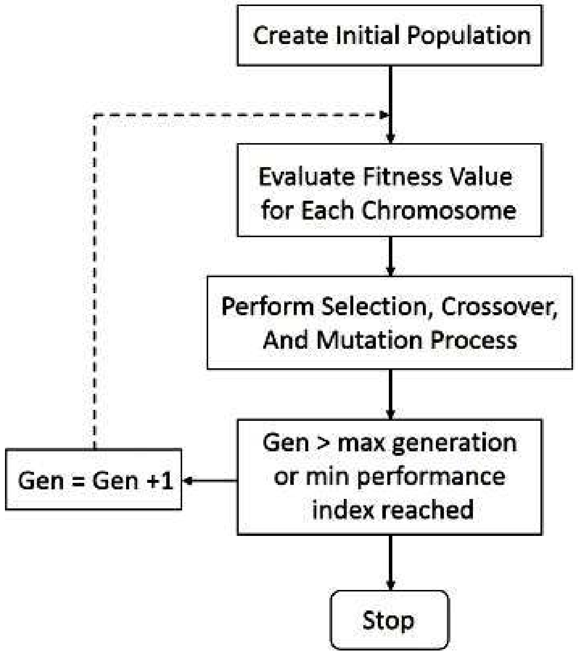

Create Initial Population
Evaluate Fitness Value for Each Chromosome
Perform Selection, Crossover, And Mutation Process
Gen > max generation or min performance index reached
Gen = Gen +1
Stop

Figure 5.8: Flowchart of GA

For obtain the gains of PID controller parameters by GA algorithm, repeat the section 5.4.5, 5.4.6, 5.4.7 once again.

Table 5.3: Gains of the GA-based controller

Parameters	K_P	K_i	K_d
Gains	10.7270	102.4343	0

5.7 Simulation Results

5.7.1 Simulink response of DFIG system

Here Wind Farm - DFIG Average Model (MATLAB 13b) shows a 9 MW wind farm with the DFIG driven by a variable speed wind turbine [147] as shown in Figure (5.9). It consisted of six 1.5 MW wind turbines and linked toward a 25 kV distribution system exports power to a 120 kV grid throughout a 30 km, 25 kV feeder. In this illustration, the wind velocity is preserved invariable at 15 m/s. The control system uses a torque controller to retain the rate at 1.2 Pu. The reactive power produced by the wind turbine is regulated at 0 Mvar. Intended for a wind velocity of 15 m/s, the turbine production power is equal to 1 Pu of rated power, the pitch angle is 8.7 deg, and the generator speed is 1.2 Pu. in the beginning, the DFIG wind farm produces 9 MW. The corresponding turbine speed is 1.2 Pu of synchronous generator speed. The DC voltage is harmonized at 1150 V, and also reactive power is reserved at 0 Mvar. At the t=0.03 s the positive-sequence voltage rapidly drops to 0.5 Pu. Causing an oscillation on the DC bus voltage and DFIG output power during a voltage sag, the control system attempts to standardize DC voltage along with reactive power at their set points (1150 V, 0 Mvar).The System recuperates in nearly four cycles.

The response of the DFIG based wind turbine with FFA-based controller gain and DE&GA-based controller regarding voltage along with current at the terminals, active power generated; reactive power requirements and DC capacitor voltage, as well as generator speed, are shown in Figure (5.10 to 5.27) respectively.

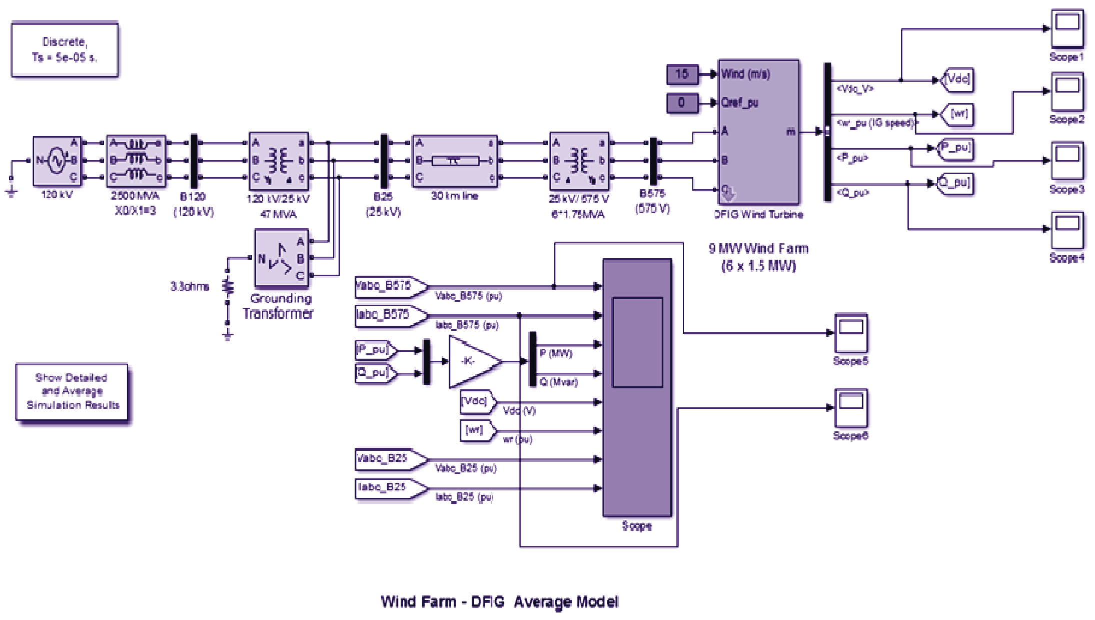


Figure 5.9: DFIG based wind turbine Average Matlab Simulink Model diagram

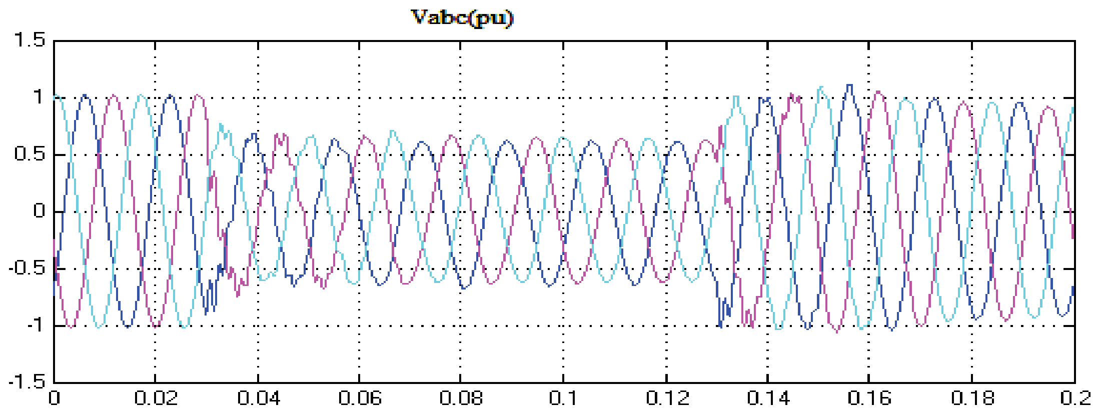


Figure 5.10: Voltages at the DFIG terminals in Pu (GA-based controller)

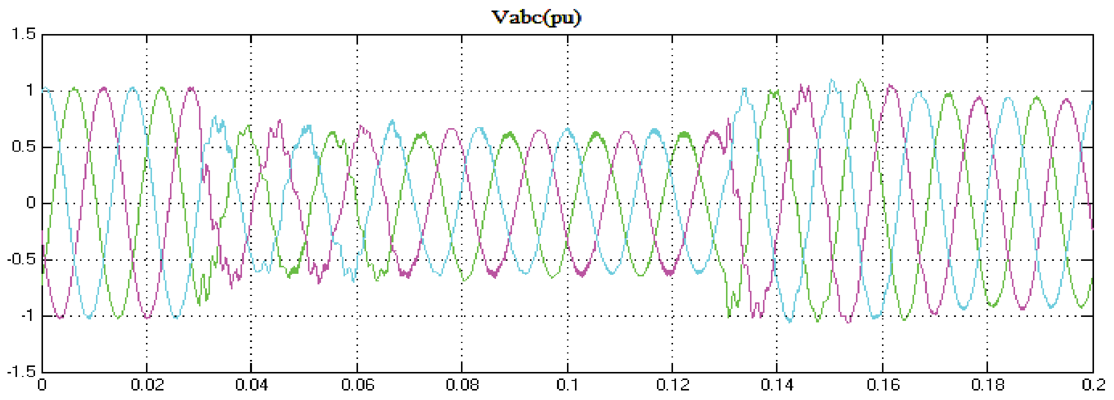


Figure5.11: Voltages at the DFIG terminals in Pu (DE-based controller)

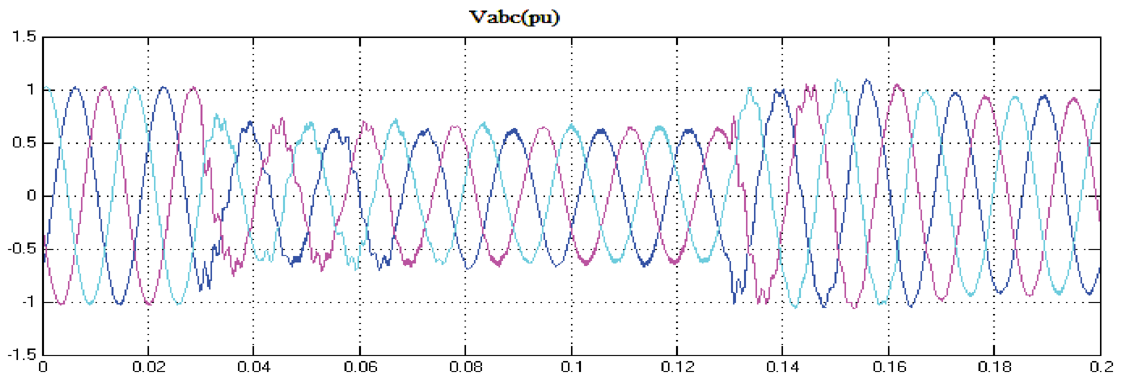


Figure 5.12: Voltages at the DFIG terminals in Pu (FFA-based controller)

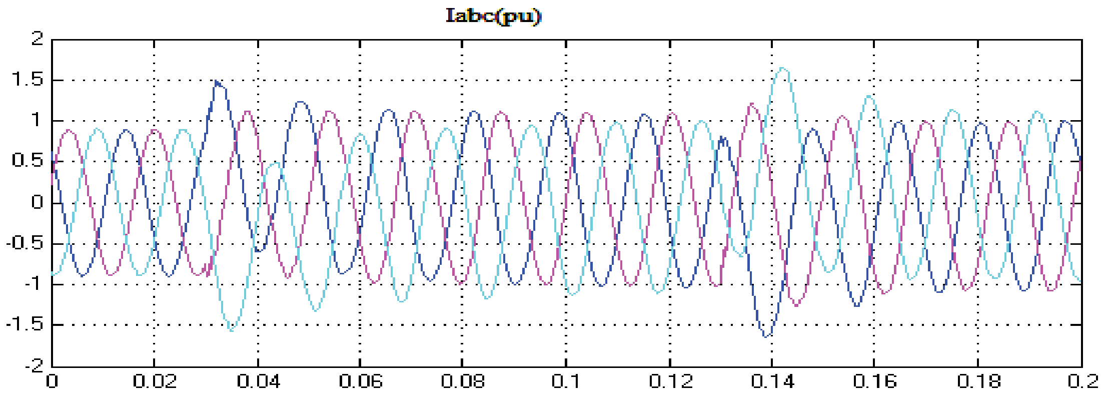


Figure 5.13: Current at the DFIG terminals in Pu (GA-based controller)

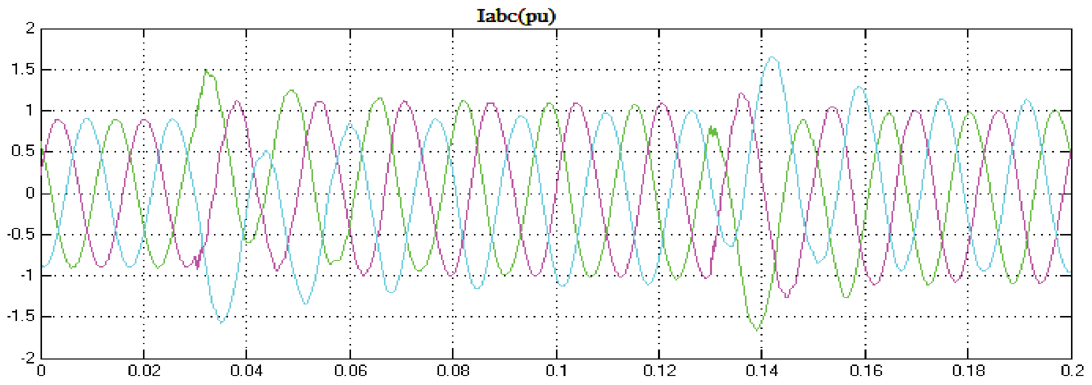


Figure 5.14: Current at the DFIG terminals in Pu (DE-based controller)

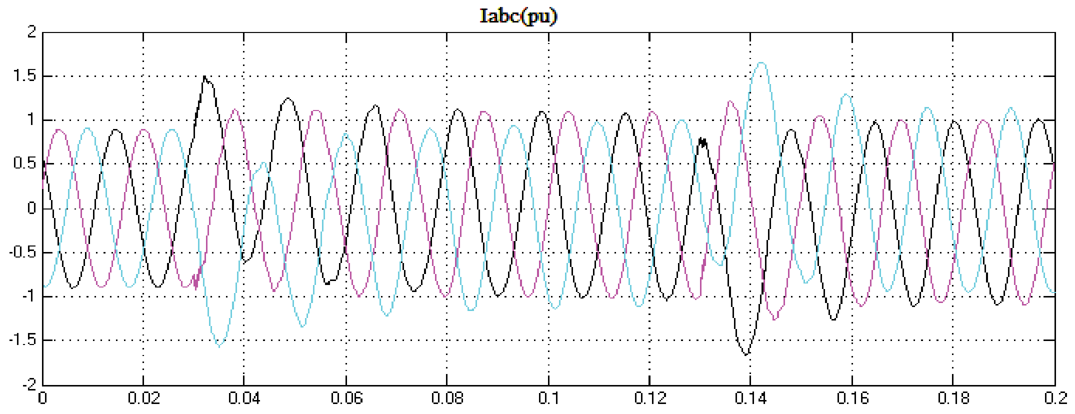


Figure 5.15: Current at the DFIG terminals in Pu (FFA-based controller)

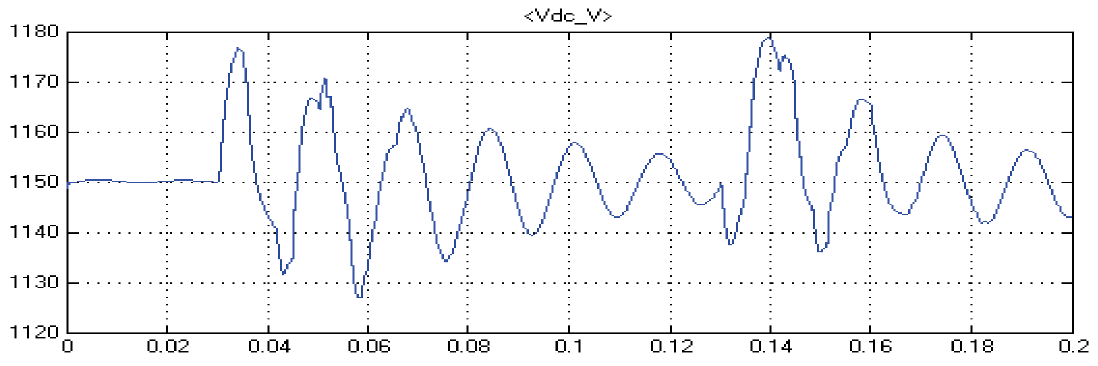


Figure5.16: DC link voltage at the common link capacitor (GA-based controller)

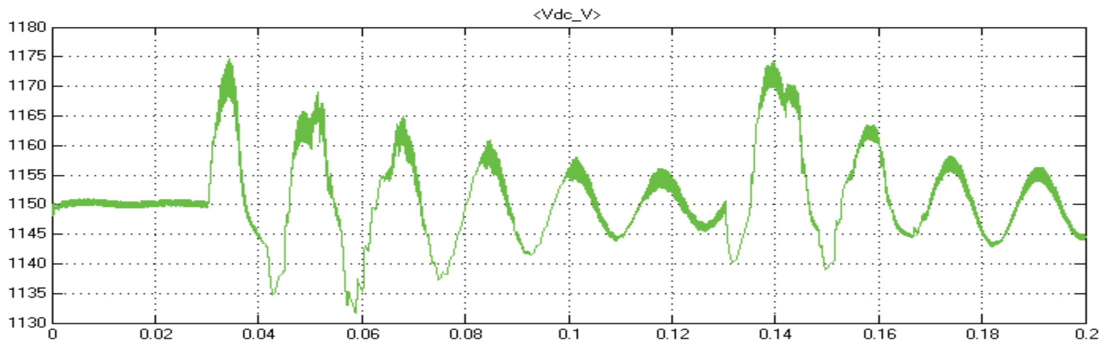


Figure 5.17: DC link voltage at the common link capacitor (DE-based controller)

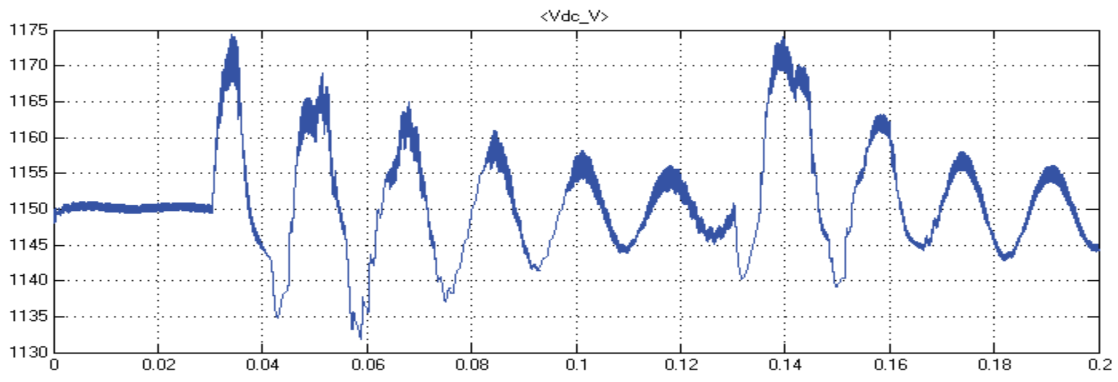


Figure5.18: DC link voltage at the common link capacitor (FFA-based controller)

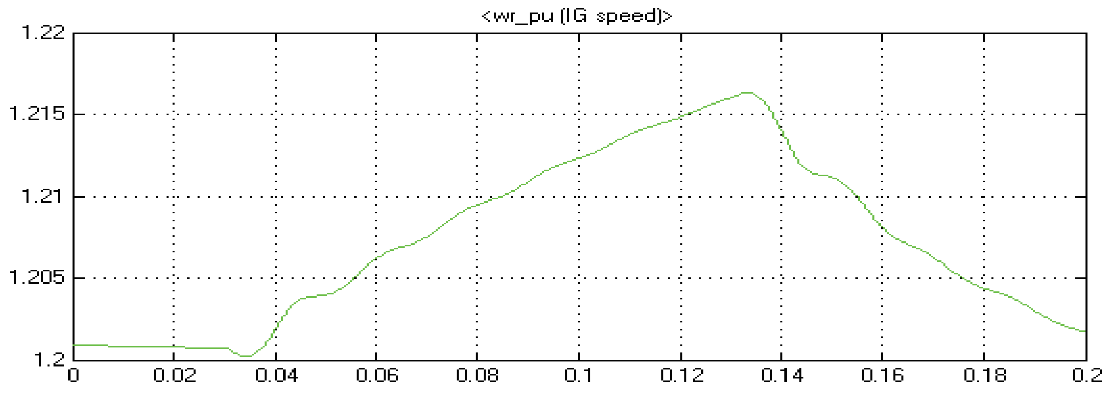


Figure 5.19: Generator speed in Pu (GA-based controller)

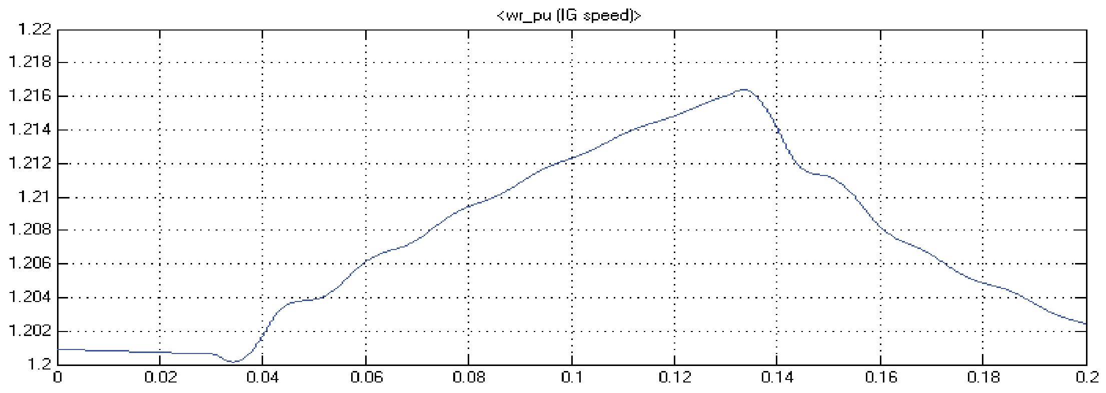


Figure5.20: Generator speed in Pu (DE-based controller)

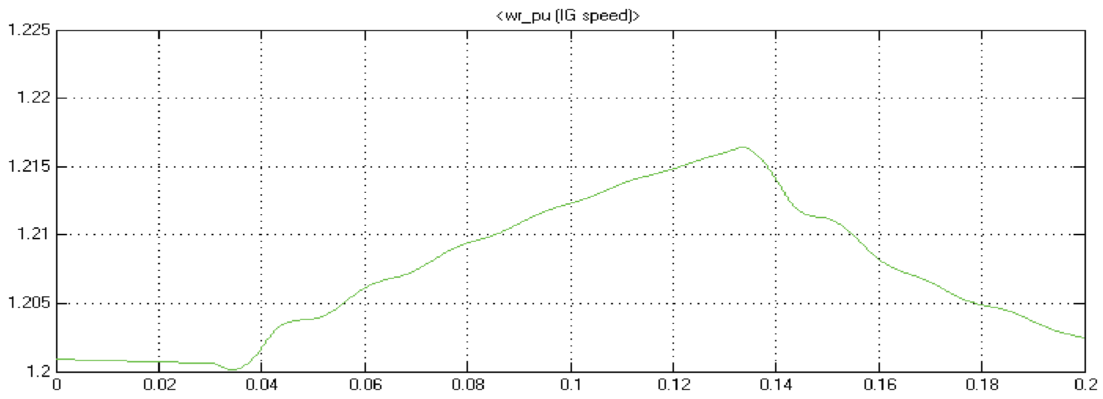


Figure 5.21: Generator speed in Pu (FFA-based controller)

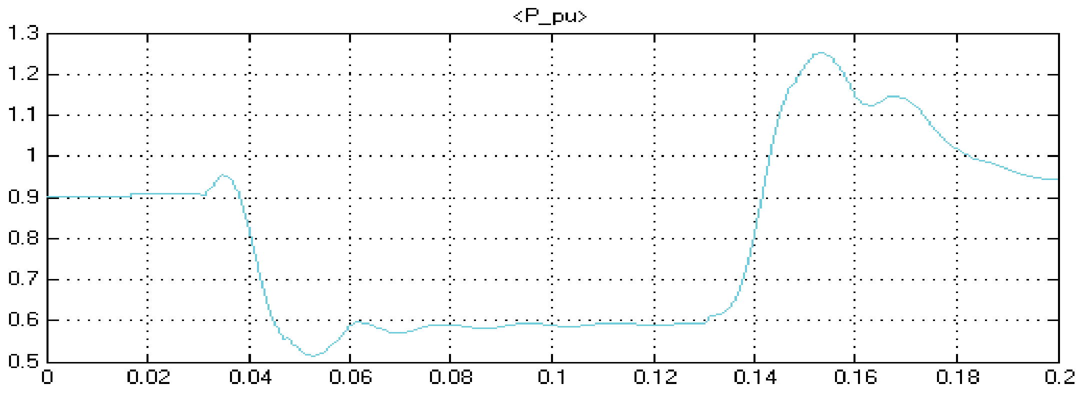


Figure 5.22: Active power given of the DFIG in Pu (GA-based controller)

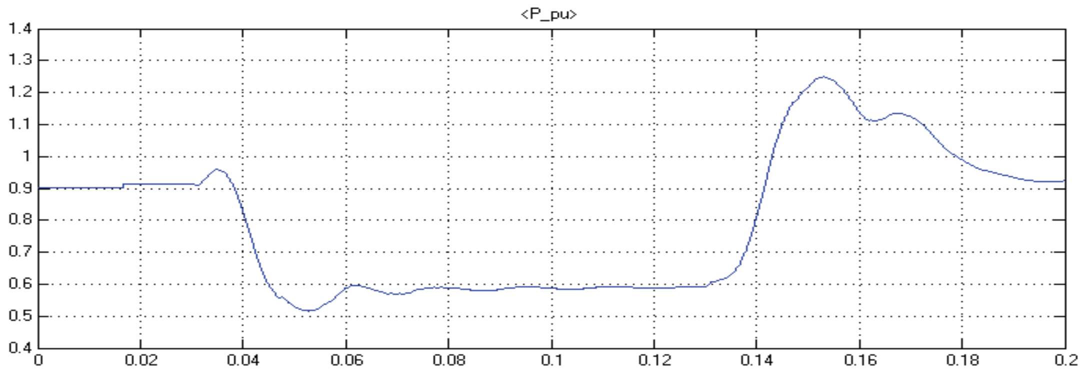


Figure 5.23: Active power given of the DFIG in Pu (DE-based controller)

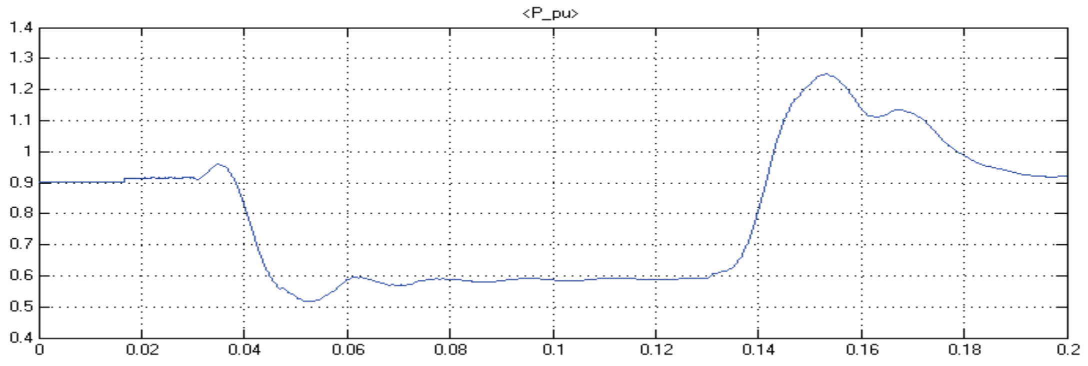


Figure 5.24: Active power given of the DFIG in Pu (FFA-based controller)

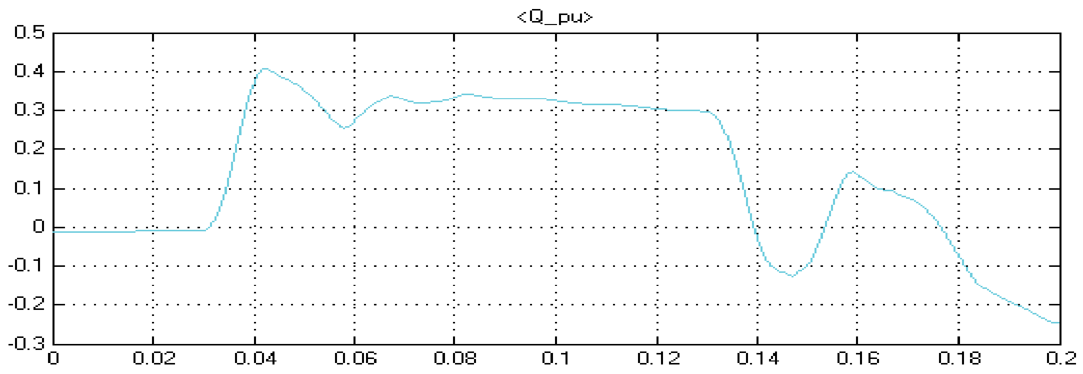


Figure5.25: Reactive power requirement of the DFIG in Pu (GA-based controller)

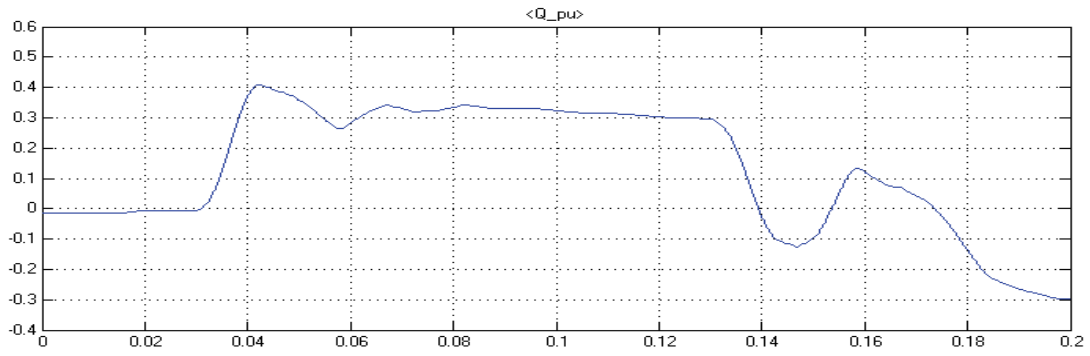


Figure 5.26: Reactive power requirement of the DFIG in Pu (DE-based controller)

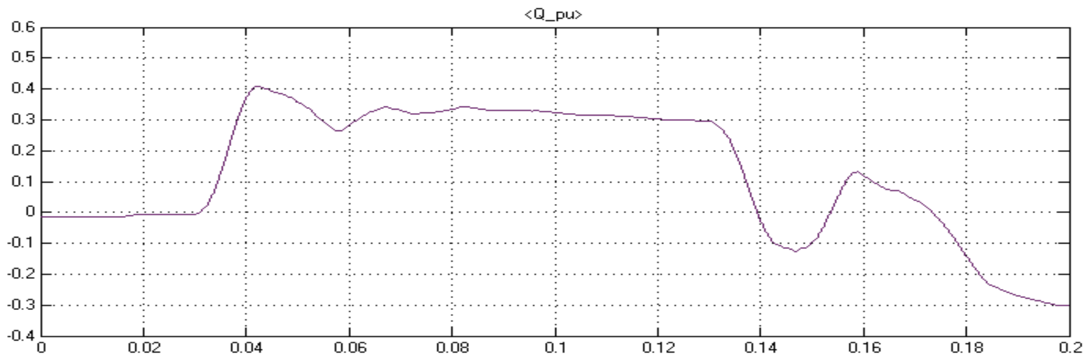


Figure 5.27: Reactive power requirement of the DFIG in Pu (FFA-based controller)

Here Figure (5.10-5.27) concluded that the response of the DFIG scheme about fatal voltage, current, active-reactive power along with DC-Link voltage with generator speed slightly improved with FFA based controller instead of a DE&GA- based controller.

5.7.2 DE-based Controller Response for 6th order Transfer function to the the DFIG system

The reaction of the DE-based PID controller intended in [44, 46], and system in revise is as publicized in Figure (5.28). Here open loop system even though stable with steady-state error100%, and the system has zero undershot in the response of open loop system. However, the step response of DE- based PID-controller for closed-loop in [44, 46] is shown in Figure (5.29). And the 6th order transfer function of DFIG system [44, 46] is given below.

$$G(s) = \frac{0.000324\, s^6 - 1.75\, s^5 - 2366\, s^4 + 7.9e06\, s^3 + 7.5e09\, s^2 + 5e12\, s + 2.18e14}{s^6 + 2340\, s^5 + 8.67e06\, s^4 + 4.79e09\, s^3 + 2.7e12\, s^2 + 1.27e14\, s + 9.6e14}$$

Open-loop step response of DFIG 6th order system

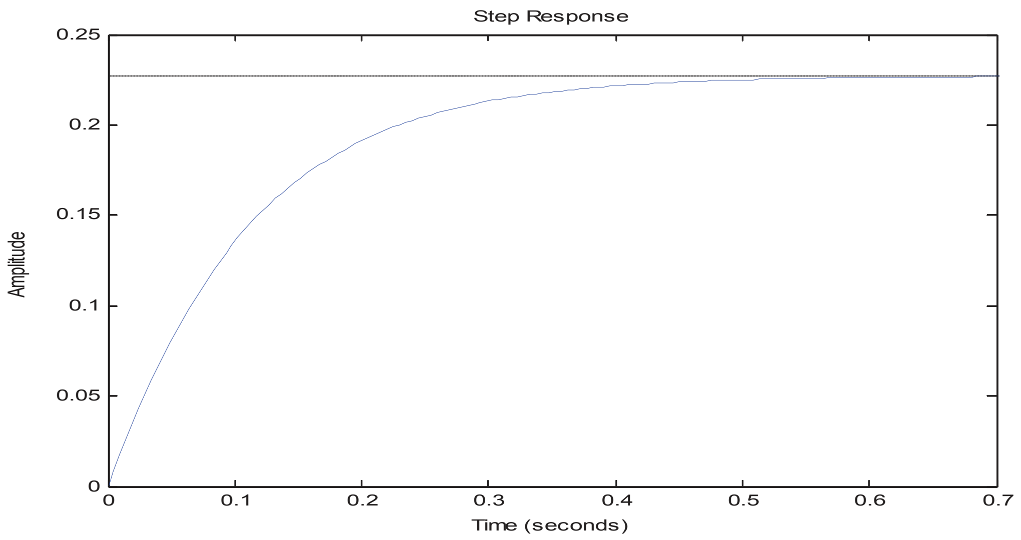


Figure 5.28: step response of DFIG (open loop)

Table 5.4: Step response of DFIG (open loop) observations of Figure (5.28)

Rise Time	Settling time	Settling Min	Settling Max	Over Shoot	Under Shoot	Peak	Peak Time
0.2359sec	0.4197sec	0.2046sec	0.2269sec	0.0 %	0.0 %	0.2269	0.7780 sec

Closed-loop step performance for DFIG 6th order transfer function system with DE-controller

Controller: $C = K_p + K_i \times \frac{1}{S}$ With $K_p = 14.7269$, $K_i = 137.9634$

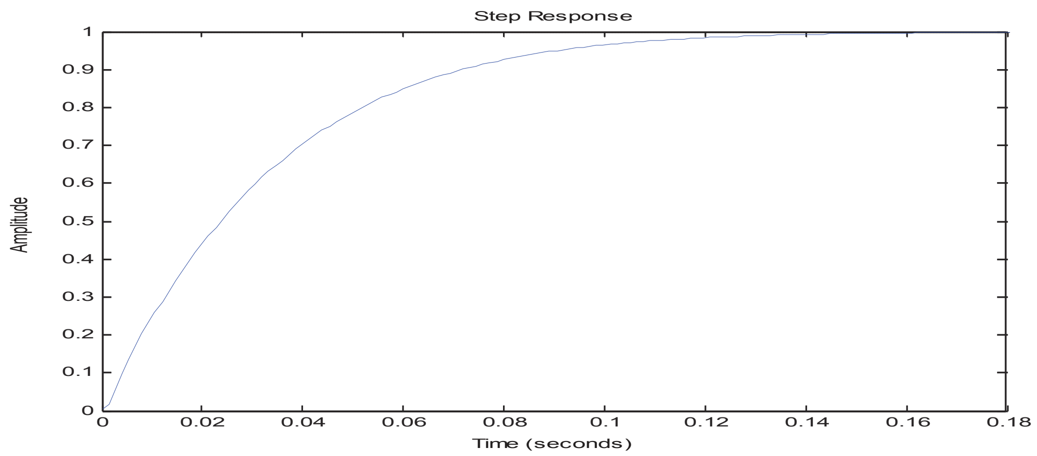


Figure 5.29: step response of DFIG using PID controller Optimized by DE

Table 5.5: Step response of DFIG (closed loop) observations of Figure (5.29)

Rise Time	Settling time	Settling Min	Settling Max	Over Shoot	Under Shoot	Peak	Peak Time
0.0675sec	0.1145sec	0.9008 sec	0.9992sec	0.0 %	0.0 %	0.9992	0.2115sec

5.7.3 FFA-based Controller Responsefor 6th order Transfer function to the the DFIG system

By using PID controller parameters that are designed in the previous Table (5.1), the step response of the system in revised for DE-based PID controller has been investigated, and step response of the system is exposed as in Figure (5.29). It is observable by a comparison result of DE- based PID control [46] along with one describe in this work which shows that; FFA- based PID controller has extended to the sufficiently damped DE-based controller without compromising the speed of response to the control loop. Now the step response of close loop reduced order system [46] by PID controller with FFA- technique is shown in Figure (5.30).

Closed-loop performance with PID controller optimized by Firefly Algorithm: Controller

$C = K_p + K_i \times \frac{1}{S}$, With K_P = 15, Ki = 140.9225

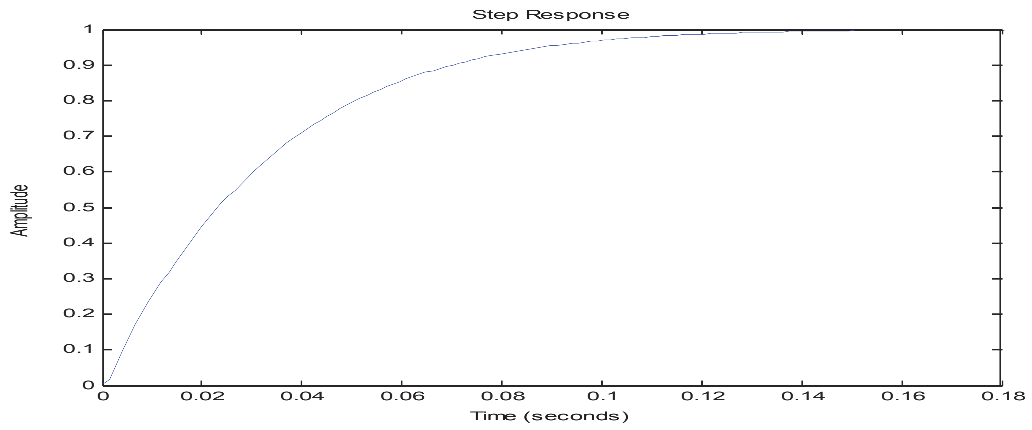


Figure 5.30: Step response of DFIG using PID controller optimized by FFA

Table5.6: Step response of DFIG (closed loop) observations of Figure (5.30)

Rise Time	Settling time	Settling Min	Settling Max	Over Shoot	Undershoot	Peak	Peak Time
0.0659sec	0.1106sec	0.9006sec	1.0000sec	0.0 %	0.0 %	1.0000	0.2103sec

5.7.4 The response of PID Controller Using GA Technique in 6th order Transfer function of DFIG system

The PID controller parameters which are designed in the previous Table (5.2), the step response of the system in revised for GA-based PID controller has been investigated, and step response of the system is exposed as in Figure (5.31). It is observable by a comparison result of DE- based PID control [44, 46], along with one illustrated in this work which shows that; GA- based PID controller has extended to sufficiently damped GA- based controller without compromising the speed of response to the control loop. Now the step response of close loop reduced order system [44, 46], by PID controller with GA- technique is shown in Figure (5.31).

Closed loop step response for 6th order DFIG system optimized by GA- Controller

$C = K_p + K_i \times \frac{1}{S}$, With $K_P = 10.7270$, $K_i = 102.4343$

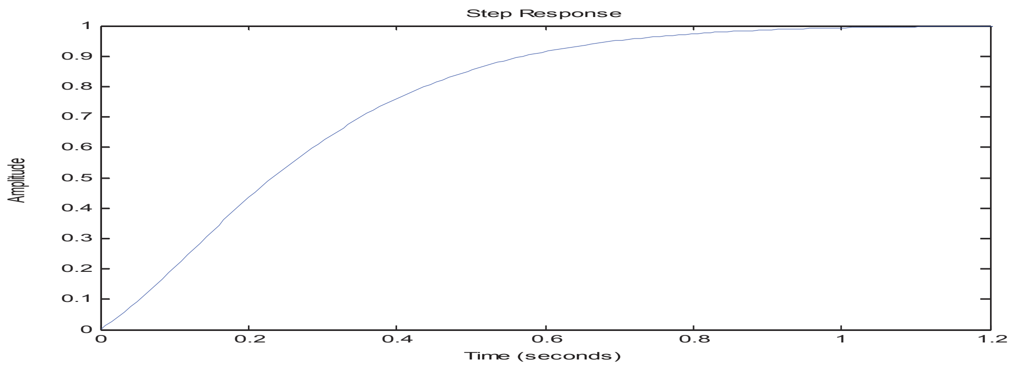
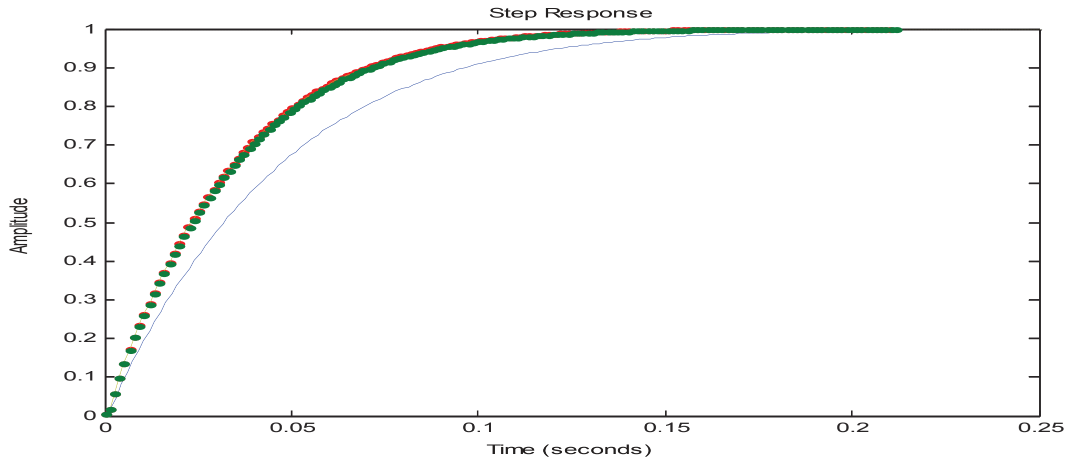


Figure 5.31: Step response of DFIG using PID controller optimized by GA

Table 5.7: Step response of DFIG (closed loop) observations of Figure (5.31)

Rise Time	Settling time	Settling Min	Settling Max	Over Shoot	Under Shoot	Peak	Peak Time
0.0911sec	0.1540sec	0.9032 sec	0.9999sec	0.0 %	0.0 %	0.9999	0.2540sec

5.7.5 Comparison between DE, GA & FFA responses



Red – Firefly Algorithm, Green – Differential Evolution, Blue – Genetic Algorithm

Figure 5.32: Comparison between DE, GA & FFA step responses

Comparison between DE, GA &FFA responses as performance parameters such as; Rise time, Settling time, Peak time, Overshoot and Peak are given in Table(5.8), concluded as:

(i) The designed FFA- based PID controller can reduce steady state error to zero as existing one.

(ii) The overshoot has been reduced to zero, and the response is no more underdamped.

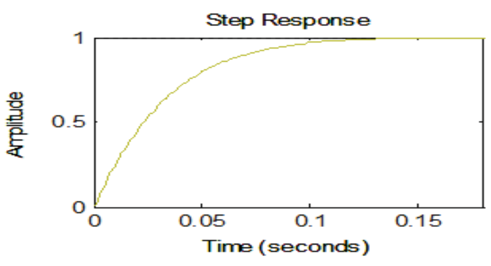
(iii) The speed response is much better than that of the current controller.

Hence intended controller in this chapter assists to eliminate reactive power variations in DFIG system.

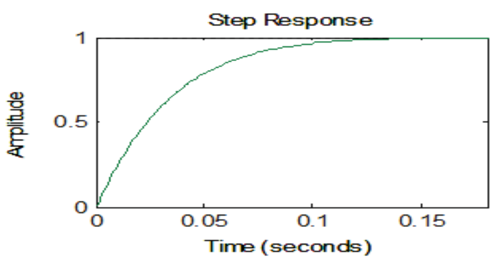
Table 5.8: Comparison of Rise time, Settling time, Peak time and Overshoot as well as Peak, between DE,GA & FFA-based PID controller

Controllers	Rise Time	Settling time	Peak Time	Overshoot	Peak
DE- based PID	0.0675sec	0.1145sec	0.2115sec	0%	0.9992
FFA- based PID	0.0659sec	0.1106sec	0.2103sec	0%	1.0000
GA- based PID	0.0911sec	0.1540sec	0.2540sec	0%	0.9999

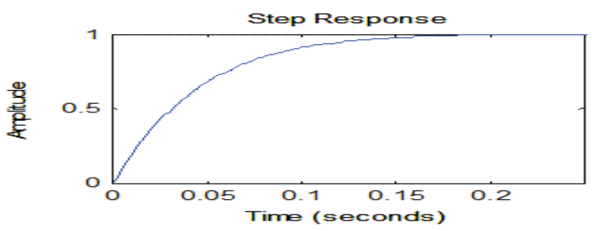
Now step responses of DE, GA &FFA evolutionary technique used as shown in Figure (5.33).



Step response of FFA-based technique



Step response of DE-based technique



Step response of GA-based technique

Figure 5.33: Step responses of DE, GA &FFA evolutionary technique used (individual)

From results, we can say that the performance of FFA is better than DE & GA. But the time taken in optimization is more in case of DE as compare to FFA.

The publications from this part of the thesis work are as follows:

- **Om Prakash Bharti**, R. K. Saket, S.K. Nagar, "*Controller Design for DFIG Based WT by Using Firefly Algorithm Technique*", Renewable and Sustainable Energy Review. Under Review 2018. SCI Journal (Thomson Reuters/Scopus/Web of Science).

- **Om Prakash Bharti**, R. K. Saket, S.K. Nagar, "*Controller Design for DFIG Based WT by Using Differential Evolution Technique", Renewable and Sustainable Energy Review*. Under Review 2018. SCI Journal (Thomson Reuters/Scopus/Web of Science).

- **Om Prakash Bharti,** R.K. Saket, S.K. Nagar, *"Controller Design of DFIG-based WT by Using DE-Optimization Techniques",* **4TH SICE International Symposium on Control Systems (ISCS)** , March 9-11, 2018 **, Setagaya Campus, Tokyo City University, Tokyo, Japan** and Publication in the IEEE Explore in **ISCS-2018**, **TCU Tokyo, Japan.**

5.8 Conclusion

Though the GA, DE-based PID controller improves system responses as compared with the open loop system; however, the FFA-based designed controller enhances the system response and also reduces the percentage overshoot equal to zero. The obtained results show that the system using FFA- based controller settles down in less time than the GA, DE-based PID controller scheme. It is concluded that the settling time is reduced and the percentage overshoots are reduced to zero using the proposed method. Finally, it is finished that the FFA- control technique provides another option to design a reliable and adequate controller for implementation in the DFIG-based wind turbine system for wind energy conversion scheme.

Chapter 6 conducts a simulation study for the on-off control based maximum power point tracking (MPPT) for doubly fed Induction generator (DFIG) based wind turbine and reliability assessment for wind energy conversion system (WECS).