Energy Conversion and Management 52 (2011) 264–273

Contents lists available at ScienceDirect

Energy Conversion and Management

journal homepage: www.elsevier.com/locate/enconman

# Off-line tuning of a PI speed controller for a permanent magnet brushless DC motor using DSP

Metin Demirtas [*]

*Electrical & Electronics Engineering Department, Balikesir University, Balikesir, Turkey*

ARTICLE INFO

*Article history:*
Received 20 February 2009
Received in revised form 3 November 2009
Accepted 29 June 2010
Available online 12 August 2010

*Keywords:*
Genetic
Neural network
Brushless DC motor
Optimization
DSP

ABSTRACT

In this paper, a new method of tuning Proportional Integral (PI) coefficients for a permanent magnet brushless DC (PMBLDC) motor drives is proposed. Artificial neural network is used to identify the whole system using maximum overshoot and settling time obtained from the application circuit for different $K_p$–$K_i$ pairs. Optimal values of PI controller coefficients are obtained using genetic algorithm. Motion Control Kit (MCK243) is used to carry out digital motion control applications. The MCK243 kit includes a power module and a three-phase brushless motor. TMS320F243 programs are used for PMBLDC motor speed control. Experimental results are given to show the validity of this method.

© 2010 Published by Elsevier Ltd.

## 1. Introduction

The Proportional Integral (PI) controller is unquestionably the most commonly used control algorithm in the process control industry. The main reason is its relatively simple structure, which can be easily understood and implemented in practice, and that many sophisticated control strategies are based on it. In spite of its wide spread use there exists no generally accepted design method for the controller. PI controllers have traditionally been tuned empirically, e.g. by the method described in Ziegler & Nichols. This method has the great advantage of requiring very little information about the process. There is, however, a significant disadvantage because the method inherently gives very poor damping [1].

The tuning of electric drive controller is a complex problem due to the many non-linearities of the machines, power converter and controller. Therefore, many tuning rules have been proposed for this type of controller. During the last three decades, one of the main focuses of research in control engineering has been devoted to provide automatic tuning of such controllers.

The permanent magnet synchronous motors (PMSM) have many applications in industries due to its compact structure, high efficiency, high power density, and high torque to inertia ratio. A robust PID control scheme is proposed by Jan et al. for the PMSM using a genetic searching approach. Numerical solutions of the Proportional Integral Derivative (PID) parameters constrained by three different objectives and simulation results are provided to illustrate the design procedure and the expected performances [2]. Espina et al. have studied the unwanted windup phenomenon reviewing and comparing different PI anti-windup strategies employed in speed control of electric drives. The tuning process of PI controllers is usually carried out considering the system as linear and therefore disregarding its physical limits such as maximum current and voltage [3]. Lee has presented a closed-loop estimation method of PMSM parameters by PI controller gain tuning. The idea of the proposed method is to tune the controller to cancel the pole of the motor transfer function with a controller zero and estimate motor parameters from the tuned controller gains [4]. Cao and Fan have investigated the uncertainties of permanent magnet synchronous servo motor; inertia, torque load and viscous damping coefficient are on-line identified based on recursive least-square estimator simultaneously. Then, a novel self-tuning PI controller based on iteration self-learning scheme is designed [5]. Zhu et al. have developed on-line identification methods based on model reference adaptive identification. Then a well-trained neural network supplies the PI controller with suitable gain according to each operating condition pair (inertia, angular velocity error, and angular velocity) detected. Self-tuning PI control technique based on neural network was executed in this research [6]. Du and Yu have researched in the double loop of PMSM speed adjustment systems, the current loop adopts PI control and the speed controller adopts compound control strategy with particle swarm optimization [7]. Wang et al. have proposed an auto-tuning algorithm for a Digital Signal Processor (DSP)-based PMSM drive. In order to be compatible with the conventional drives, PI speed control with gains indi-

* Tel.: +90 266 6121194; fax: +90 266 6121257.
*E-mail address:* mdtas@balikesir.edu.tr

0196-8904/$ - see front matter © 2010 Published by Elsevier Ltd.
doi:10.1016/j.enconman.2010.06.067

vidually designed by Bode diagram and an extension of the frequency-zone method to decouple the tuning gains in a multiloop control system is studied. The auto-tuning scheme includes two steps, an adaptive load observer to estimate the load inertia and adaptive two-degree-of-freedom PI tuning [8]. Pant et al. have worked a comparative study of three popular, evolutionary algorithms; genetic algorithms (GA), particle swarm optimization and differential evolution for optimal tuning of PI speed controller in PMSM drives [9]. Lo et al. have introduced an enhanced output predictive PI controller which overcomes the problems of large control action and derivative kick, which may arise in some situations. The practical problem of auto-tuning of the output predictive PI controller is considered in this paper [10]. Mudi et al. have studied an improved auto-tuning scheme for Ziegler–Nichols tuned PI controllers. With a view to improving the transient response, the proportional and integral gains of the proposed controller are continuously modified based on the current process trend [11]. Jiang et al. have proposed an improved evolutionary programming (EP) method with deterministic mutation factor for on-line PID parameters optimization of hydro-turbine governing systems. The mutation factors are usually generated with Gaussian or Cauchy random series in conventional evolutionary programming algorithms. Considering the difficulties of on-line optimal parameters settings resulting from non-linear time-variant hydro-turbine governing systems, this paper introduces deterministic chaos dynamics into the mutation operation of EP and provides a deterministic chaotic mutation evolutionary programming method [12].

System model is necessary for tuning controller coefficients in an appropriate manner (e.g. percent overshoot, settling time). Because of neglecting some parameters, the mathematical model cannot represent the physical system exactly in most applications. That's why, controller coefficients cannot be tuned appropriately.

Many of the recent developed computer control techniques are grouped into a research area called Intelligent Control, that result from the integration of Artificial Intelligent techniques within automatic control systems [13]. Artificial neural networks (ANNs) are one of these techniques and can be used to identify the system properly. Elmas et al. have studied a neuro-fuzzy controller for the speed control of a PMSM. A four layer neural network (NN) is used to adjust input and output parameters of membership functions in a fuzzy logic controller [14].

As mentioned in above references, optimization process of PI coefficients is generally based on mathematical model and methods. These studies include a lot of error, such as linearization, neglecting and not estimating parameters. There are no neglecting and estimating parameters of the system in this work. Also, there are no equations of the inverter, BLPMDC motor and DSP. Therefore, this work will provide off-line tuning PI coefficients using real system parameters. In this study, the whole system is modelled by ANNs using input/output data obtained the real system consisted of three-phase inverter, BLPMDC and DSP. Mathematical equations are not used for modelling the system. The modelling process is realized according to inputs controller coefficients ($K_p$, $K_i$) and outputs maximum overshoot and settling time. Optimization process is performed via the ANN model using GA. In this paper, firstly experimental setup is illustrated. Then, modelling system with ANN and optimization of PI coefficients by using GA is explained. Results are presented in the discussion section, and the conclusion is in the final section.

## 2. Experimental setup

MCK243 kit is a complete motion structure, including a power amplifier and a motor, thus offering the basic platform for motion applications evaluation. This kit is evaluation kit that allows you to experiment with and use the TMS320F243 (F243) Digital Signal Processor (DSP) controller for digital motion control (DMC) applications. External power modules may be easily interfaced with the DSP board through a universal motion control bus (MC-BUS). The MCK243 kit includes such a power module and a three-phase brushless motor. TMS320F243 programs for PMBLDC motor speed control.

The accompanying software of the MCK243 kit (monitor, chip evaluation applications, advanced IDE graphical analysis toll) represents a basic evaluation and development platform for motion application engineers.

The MC-BUS connectors include the basic I/O signals required in standard motion control applications with DC, AC or step motors. Fig. 1 presents the block diagram of the PM-50 board.

The board contains a 1.7 A, 36 V three-phase inverter, and can be connected via the MC-BUS connectors to the MCK243 board. Motor phases, encoder and Hall signals are also connected to the PM-50 board. 3-phase brushless motor coupled with 500-line quadrature incremental encoder and three hall position sensors [15]. The used experimental setup is shown in Fig. 2.

### 2.1. Basic structure of the control scheme for the PMBLDC motor application

The PMBLDC application control scheme is presented in Fig. 3. As one can see, the scheme is based on the measurement of two phase currents and of the motor position. The speed estimator

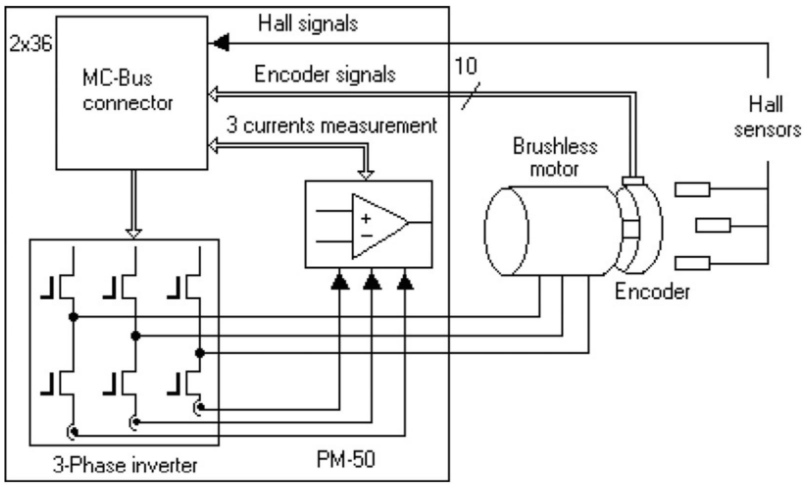


**Fig. 1.** The block diagram of the PM-50 power module.

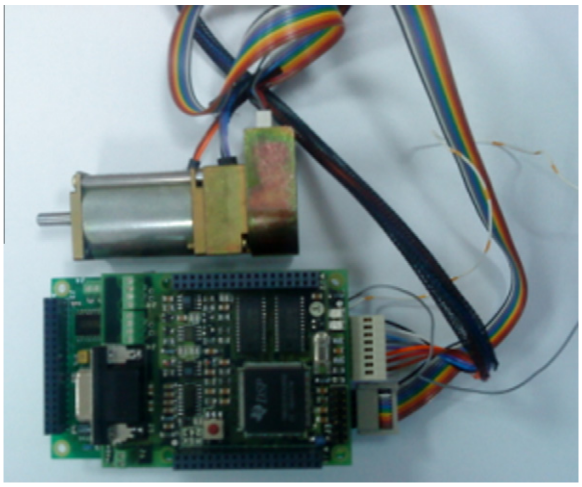


**Fig. 2.** The experimental setup is shown.

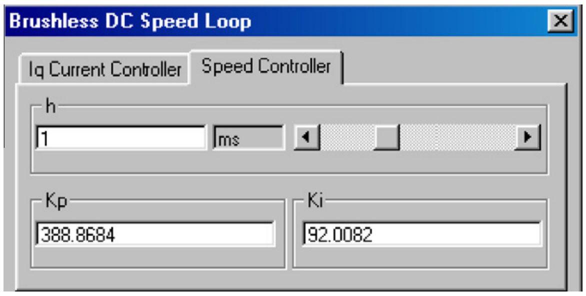
block is a simple difference block. The measured phase currents, $i_a$ and $i_b$, are used to compute the equivalent DC current in the motor, based on the Hall sensors position information. Remark that the Hall sensors give a 60 electrical degrees position information. The speed and current controllers are PI discrete controllers. Only one current controller is needed in this case, similar to a DC motor case. The voltage commutator block implements (by software) the computation of the phase voltages references, Vas*, Vbs* and Vcs*, applied to the inverter. Practically, the six full compare PWM outputs of the DSP controller are directly driven by the program, based on these reference voltages. In the PMBLDC case, only four of the inverter transistors are controlled for a given position of the motor. The scheme will commute to a specific command configuration, for each of the 60° position sectors, based on the information read from the Hall sensors.

The speed reference signal is obtained using the reference generator of DSPMOT 32, included in the application. Thus, the speed reference may be imposed at Windows level, from DSPMOT 32.

Also the controllers' parameters are set in DSPMOT 32, at Windows level, from the Motion Setup Controller menu command. The proportional and integral control factors, as well as the sampling periods for the current and speed control loops, are set using this DSPMOT 32 command [15].

### 2.2. Motion control applications (MCK243 kit)

DSPMOT is an integrated, graphical-environment analysis toll for DMC applications. It offers you the possibility of analysing your DSP program variables by using on-line watches, or off-line tracking of real-time stored data. Furthermore, a point-to-point linear interpolation reference generator block may easily be included into your DSP application and its parameters may be set in the windows environment of the DSPMOT program. Similar facilities may be used for standard speed/current PI controllers The controllers' parameter PI is entered via the dialog box shown in Fig. 4. DSP program is executed after entering PI coefficients to run the PMBLDC.


Brushless DC Speed Loop

Iq Current Controller | Speed Controller

h: 1 ms

Kp: 388.8684

Ki: 92.0082

**Fig. 4.** The Motion Setup Controller.

The main purpose of this dialog is to allow the examination and/or modification of the parameters of the digital controllers implemented on the DSP board. These operations may be done before the execution of the DSP program, in order to set the initial values of the controller's parameters.

*h:* the sampling period for the selected controller, expressed in milliseconds. Based on this value, DSPMOT will compute the required parameters in order to properly set the real-time interrupts on the MCK243 board.

$K_p$–$K_i$: the proportional, respectively the integral constant of the discrete PI controller. These values are converted by DSPMOT, for the DSP program level, into sets of two parameters, the scaled values (normalised in Q15 format), and the associated scaling factors.

The motor reference is the input in the motor control block, containing the speed controller in the case of the PMBLDC application. This block will generate the reference of the quadrature current ($i_q$), which provides the motor torque. The outputs of this control block are the PWM reference signals.

Finally, the Pulse Width Modulation (PWM) reference signals are used the PWM generator block, to drive the power inverter. A symmetric PWM generation technique was used for the application [15].

## 3. Modelling of the PMBLDC motor using ANN

ANNs are successfully used in a lot of areas such as control, early detection of electrical machine faults, and digital signal pro-

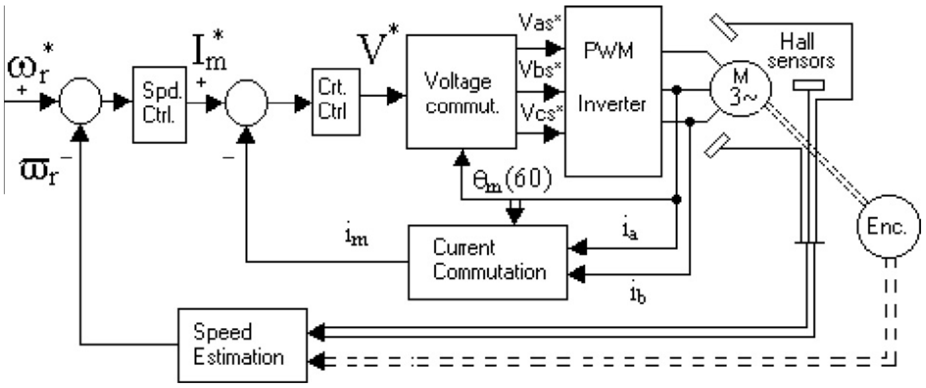


**Fig. 3.** PMBLDC control scheme.

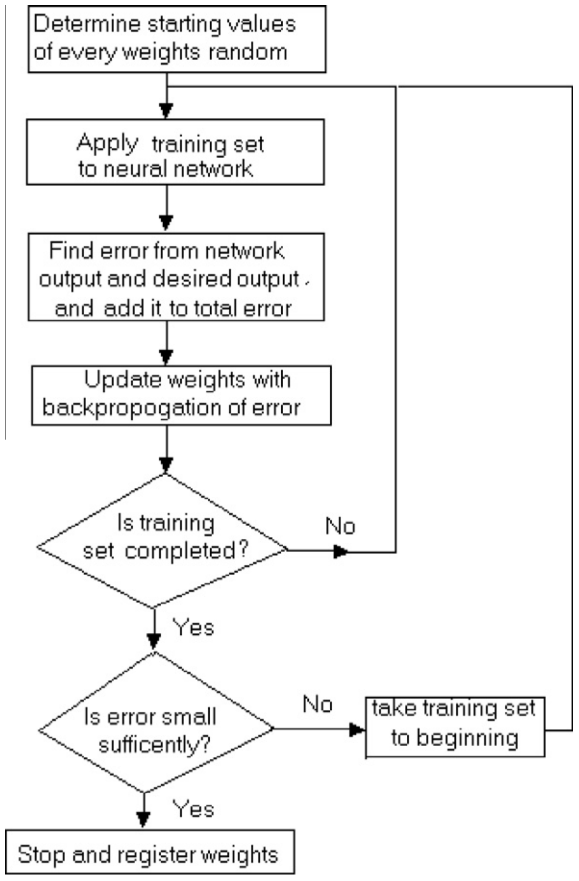

Determine starting values of every weights random
Apply training set to neural network
Find error from network output and desired output and add it to total error
Update weights with backpropogation of error
Is training set completed?
No
Yes
Is error small sufficently?
No
take training set to beginning
Yes
Stop and register weights

**Fig. 5.** The flow chart of training process.

cessing in everyday technology. The memory of a neural network lies in the weights and biases. The neural networks can be classified, in terms of how the weights and biases are obtained, into three categories. ANNs have been used in non-linear systems modelling and simulation. In general, ANNs are simply mathematical techniques designed to accomplish a variety of tasks. ANNs consist of an inter-connection of a number of neurons. There are different network types like cascade forward back propagation, feed forward back propagation, competitive, generalized regression and radial basis. The back propagation algorithm is a popular algorithm that has different variants. These are cascade forward, Elman, feed forward and time delay back propagation algorithms. The structure and operation of back propagation neural networks are not complex [16].

Multi-layer perceptrons (MLPs) are the simplest and therefore most commonly used neural network architectures. The backpropagation algorithm is the most commonly adopted MLP training algorithm. This type of neural network is known as a supervised network, because it requires a desired output in order to learn. The goal of this type of network is to create a model that correctly maps the input to the output using historical data, so that the model can then be used to produce the output when the desired output is unknown.

The ANN model used is a multi-layer perceptron model, in which there is more than one layer between input and output. The backpropogation of the error algorithm used as the training algorithm is used for training of generalized delta rule. The training process of this ANN model is shown Fig. 5.

Thirty sets of input–output data taken from the application circuit are given in Table 1. The ANN structure for this system is shown in Fig. 6, where $K_p$ and $K_i$, $M_o$, and $T_s$ are PI coefficients, the maximum overshoot, the settling time, respectively. The ANN parameters of the modelled system are presented in Table 2.

There was no criterion to select cell number at every layer of the ANN structure; layer number and cell number were determined

**Table 1**
Data used for the training of the ANN.

| Data set | $K_p$ | $K_i$ | $M_o$ (rpm) | $T_s$ (ms) |
|---|---|---|---|---|
| 1 | 1950 | 15 | 104 | 400 |
| 2 | 1950 | 50 | 104 | 139 |
| 3 | 1950 | 100 | 105 | 79 |
| 4 | 1950 | 225 | 105 | 35 |
| 5 | 1950 | 350 | 104 | 24 |
| 6 | 1950 | 500 | 104 | 23 |
| 7 | 1500 | 15 | 101 | 300 |
| 8 | 1500 | 50 | 103 | 92 |
| 9 | 1500 | 100 | 103 | 50 |
| 10 | 1500 | 225 | 104 | 25 |
| 11 | 1500 | 350 | 105 | 23 |
| 12 | 1500 | 500 | 105 | 22 |
| 13 | 1000 | 15 | 103 | 200 |
| 14 | 1000 | 50 | 103 | 61 |
| 15 | 1000 | 100 | 103 | 32 |
| 16 | 1000 | 225 | 104 | 24 |
| 17 | 1000 | 350 | 105 | 20 |
| 18 | 1000 | 500 | 109 | 23 |
| 19 | 500 | 15 | 102 | 90 |
| 20 | 500 | 50 | 102 | 23 |
| 21 | 500 | 100 | 105 | 18 |
| 22 | 250 | 15 | 102 | 34 |
| 23 | 250 | 50 | 102 | 44 |
| 24 | 250 | 100 | 102 | 62 |
| 25 | 100 | 15 | 102 | 80 |
| 26 | 100 | 50 | 102 | 123 |
| 27 | 50 | 15 | 102 | 144 |
| 28 | 50 | 50 | 106 | 23 |
| 29 | 10 | 15 | 105 | 331 |
| 30 | 10 | 1 | 103 | 240 |

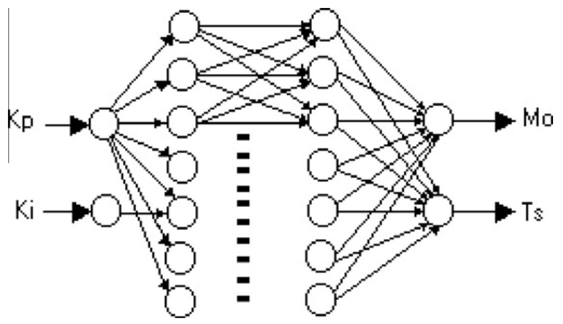


**Fig. 6.** The ANN model structure of the system.

**Table 2**
The ANN parameters for the model system.

| Parameter | Value |
|---|---|
| Number of neurons for input layer | 2 |
| Number of neurons of the output layer | 2 |
| Layer number | 2 |
| First layer cell number | 7 |
| Second layer cell number | 7 |
| First layer activation function | Sigmoid |
| Second layer activation function | Sigmoid |
| Maximum iteration number | 30,000 |
| Error limit | 0.0001 |
| Training coefficient | 0.7 |
| Momentum coefficient | 0.9 |

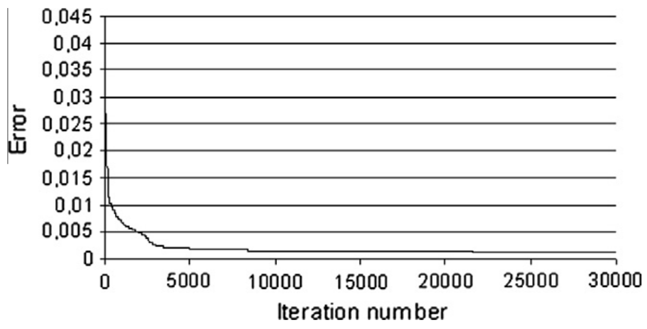


**Fig. 7.** Mean-squared error values according to iteration number.

**Table 3**
Fitness values of the members and GA parameters in the first generation.

| Parameter | Value |
|---|---|
| Population size | 30 |
| Crossover operator | 0.90 |
| Mutation size | 0.80 |
| Fitness of member 1 | 0.008504 |
| Fitness of member 2 | 0.007954 |
| Fitness of member 3 | 0.007856 |
| Fitness of member 4 | 0.007854 |
| Fitness of member 5 | 0.007824 |
| Fitness of member 6 | 0.007821 |
| Fitness of member 7 | 0.007795 |
| Fitness of member 8 | 0.007776 |
| Fitness of member 9 | 0.007769 |
| Fitness of member 10 | 0.007763 |
| Optimal fitness = 0.008504 | |

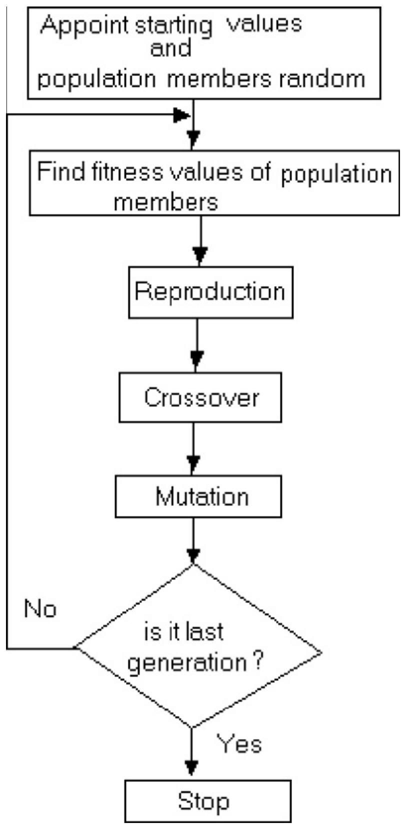

Appoint starting values and population members random
Find fitness values of population members
Reproduction
Crossover
Mutation
is it last generation ?
No
Yes
Stop

**Fig. 8.** The flow chart of the GA.

with experiment. In the same way, the learning and momentum coefficients were determined by experiences at previous studies.

A part of the training data and change in the error for the training process are shown in Fig. 7. Mean-squared error reduced to lower than 0.0016 by 15,000 iterations, but iteration was still continued to 30,000 iterations. The training process was finished when mean-squared error reduces to 0.0001 at 30,000 iterations.

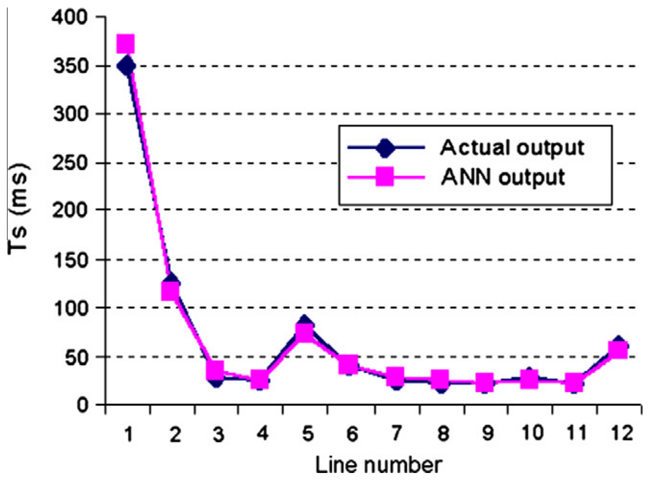


**Fig. 9.** The change in the settling time.

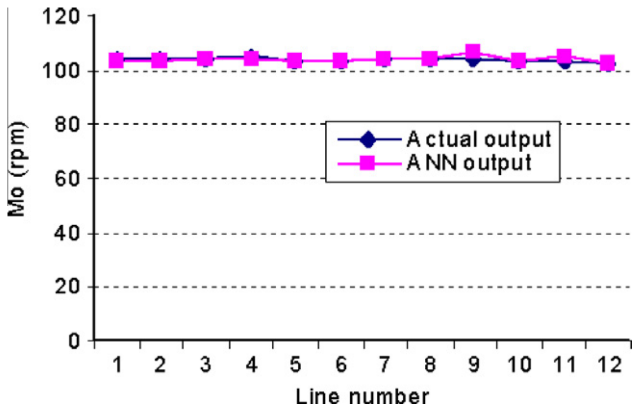


**Fig. 10.** The change in the maximum overshoot.

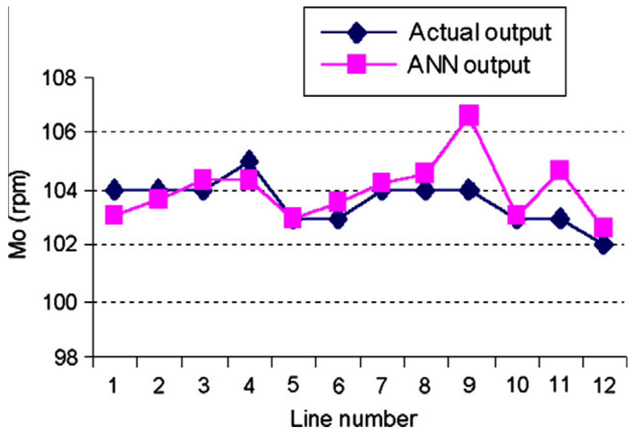


**Fig. 11.** The change in the maximum overshoot.

## 4. Optimization of PI coefficients using GA

GAs are based on an analogy to the genetic code in our own DNA (deoxyribonucleic acid) structure, where its coded chromosome is composed of many genes. GA approach involves a population of individuals represented by strings of characters or digits. Each string is, however, coded with a search point in the hyper search-space. From the evolutionary theory, only the most suited individuals in the population are likely to survive and generate off-spring that passes their genetic material to the next generation.

The GA is a subset of evolutionary algorithms that model biological processes to optimize highly complex cost functions. A ge-

**Table 4**
Data used for testing of ANN output.

| | | Actual output | | ANN output | |
|---|---|---|---|---|---|
| $K_p$ | $K_i$ | $M_o$ (rpm) | $T_s$ (ms) | $M_o$ (rpm) | $T_s$ (ms) |
| 1750 | 15 | 104 | 350 | 103.0917 | 368.6841 |
| 1750 | 50 | 104 | 125 | 103.5981 | 115.9434 |
| 1750 | 225 | 104 | 28 | 104.2734 | 31.9541 |
| 1750 | 350 | 105 | 24 | 104.301 | 23.17653 |
| 1250 | 50 | 103 | 82 | 102.923 | 74.15874 |
| 1250 | 100 | 103 | 39 | 103.5141 | 39.82892 |
| 1250 | 225 | 104 | 23 | 104.2241 | 27.91058 |
| 1250 | 350 | 104 | 20 | 104.5569 | 22.89344 |
| 1250 | 500 | 104 | 21 | 106.5633 | 22.38476 |
| 750 | 100 | 103 | 28 | 103.0109 | 25.02495 |
| 750 | 225 | 103 | 20 | 104.6668 | 22.48236 |
| 480 | 21 | 102 | 62 | 102.6398 | 55.80698 |

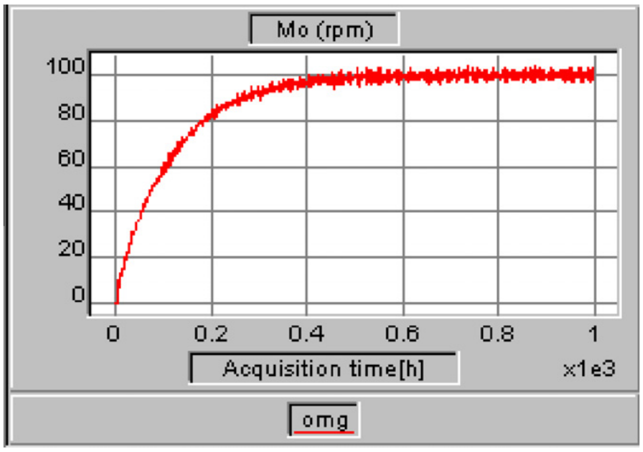


**Fig. 14.** Rotor speed versus time for $K_p = 1750$, $K_i = 15$ ($M_o = 104$, $T_s = 452$).

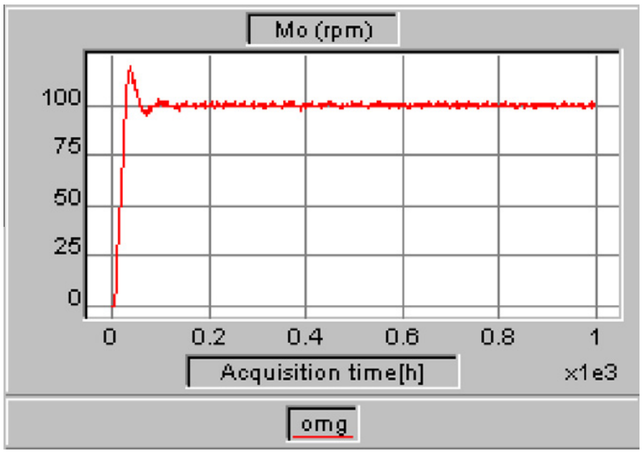


**Fig. 12.** Rotor speed versus time for $K_p = 140$, $K_i = 20$ ($M_o = 102$, $T_s = 52$).

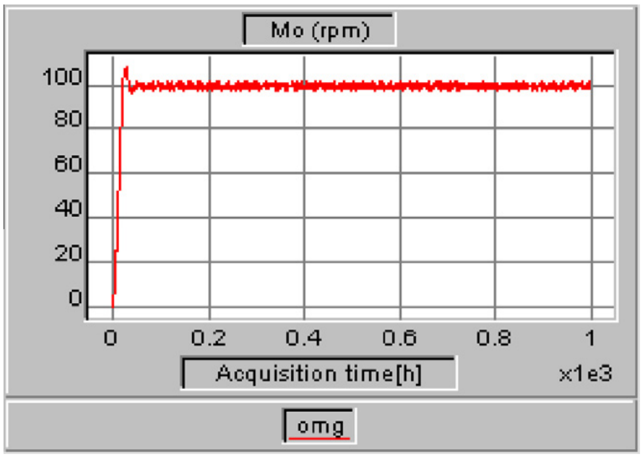


**Fig. 15.** Rotor speed versus time for optimum $K_p = 388{,}868{,}392$, $K_i = 92{,}008{,}196$ ($M_o = 102$, $T_s = 31$).

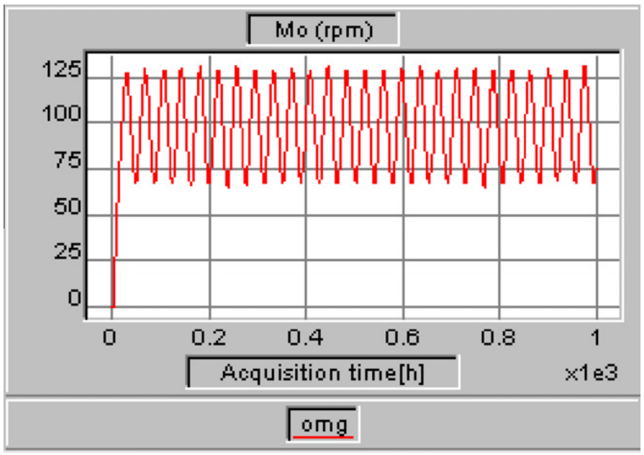


**Fig. 13.** Rotor speed versus time for $K_p = 10$, $K_i = 50$ ($M_o = 130$, system is unstable).

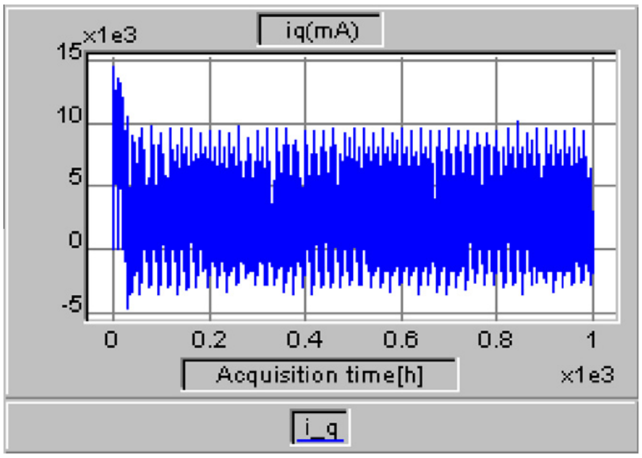
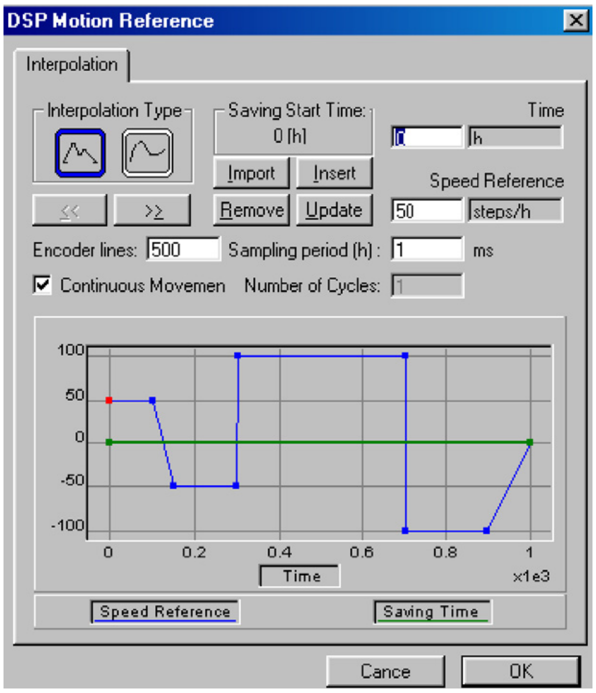


**Fig. 16.** Change in the DC equivalent current ($i_q$) for optimum $K_p = 388{,}868{,}392$, $K_i = 92{,}008{,}196$.

netic algorithm allows a population composed of many individuals to evolve under specified selection rules to a state that maximizes the “fitness” (i.e., minimizes the cost function). Some of the advantages of a genetic algorithm include that it

- optimizes with continuous or discrete parameters,
- does not require derivative information,
- simultaneously searches from a wide sampling of the cost surface,
- deals with a large number of parameters,
- is well suited for parallel computers,
- optimizes parameters with extremely complex cost surfaces; they can jump out of a local minimum,
- provides a list of optimum parameters, not just a single solution,
- may encode the parameters so that the optimization is done with the encoded parameters, and
- works with numerically generated data, experimental data, or analytical functions [17].

In general GAs run repeatedly by using three basic operators such as reproduction, crossover and mutation, to find the best parameters in the whole parameter searching space. GAs are global numerical optimization methods, patterned after the natural processes of genetic recombination and evolution. First, an initial population is produced randomly. Then, genetic operators are applied


DSP Motion Reference

Interpolation

Interpolation Type

Saving Start Time: 0 [h]

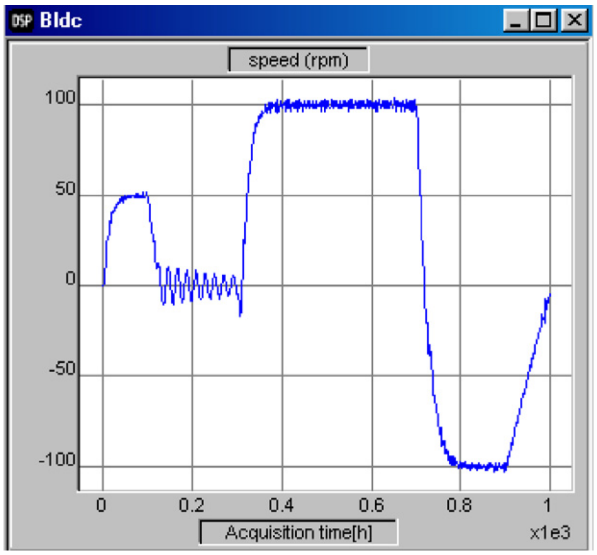
Time: 0 h

Speed Reference: 50 steps/h

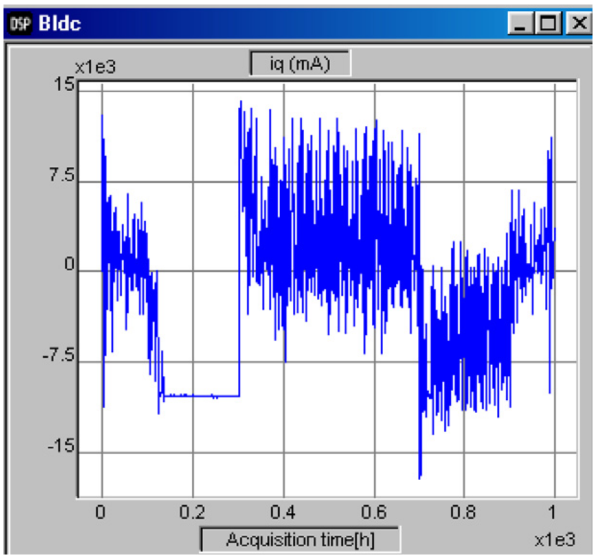
Encoder lines: 500 Sampling period (h) : 1 ms

☑ Continuous Movemen Number of Cycles: 1

(a) Speed reference



(b) The change in the speed



(c) The change in the $i_q$

**Fig. 17.** A different reference speed application for $K_p = 1500$, $K_i = 100$.

to the population to improve their fitness gradually. The procedure yields a new population at each iteration [18].

The GA used in this paper known as the simple genetic algorithm. Different crossover and mutation rates are used for processing of optimization of genetic algorithms. Ten of the fitness values obtained, listed from the largest fitness value to the smallest, and the fitness values of the members of the first generation are shown in Table 3. The flow chart of the GA is shown in Fig. 8 [19].

A PI controller with the transfer function.

$G_c(s) = K_p + \frac{K_i}{s}$ is employed to control the process.

The optimum values for the $K_p$ and $K_i$ pairs were obtained using a computer program written in C++ language for the GA. This process executes with three different operators at bit level. 30 of the $K_p$ and $K_i$ pairs were determined at random. $K_p$ and $K_i$ consisted of 15 bits and 12 bits, respectively. These $K_p$ and $K_i$ pairs were entered to ANN model as input. The maximum overshoot and settling time were obtained from the ANN output. These values were then used as the fitness function.

The one-point crossover method was used on the crossover operator. Mutual parameters of two random members on the crossover were divided into two parts and their positions were changed. A random bit of a random number on the mutation process was changed 0–1 and 1–0. For the optimization process, mutation rate is increased when converge occurs in 5–10 generation. Therefore, early converge is prevented, and in addition, members that have high fitness values were obtained.

The range of $K_p$ and $K_i$ values chosen lay between (10-1950) and (15-500) respectively. The fitness function is defined as

$$f = \frac{1}{M_o + T_s + 1}$$

In this algorithm, the genetic algorithm parameters are selected for the training cycles were:

- Population size: 30
- Number of generations: 60
- Crossover rate:0.60
- Mutation rate:0.04
- Chromosome length: 30 bits (15 each for $K_p$ and $K_i$)

## 5. Results and discussion

The changes in the settling time and maximum overshoot for the actual system output and ANN output were given in Table 4. These data are different from data in Table 1.

The obtained model was tested with data given in Table 4 that was not in the training set after the training process, in order to discern the appropriateness of the ANN model. The change in the settling time with $K_i$ and $K_p$ for actual system and ANN model, given in Table 4, is shown in Fig. 9.

The change in the maximum overshoot value of the speed with $K_i$ and $K_p$ for the actual system and ANN model are demonstrated in Figs. 10 and 11, respectively.

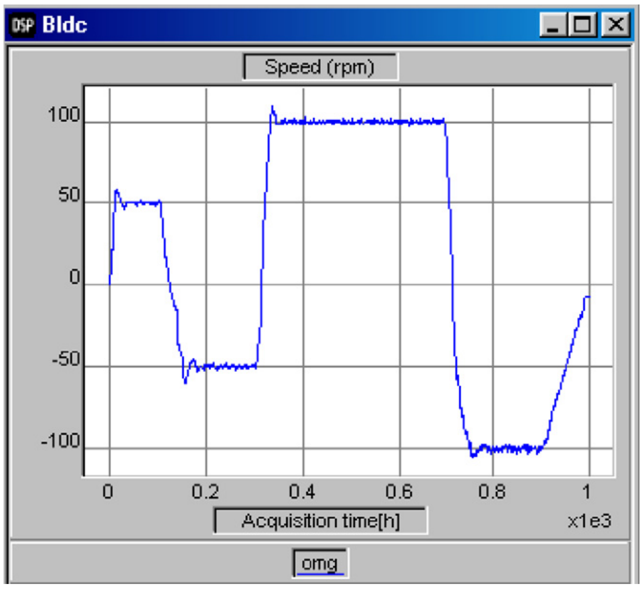
Fig. 11 shows the zoom in version of Fig. 10. The ANN model follows the system output, with a small error that arises from differences between experimental conditions and the model of the nonlinear system. It shows that the ANN model created for the system models it successfully.

The optimum PI coefficients were found to be: $K_p$ = 388,868,392, $K_i$ = 92,008,196 (Generation number: 100). Optimal fitness value was not change after generation 100. Therefore, optimal $K_p$ and $K_i$ value are taken for generation number 100. The responses of the system for these values of $K_p$ and $K_i$ are shown in Fig. 15. The settling time is shorter and the maximum overshoot is minimized for these values. This shows that full system is a good control system.

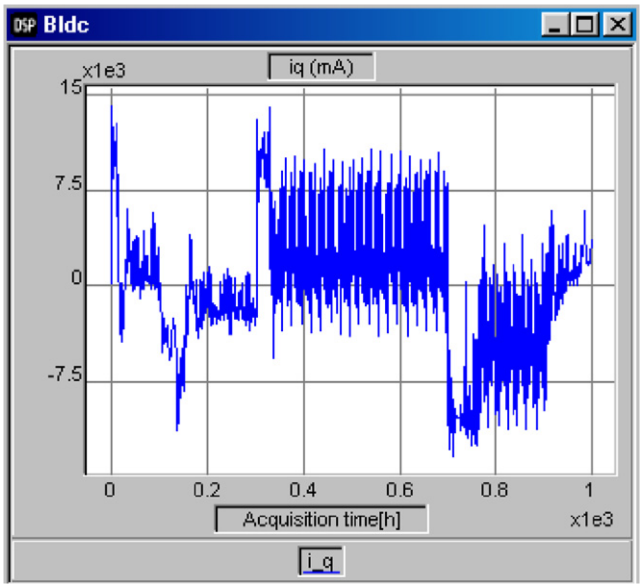
The change in the maximum overshoot of the speed with different $K_p$ and $K_i$ values for the actual system were demonstrated in Figs. 12–14.

$I_q$ current is estimated from measured phase currents. As shown in Fig. 15, the settling time is shorter and maximum overshoot is lower than the others. $I_q$ current for optimum $K_p$ and $K_i$ is shown in Fig. 16.

Correctness of the optimum $K_p$ and $K_i$ pairs was tested for a different reference speed shown in Fig. 17a, in Appendix A. The obtained results from the actual system for a different $K_p$ ($K_p$ = 1500) and $K_i$ ($K_i$ = 100), and optimum $K_p$ and $K_i$ pairs are shown in Figs. 17 and 18, respectively, in Appendix A. As shown in Fig. 18, the maximum overshoot is minimized and the settling time is shorter for optimum values.



(a) The change in the speed



(b) The change in the $i_q$

**Fig. 18.** The same reference speed application in Fig. 17.

## 6. Conclusion

In this paper, the optimal PI coefficients design method that achieves high performance for a PMBLDC motor using GA was proposed. Actual system (motor, and inverter and DSP) was modelled by ANN. It was also determined that the maximum overshoot and settling time are very small if the system is controlled by control parameters obtained from the optimization process which uses GA.

The results presented show that ANN method has improved dynamic performance. It was found that GA is suitable for optimization of controller coefficients by the performance criteria considered. This process can be also applied for non-linear systems controlled by PD and PID controller. The PI optimization method given in this paper can be adopted by the industry. This method is very useful for non-linear and complex systems.

## Appendix A

*A.1. PMBLDC motor parameters*

P: 50 watt
Phase resistance: 7.5 ohm
Phase inductance: 480 mH
Back-EMF constant: 2.1 V/1000 rpm
Torque constant: 20 mNm/A
Rated voltage: 10 V
Max. Voltage: 36 V.
Rotor inertia: $4.6 \times 10^{-7}$ kgm$^2$
Mechanical time constant: 8.6 ms

*A.2. Some of the obtained results using GA were given as below*

Generation number: 1.

```
Member=1 ***** Fitness value=0.008504 ****
Member=2 ***** Fitness value=0.007954 ****
Member=3 ***** Fitness value=0.007856 ****
Member=4 ***** Fitness value=0.007854 ****
Member=5 ***** Fitness value=0.007824 ****
Member=6 ***** Fitness value=0.007821 ****
Member=7 ***** Fitness value=0.007795 ****
Member=8 ***** Fitness value=0.007776 ****
Member=9 ***** Fitness value=0.007769 ****
Member=10***** Fitness value=0.007763 ****
Optimal fitness = 0.008504
K_p = 404.182047, K_i = 130.737707
```

Generation number: 15.

```
Member=1 ***** Fitness value=0.008567 ****
Member=2 ***** Fitness value=0.008527 ****
Member=3 ***** Fitness value=0.007828 ****
Member=4 ***** Fitness value=0.007828 ****
Member=5 ***** Fitness value=0.007828 ****
Member=6 ***** Fitness value=0.007828 ****
Member=7 ***** Fitness value=0.007828 ****
Member=8 ***** Fitness value=0.007828 ****
Member=9 ***** Fitness value=0.007828 ****
Member=10 ***** Fitness value=0.007828 ****
Optimal fitness value = 0.008567
K_p = 406.150073, K_i = 105.942622
```

Generation number: 30.

```
Member = 1 ***** Fitness value = 0.008567 ****
Member = 2 ***** Fitness value = 0.008191 ****
Member = 3 ***** Fitness value = 0.008191 ****
Member = 4 ***** Fitness value = 0.008188 ****
Member = 5 ***** Fitness value = 0.008188 ****
Member = 6 ***** Fitness value = 0.008188 ****
Member = 7 ***** Fitness value = 0.008188 ****
Member = 8 ***** Fitness value = 0.008188 ****
Member = 9 ***** Fitness value = 0.008188 ****
Member = 10 ***** Fitness value = 0.008188 ****
Optimal fitness value = 0.008567
K_p = 406.150073, K_i = 105.942622
```

Generation number: 71.

```
Member = 1 ***** Fitness value = 0.008580 ****
Member = 2 ***** Fitness value = 0.008579 ****
Member = 3 ***** Fitness value = 0.008579 ****
Member = 4 ***** Fitness value = 0.008459 ****
Member = 5 ***** Fitness value = 0.008456 ****
Member = 6 ***** Fitness value = 0.008253 ****
Member = 7 ***** Fitness value = 0.007659 ****
Member = 8 ***** Fitness value = 0.007659 ****
Member = 9 ***** Fitness value = 0.007659 ****
Member = 10 ***** Fitness value = 0.007659 ****
Optimal fitness value = 0.008580
K_p = 392.189413, K_i = 88.319674
```

Generation number: 100.

```
Member = 1 ***** Fitness value = 0.008583 ****
Member = 2 ***** Fitness value = 0.008564 ****
Member = 3 ***** Fitness value = 0.008564 ****
Member = 4 ***** Fitness value = 0.008564 ****
Member = 5 ***** Fitness value = 0.008562 ****
Member = 6 ***** Fitness value = 0.008562 ****
Member = 7 ***** Fitness value = 0.008562 ****
Member = 8 ***** Fitness value = 0.008562 ****
Member = 9 ***** Fitness value = 0.008562 ****
Member = 10 ***** Fitness value = 0.008562 ****
Optimal fitness value = 0.008583
K_p = 388.868392, K_i = 92.008196
```

## References

[1] Wang Y, Shao H. Optimal tuning for PI controller. Automatica 2000;36:147–52.
[2] Jan Rong-Maw, Tseng Chung-Shi, Liu Ren-Jun. Robust PID control design for permanent magnet synchronous motor: a genetic approach. Electric Power Syst Res 2008;78:1161–8.
[3] Espina Jordi, Arias Antoni, Balcells Josep, Ortega Carlos. Speed anti-windup PI strategies review for field oriented control of permanent magnet synchronous machines. IEEE Power Electr Contr Power Syst 2009:279–85.
[4] Lee Seok-Beom. Closed-loop estimation of permanent magnet synchronous motor parameters by PI controller gain tuning. IEEE Trans Energy Convers 2006;21(4):863–70.
[5] Cao Xianqing, Fan Liping. Self-tuning PI controller for permanent magnet synchronous motor based on iterative learning control. In: IEEE second international symposium on intelligent information technology application; 2008. p. 756–60.
[6] Zhu Jianguang, Zhang Zhifeng, Tang Renyuan. IEEE self-tuning PI controller based on neural network for permanent magnet synchronous motor. in: Fourth international conference on natural computation; 2008. p. 532–7.

[7] Du Chun-Yu, Yu Gwo-Ruey. Optimal PI control of a permanent magnet synchronous motor using particle swarm optimization. in: IEEE ICIC '07 second international conference; 2007. p. 255–8.
[8] Wang Ming-Shyan, Shau Tzu-Chang, Chang Chia-Ming. DSP-based auto-tuning design of permanent magnet synchronous motor drives. In: The 33rd annual conference of the IEEE industrial electronics society (IECON), Taipei, Taiwan; 2007. p. 1044–8.
[9] Pant Millie, Thangaraj Radha, Abraham Ajith. Optimal tuning of PI speed controller using nature inspired heuristics. In: IEEE eighth international conference on intelligent systems design and applications, vol. 3; 2008. p. 420–5.
[10] Lo WL, Rad AB, Tsang KM. Auto-tuning of output predictive PI controller. ISA Trans 1999;38:25–36.
[11] Mudi Rajani K, Dey Chanchal, Lee Tsu-Tian. An improved auto-tuning scheme for PI controllers. ISA Trans 2008;47:45–52.
[12] Jiang Chuanwen, Ma Yuchao, Wang Chengmin. PID controller parameters optimization of hydro-turbine governing systems using deterministic–chaotic-mutation evolutionary programming (DCMEP). Energy Convers Manage 2006;47:1222–30.
[13] Astrom KJ, Hang CC, Persson P, Ho WK. Towards intelligent PID control. Automatica 1992;28(1):1–9.
[14] Elmas Cetin, Ustun Oguz, Sayan Hasan H. A neuro-fuzzy controller for speed control of a permanent magnet synchronous motor drive. Expert Systems with Applications 2008;34:657–64.
[15] Technosoft DSP Motion Solutions, MxWIN243 User Manuel, Switzerland, October 2001.
[16] Golcu Mustafa. Artificial neural network based modeling of performance characteristics of deep well pumps with splitter blade. Energy Convers Manage 2006;47:3333–43.
[17] Randy LH, Haupt SE. Practical genetic algorithms. A wiley-Interscience publication, John Wiley & Sons, Inc.; 1998.
[18] Dimeo R, Lee KY. The use of a genetic algorithm in power plant control system design. In: IEEE proceeding of the 34th conference on decision and control; 1995. p. 737–42.
[19] Ustun Seydi Vakkas, Demirtas Metin. Modeling and control of V/f controlled induction motor using genetic-ANFIS algorithm. Energy Convers Manage 2009;50:786–91.