24th COBEM - 2017

24th ABCM International Congress of Mechanical Engineering
December 3-8, 2017, Curitiba, PR, Brazil

COBEM-2017-1317
CONTROLLER TUNING BASED ON OPTIMIZATION METAHEURISTICS APPLIED TO A TWIN-ROTOR SYSTEM

Guilherme Felipe da Silva
Electrical Engineering Graduate Program (PPGEE), Federal University of Parana (UFPR), Curitiba, Brazil
gfsilva.eng@gmail.com

André Luiz Luppi
Felipe Hartcopp Betoni
Computer Engineering, Undergraduate course, Pontifical Catholic University of Parana (PUCPR), Curitiba, Brazil
andrell_alemao@hotmail.com , felipe@younicsolutions.com.br

Ivan Lucas Reis Silva
Electrical Engineering Graduate Program (PPGEE), Federal University of Parana (UFPR), Curitiba, Brazil
ivanlucas13@gmail.com

Leandro dos Santos Coelho
Industrial and Systems Engineering Graduate Program (PPGEPS), Pontifical Catholic University of Parana (PUCPR), Curitiba, Brazil, and Electrical Engineering Graduate Program (PPGEE), Federal University of Parana (UFPR), Curitiba, Brazil
leandro.coelho@pucpr.br

Abstract. *Proportional-integral-derivative (PID) control is the most popular control architecture used in industrial problems. Many techniques have been proposed to tune the gains for the PID controller. Over the last few years, as an alternative to the conventional mathematical approaches, modern optimization metaheuristics, such as evolutionary computation and swarm intelligence paradigms, have been given much attention by many researchers due to their ability to find good solutions in PID tuning. This paper proposes a comparative study of PID tuning based on six optimization metaheuristics applied to an experimental twin-rotor system. The adopted optimization metaheuristics are: (i) firefly algorithm, (ii) harmony search, (iii) multi-verse optimization, (iv) grey wolf optimizer, (v) ant lion optimizer, and (vi) whale optimization algorithm. The results show that the evaluated optimization approaches present good performance in the PID controller tuning when applied to the twin-rotor system in terms of the closed-loop performance.*

Keywords: *Optimization metaheuristics, swarm intelligence, evolutionary algorithms, controller design.*

1. INTRODUCTION

One of the most popular controllers in industrial processes is the proportional-integral-derivative (PID) controller. This control strategy offers a simple and effective solution for many real problems. About 90% of the control problems are solved by using some type of PID controller (Levine, 1996).

On the other hand, evolutionary algorithms have yielded promising results for solving nonlinear, non-differentiable and multi-modal optimization problems. Due to its population-based nature, they can avoid being trapped in a local optimum, and, consequently, have the ability to find global optimal solutions. The use of optimization metaheuristics to tune gains of PID controllers (Hunaini *et al.*, 2016) has demonstrated ability of finding a set of good solutions. It is not restricted to controllers, bioinspired optimization metaheuristics are being applied to different optimization problems such as fault classification (Abdelgayed *et al.*, 2017) and artificial neural networks design (Aljarah *et al.*, 2017).

Metaheuristic optimization algorithms are problem-solving methods which try to find good-enough solutions to hard optimization problems, at a reasonable computation time, where classical approaches fail, or cannot even be applied (Salcedo-Sanz, 2016).

In this paper, a comparative study of PID control parameters tuning based on six optimization metaheuristics applied to a twin-rotor system is realized. The twin rotor system (TRS) imposes challenging control problems due to its given

nonlinearities as well as significant couplings between the pitch axis and the azimuth axis (Butt and Aschemann, 2015).

The adopted metaheuristics to the PID tuning are the following: (i) firefly algorithm (FA) (Yang, 2009), (ii) harmony search (HS) (Geem and Kim, 2001), (iii) multi-verse optimization (MVO) (Mirjalili *et al.*, 2015), (iv) grey wolf optimizer (GWO) (Mirjalili *et al.*, 2014), (v) ant lion optimizer (ALO) (Mirjalili and Lewis, 2015), and (vi) whale optimization algorithm (WOA) (Mirjalili and Lewis, 2016).

The remainder of the paper is arranged as follows. In Section 2, the fundamentals of the PID control are presented. After, the details about the adopted optimization metaheuristics are introduced in Section 3. The description of the twin-rotor case study is approached in Section 4. In Section 5, the optimization and closed-loop control results are analysed. Finally, concluding remarks are made in the last section.

2. FUNDAMENTALS OF PID CONTROLLER

The first step in the design of a PID controller is its tuning, that is, the choice of parameters design that satisfy the conditions proposed by the project. In a PID controller, there are three parameters (gains) to be determined: K_p (proportional), K_i (integral) and K_d (derivative).

The proportional control action is, in essence, an operational amplifier of adjustable gain, being responsible for minimizing the acting error (Ogata, 2012), the integral control is the control action responsible for zeroing the residual error in steady state, and the derivative control has an anticipatory characteristic, that is, it produces an action contrary to the error during the transient regime of the signal to be controlled and improves the speed of response of the system and reduces the value of the signal. It is important to note that there is no purely derivative control action.

The sum of proportional, integral and derivative control actions constitutes a PID controller, which is given according to Eq. (1) given by

$$u(t) = K_p \cdot e(t) + K_i \cdot \int_0^t e(t)\,dt + K_d \cdot \frac{de(t)}{dt} \tag{1}$$

where t is the time variable, $u(t)$ is the output control signal, the K_p, K_i and K_d are the gains as stated earlier and $e(t)$ is the error signal.

3. DESCRIPTION OF OPTIMIZATION METAHEURISTICS

In the subsections below, all the six optimization metaheuristics used in this work are introduced and briefly explained.

3.1 Firefly algorithm (FA)

The FA proposed by Yang (2009) is inspired by the flashing behavior of fireflies. Yang formulated the algorithm assuming that: a) all fireflies are unisexual, so that any individual firefly will be attracted to all other fireflies; b) attractiveness is proportional to their brightness, and for any two fireflies, the less bright one will be attracted by (and thus move towards) the brighter one; c) if there are no fireflies brighter than a given firefly, it will move randomly. The FA pseudocode is shown in Fig. 1.

```
Objective function:  f(x), with x = (x_1, ..., x_d)^T
Generate an initial population of fireflies x_i (i = 1, 2, ..., n)
Formulate light intensity I_i in x_i as f(x_i)
Define absorption coefficient γ
while(t<Max number of iterations)

          for i = 1 : n (all n fireflies)
                    for j = 1 : n (n fireflies)
                              if (I_i < I_j)
                                        Move firefly i towards j
                              endif
                              Attractiveness varies with distance r via e^(-γr)
                              Evaluate new solutions and update light intensity
                    end for j
          end for i
          Rank the fireflies and find the current best
          t = t + 1
end while
Postprocess results and visualization
```

Figure 1. Firefly algorithm pseudocode.

As stated by Yang (2010), there are two important issues in the FA: the variation of light intensity and formulation of the attractiveness. The light intensity $I(r)$ varies according to the inverse square law and is approximated using the Gaussian form shown in Eq. (2), where r is the distance, I_0 is the original light intensity and γ is the absorption coefficient, that typically varies from 0.01 to 100, where

$$I(r) = I_0 e^{-\gamma r^2} \tag{2}$$

As a firefly's attractiveness is proportional to the light intensity seen by adjacent fireflies, the attractiveness β of a firefly can be defined by

$$\beta(r) = \beta_0 e^{-\gamma r^m}, (m \geq 1) \tag{3}$$

The distance between any two fireflies*i* and *j* at $x_i$and x_j, respectively, is the Euclidian distance as defined in Eq. (4):

$$r_{ij} = \left\| x_i - x_j \right\| = \sqrt{\sum_{k=1}^{d} (x_{i,k} - x_{j,k})^2} \tag{4}$$

where $x_{i,k}$ is the kth component of the spatial coordinate x_i of the ith firefly. Finally, Eq. (5) shows how a firefly *i* moves towards a more attractive (brighter) firefly *j*:

$$x_i = x_i + \beta_0 e^{-\gamma r_{ij}^2}(x_j - x_i) + \alpha \left(rand - \frac{1}{2}\right) \tag{5}$$

where $\alpha \in [0, 1]$ and *rand* is a random number uniformly generated in range [0,1].

3.2 Harmony search (HS)

HS algorithm is a metaheuristic, proposed by Geem and Kim (2001), with common characteristics in evolutionary algorithms. However, it is based on the process of musical performance that happens when a musician tries to reach a better state of harmony (Geem *et al*., 2001), as well as during jazz improvisation.

The HS algorithm includes some quantities of optimization operators, like a harmony memory (*HM*), the harmony memory size (*HMS*, number of solution vectors in the harmony memory), the improvisations number (*NI*) the harmony memory considering rate (*HMCR*) and the pitch adjusting rate (*PAR*). The HS pseudocode is shown in Fig. 2.

```
Objective function f(x), x=(x1, x2, ...xd)T
Generate initial harmonics (real number arrays)
Define pitch adjusting rate (rpa), pitch limits and bandwidth
Define harmony memory accepting rate (raccept)
while (t<Max number of iterations)
            Generate new harmonics by accepting best harmonics
            Adjust pitch to get new harmonics (solutions)
            if(rand>raccept)
                        choose na existing harmonic randomly
            else if(rand>rpa)
                        adjust the pitch randomly withing limits
            else
                        generate new harmonics via randomization
            end if
            Accept the new harmonics (solutions) if better
            t = t + 1
end while
Find the current best solution
```

Figure 2. Pseudocode of the HS

The HS algorithm starts generating multiple solution vectors randomly, $x_1 \ldots x^{HMS}$, based in three control parameters, randomization, *HMCR* and *PAR*, that are stored in a matrix called harmony memory (*HM*) given by

$$HM = \begin{bmatrix} x_1^1 & \cdots & x_2^1 & f(x^1) \\ \vdots & \ddots & \vdots & \vdots \\ x_1^{HMS} & \cdots & x_n^{HMS} & f(x^{HMS}) \end{bmatrix} \tag{6}$$

where *n* is the vector solution dimension (number of variables to be optimized), *f*(.) is the objective function, and HMS is the number of solution vectors stored in *HM*.

One of the major disadvantages of metaheuristic algorithms is to define the parameters of the same, as in the case of the HS algorithm, the parameters *HMS*, *PAR*and *HMCR*. To avoid this problem, a modification proposed by Mahdavi *et al.*(2007) was adopted, where the *PAR* parameter is varied according to Eq. (7) given by

$$PAR = PAR_{min} + \frac{PAR_{max} - PAR_{min}}{MaxIter - 1}(Iter - 1) \tag{7}$$

where PAR_{min} is the minimum initial *PAR*, PAR_{max} is the maximum initial *PAR*, *Iter* is the current iteration, and *MaxIter* is the maximum iteration (the stopping criterion adopted in this paper).

Thus, the value of *PAR* is linearly increased from PAR_{min} to PAR_{max} according to the iterations. Mahdavi *et al.* (2007) suggest values in range [0.35, 0.45] as PAR_{min} and 0.99 as PAR_{max}. In addition to simplifying the *PAR* parameter determination, its variation can improving the HS performance.

3.3 Multi-verse optimizer (MVO)

The MVO was proposed by Mirjalili (2015), and it uses the concept of multiverses and their elements – white holes, black holes and wormholes – to optimize different processes. In the MVO, the matrix given by Eq. (8) represents the universes (*U*), where

$$U = \begin{bmatrix} x_1^1 & \cdots & x_1^d \\ \vdots & \ddots & \vdots \\ x_n^1 & \cdots & x_n^d \end{bmatrix} \tag{8}$$

and d is the number of parameters (variables) and n is the number of universes (possible solutions). Eq. (9) shows the parameters for each universe, where x_i^j is the j-th parameter of the *i*-th universe, U_i is the *i*-th universe, $NI(U_i)$ is the normalized inflation rate of the *i*-th universe, $r1$ is a random number generated with uniform distribution in range [0, 1] and x_k^j is the *j*-th parameter of the *k*-th universe, where

$$x_i^j = \begin{cases} x_k^j & r1 < NI(Ui) \\ x_i^j & r1 \geq NI(Ui) \end{cases} \tag{9}$$

Equation (10) represents the mathematical model of the objects changes through the wormholes

$$x_i^j = f(x) = \begin{cases} \begin{cases} x_j + TDR * \left((ub_j - lb_j) * r4 + lb_j \right), & r3 < 0.5 \\ x_j - TDR * \left((ub_j - lb_j) * r4 + lb_j \right), & r3 \geq 0.5 \end{cases}, & r2 < WEP \\ x_i^j, & r2 \geq WEP \end{cases} \tag{10}$$

where x_j indicates the *j*-th parameter of the best universe, *TDR* (Traveling Distance Rate) is a coefficient, *WEP* (Wormhole Existence Probability) is another coefficient, lb_j is the lower limit of the *j*-th variable, ub_j is the upper limit of the *j*-th variable, x_i^j is the *j*-th parameter of the *i*-th universe and $r2$, $r3$ and $r4$ are random numbers in [0, 1].

Equations (11) and (12) present, respectively, the mathematical formulation of the coefficients *WEP* and *TD*, given by

$$WEP = \min + l * \left(\frac{\max - min}{L} \right) x_i^j = \begin{cases} x_k^j & r1 < NI(Ui) \\ x_i^j & r1 \geq NI(Ui) \end{cases} \tag{11}$$

$$TDR = 1 - \frac{l^{1/p}}{L^{1/p}} \tag{12}$$

where *min* is a constant (0.2 in the original paper of Mirjalili (2015)), *max* is another constant (with value 1 in the original publication), l is the current iteration, L is the maximum number of iterations and p is another constant that indicates the precision of the exploitation stage. The bigger is the value of p, the better is the performance of the local search. Figure 3 shows the pseudocode for the MVO.

```
Create random universes (U), Initialize WEP, TDR and Best_Universe
SU = Sorted universes, NI = Normalize the inflation rate (fitness) of the universes
while(t<Max number of iterations)
         Evaluate the fitness of all universes
            for i = 1 : n (all nuniverses)
                    Update WEP and TDR
            Black_hole_index = i
                       for j = 1 : m (mobjects)
                       r1 = random([0, 1])
            if (r2 < Wormholeexistanceprobability)
            r3 = random([0, 1]); r4 = random([0, 1])
            if(r3 < 0.5)
            U(i,j)=Best_Universe(j)+Travelling_distance_rate*((ub(j)-lb(j))*r4+lb(j))
                                 else
            U(i,j)=Best_Universe(j)-Travelling_distance_rate*((ub(j)-lb(j))*r4+lb(j))

            endif
                          end if
                       end for j
            end for i
end while
```

Figure 3. Multi-verseoptimizer pseudocode.

3.4 Grey wolf optimizer (GWO)

The GWO was proposed by Mirjalili *et al*. (2014), and it consists in an optimization metaheuristic algorithm based on the hunting behavior of the grey wolves. In order to mathematically model encircling behavior of the wolves, Eqs. (13) and (14) are proposed

$$\vec{D} = |\vec{C}.\overrightarrow{X_p}(t) - \vec{X}(t)| \tag{13}$$

$$\vec{X}(t+1) = \overrightarrow{X_p}(t) - \overrightarrow{A}.\vec{D} \tag{14}$$

where t indicates the current iteration, $\overrightarrow{X_p}$ indicates the position vector of the prey, $\vec{X}$ is the position vector of a grey wolf and the vectors $\vec{A}$ and $\vec{C}$ are coefficient vectors. The vectors $\overrightarrow{A}$ and $\vec{C}$ are calculated as shown in Eq. (15) and Eq. (16), respectively. In this case,

$$\vec{A} = 2\overrightarrow{a}.\overrightarrow{r_1} - \vec{a} \tag{15}$$

$$\vec{C} = 2\,.\overrightarrow{r_2} \tag{16}$$

where $\vec{a}$ represents a vector linearly decreased from 2 to 0 over the course of iterations and $\overrightarrow{r_1}$ e $\overrightarrow{r_2}$ are random vectors generated with uniform distribution in range [0 ,1].

The three best solutions of each iteration are saved as alpha, beta and delta, respectively, and oblige the other search agents to update their positions as stated by Eqs. (17)- (19). Figure 4 shows the pseudocode for the GWO.

$$\overrightarrow{D_\alpha} = \left|\overrightarrow{C_1}\,.\overrightarrow{X_\alpha} - \vec{X}\right|, \overrightarrow{D_\beta} = \left|\overrightarrow{C_2}\,.\overrightarrow{X_\beta} - \vec{X}\right|, \overrightarrow{D_\delta} = \left|\overrightarrow{C_3}\,.\overrightarrow{X_\delta} - \vec{X}\right| \tag{17}$$

$$\overrightarrow{X_1} = \overrightarrow{X_\alpha} - \overrightarrow{A_1}\,.\left(\overrightarrow{D_\alpha}\right), \overrightarrow{X_2} = \overrightarrow{X_\beta} - \overrightarrow{A_2}\,.\left(\overrightarrow{D_\beta}\right), \overrightarrow{X_3} = \overrightarrow{X_\delta} - \overrightarrow{A_3}\,.\left(\overrightarrow{D_\delta}\right) \tag{18}$$

$$\vec{X}(t+1) = \frac{\overrightarrow{X_1}+\overrightarrow{X_2}+\overrightarrow{X_3}}{3} \tag{19}$$

```
Initialize the grey wolf population X_i(i = 1,2, … , n)
Initialize a, A and C
Calculate the fitness of each agent
X_α = the best search agent
X_β = the second best search agent
X_δ = the third best search agent
while(t<Max number of iterations)

            for i = 1 : n (all n search agents)
                        Update the position of the current search agent
            end for i
            Update a, A and C
            Calculate the fitness of all search agents
            Update X_α, X_β and X_δ
            t=t+1
end while
return X_α
```

Figure 4. Grey wolf optimizer pseudocode.

3.5 Ant lion optimizer (ALO)

The ALO proposed by Mirjalili and Lewis (2015) is inspired by the interaction between the antlions and ants inside the antlion's trap. To be able to model this interaction, it's required the ants to move randomly in the search space so the antlions can hunt them and turn into better possible solutions along iterations. Equation (20) presents the mathematical model for stochastic behavior of the ants' moves in the nature. The mentioned equation is given by

$$X(t) = [0, cumsum(2r(t_1) - 1), cumsum(2r(t_2) - 1), \dots, cumsum(2r(t_n) - 1)] \tag{20}$$

where *cumsum* is the cumulative sum, *n* is the maximum number of iterations, *t* is the current iteration and *r(t)* is a stochastic function, represented by Eq. (21) given by

$$r(t) = \begin{cases} 1 \; if \; rand > 0.5 \\ 0 \; if \; rand \leq 0.5 \end{cases} \tag{21}$$

where *rand* is a random number generated uniformly distributed in range [0, 1].

The positions of the ants during the optimization process are stored along the iterations in a matrix M_{ant} represented by Eq. (22), where $A_{n,d}$ represents the value of the *d*th parameter of the *n*th ant.

$$M_{ant} = \begin{pmatrix} A_{1,1} & \cdots & A_{1,d} \\ \vdots & \ddots & \vdots \\ A_{n,1} & \cdots & A_{n,d} \end{pmatrix} \tag{22}$$

In order to evaluate the solution given by each ant, the objective function is used and the matrix M_{OA}, shown in Eq. (23), stores the fitness of all ants where

$$M_{OA} = \begin{pmatrix} f([A_{1,1}, & \cdots, & A_{1,d}]) \\ \vdots & \ddots & \vdots \\ f([A_{n,1} & \cdots, & A_{n,d}]) \end{pmatrix} \tag{23}$$

and $A_{n,d}$ represents the value of the *d*th parameter of the *n*th ant and f is the objective function. The same technique is used to store antlions data. The pseudocode for the ALO is shown in Fig. 5.

```
Initialize the first population of ants and antlions randomly
Calculate the fitness of ants and antlions
Find the best antlions and assume it as the elite (determined optimum)
while(t<Max number of iterations)

            for every ant
                        Select an antlion using Roulette wheel
                        Update c and d
                        Create a random walk and normalize it
                        Update the position of ant
            end for
            Calculate the fitness of all ants
            Replace na antlion with its corresponding ant if it becomes fitter
            Update elite if an antlion becomes fitter than the elite
end while
```

Figure 5. The Ant Lion Optimizer pseudocode.

3.6 Whale optimization algorithm (WOA)

Whale optimization algorithm is a metaheuristic optimization algorithm based on the hunting behavior of humpback whales, members of the Megapteranovaeangliae family, one of the largest whales in the oceans. The most interesting fact of these whales is precisely their method of hunting, called 'method of feeding the bubble net'.

In observations made by scientists, it has been observed that humpback whales surround their prey by creating air bubbles underwater in a spiral formed around their prey, preferably a set of krill and small fishes.

Humpback whales are able to recognize the position of the prey and close it. In the algorithm, since the optimization point (analogue to the prey) is not known a priori, the algorithm assumes that the current best fitness is the position of the prey or is minimally close to the optimal point. After identifying the optimization point - the point that has the least fitness - the other search agents update their positions according to this best agent.

In order to mathematically model the bubble network model, two approaches are required: Siege mechanism and the approach by spiral. The first one is represented by

$$\vec{D} = | \vec{C} \cdot \vec{X}^*(t) - \vec{X}(t) \tag{24}$$

where *t* indicates the iteration, *(X*)* vector indicates the position of the best solution obtained so far, *(X)* vector indicates the position of the whale and vector *C* represents coefficients.

The spiral approximation is obtained by the equation of a spiral, calculated after calculating the distance between the prey and the whale. This approximation is represented by

$$\vec{X}(t+1) = \overrightarrow{D'} \cdot e^{bl} \cdot \cos(2\pi l) + \vec{X}^*(t) \tag{25}$$

where *(D')* = | *X*(t)* - *X(t)* | and indicates the distance between the whale and the prey, *b* is a constant that defines the shape of the spiral and *l* is a random value generated using uniform distribution in [-1, 1].

The pseudo code of the WOA is detailed in Fig. 6.

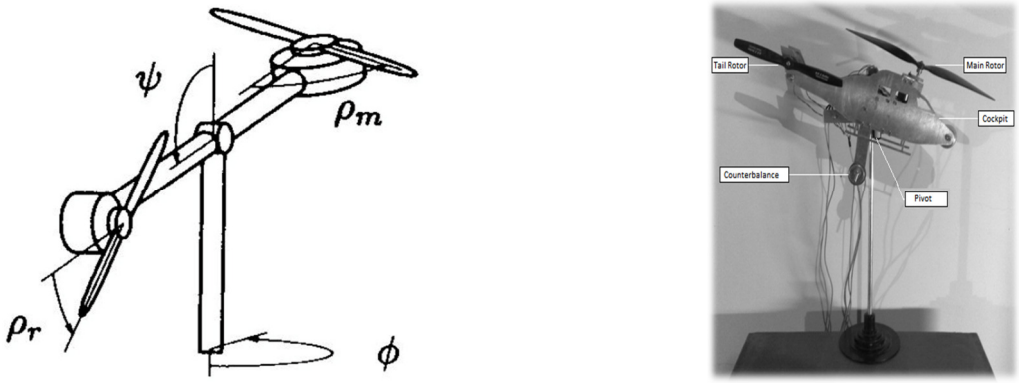
```
Initialize the whales population X_i(i = 1, 2, ..., n)
Calculate the fitness of each search agent
X* = the best search agent
while (t <Max number of iterations)
            for each search agent
                        if (p< 0.5)
                                    if (|A| < 1)
                                                Update the position of the current search agent
                                    else if (|A| ≥ 1)
                                                Select a random search agent (X_rand)
                                                Update the position of the current search agent
                                    end if
                        else if (p ≥ 0.5)
                                    Update the position of the current search
                        end if
            end for
            Check if any search agent goes beyond the search space and amend it
            Calculate the fitness of search agent
            Update X* if there is a better solution
            t=t+1
end while
return X*
```

Figure 6. WOA pseudocode.

4. DESCRIPTION OF THE CASE STUDY: THE EXPERIMENTAL TWIN-ROTOR SYSTEM

The twin-rotor is a laboratory-scale helicopter that simulates just a few of its movements. The device consists of a cockpit in the shape of a helicopter, where there are two propellers fixed to two brushed DC (Direct Current) motors (Pessoa, 2010). This whole structure is then attached to a rod which in turn is attached to a base, leaving only two degrees of freedom. In a helicopter the aerodynamic forces are controlled by the change in the angles of attack of the propellers of the rotors, in the twin rotor the angles of the propellers remain constant and the aerodynamic forces are controlled by the speed variations of the motors (Ahmad *et al*., 2000).

The process is a non-linear MIMO (Multiple Input Multiple Output) type system, where it has two input variables and two output variables. The inputs are the voltages applied to each motor, changing its angular velocity, while the outputs are the yaw angle Ψ and pitch angle φ (Ahmad *et al*., 2000) as represented in Fig 7(a). Figure 7(b) presents a photograph of the experimental twin-rotor system (Pessoa, 2010). The ρ_m is the angular velocity of the main rotor and the ρ_r is the angular velocity of the tail rotor.



(a) Representation of twin-rotor system (b) Photograph of the experimental twin-rotor

Figure 7. Twin-rotor system and its variables.

4.1 Acquisition and actuator system

The acquisition and actuator modules consist of the following components: a power system, a development board, a general purpose computer and the process. The development board used was an Arduino UNO that uses an Atmel ATmega328P microcontroller. It's an 8-bit RISC that works at 16 MHz and relies on 32 KB of flash memory, 8 and 16-bit timers, 6 PWM (Pulse Width Modulation) outputs, A/D (Analog-to-Digital) converter, 14 digital and 6 analog I/O (Input/Output) pins.

The power system is composed by a low-pass RC (resistor-capacitor) filter used to convert the PWM signal from the Arduino board into a voltage signal, an amplification circuit and a push-pull circuit used to source the high current needed by the twin rotor. Figure 8 shows the complete diagram of the system.

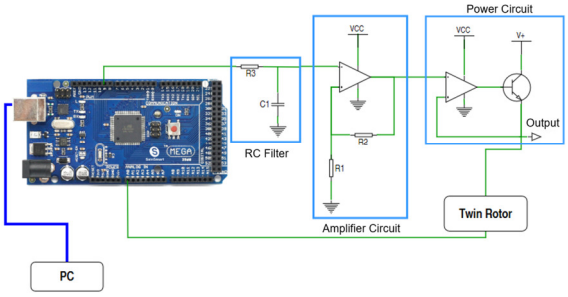


Figure 8. Diagram of the complete system.

4.2 Acquisition and Identification of the Twin-rotor System

To identify the twin-rotor system dynamic behavior, it was necessary to acquire both motors. Although the plant is a MIMO type system, it has been treated as a SISO (Single Input Single Output) system, where each motor is treated independently.

Because the dynamics and the process's physical limiter are different for each motor, the acquisitions were made in different ways. For the tail motor, the main motor remained on causing the twin-rotor to remain at a 0-degree angle. As a consequence, the main motor causes the Twin-rotor to rotate in the direction opposite to the direction of rotation caused by the tail motor. This methodology was adopted to more coherently simulate the dynamics of a helicopter. For this same reason, the tail motor only rotates in a certain direction.

As can be observed by Fig. 9, the main motor generates a disturbance in the tail movement, causing the tail to turn to the other side when a tail motor deceleration occurs. For the identification of the dynamics of both motors, due to their non-linear behavior, a genetic programming approach (Madar *et al.*, 2005) was used through MATLAB environment. The algorithm used determines the model according to complexity and the MSE (Mean Squared Error).

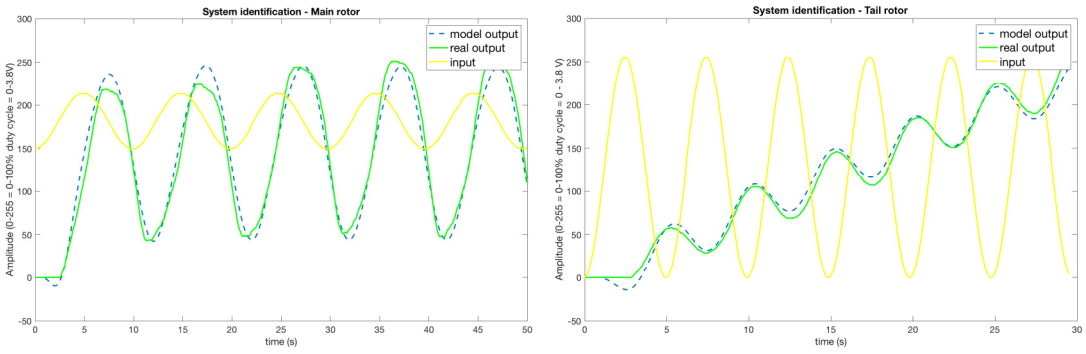


Figure 9. Acquisition and identification of main motor and tail motor.

The mathematical models found for the main motor and the tail motor achieved 93% and 94% of fitness, where the fitness is based on R^2 (coefficient of determination) value using the calculated and the measured output values.

5. RESULTS ANALYSIS

The next subsections show the results obtained for each optimization metaheuristic using the ITAE (Integral of Time-weighted Absolute Error) index as objective function (minimization problem) and the closed-loop control system response with implemented the best control parameters gains found.

5.1 The adopted performance index (ITAE)

The optimization metaheuristics were applied and compared in PID tuning applied to the twin-rotor system. The problem is to find a configuration of the gains of the PID controller that minimizes the ITAE. In this context, the optimization problem is related to ITAE index minimization as the objective function when the PID controller acts in closed-loop in twin-rotor system. Tables 1 and 2 show the optimization results of the PID for the main and tail rotor, respectively. The expression for the ITAE index (minimization problem) is given by

$$ITAE = \int_0^\infty t|e(t)|dt \tag{26}$$

where t is the time and $e(t)$ is the difference between set point and the value controlled variable.

Table 1. Results of the optimization metaheuristic in the PID tuning to the main rotor of the twin-rotor system (best result is in *italics*).

Optimization metaheuristics	Proportional gain (K_p)	Integral gain (K_i)	Derivative gain (K_d)	Performance index, ITAE
GWO	0.00126	0.00294	0.00193	3958
MVO	0.00100	0.00100	0.81088	4084
ALO	*0.00105*	*0.00251*	*0.00110*	*2764*
WOA	-	-	-	unstable behavior(1)
HS	-	-	-	unstable behavior(1)
FA	-	-	-	unstable behavior(1)

(1) results with unstable behavior of the closed-loop system.

Table 2. Results of the optimization metaheuristic in the PID tuning to the tail rotor of the twin-rotor system (best result is in *italics*).

Optimization metaheuristics	Proportional gain (K_p)	Integral gain (K_i)	Derivative gain (K_d)	Performance index, ITAE
GWO	*1.7950*	*0.001054*	*1.1406*	*500.56*
MVO	1.2613	0.001	1.2119	520.98
ALO	0.38095	0.001	1.2742	560.77
WOA	0.011186	0.001	4.2237	731.71
HS	0.005850	3.1965	5.9183	1143.00
FA	0.024036	10.81	1.3646	758.71

5.2 Results of closed-loop control

The closed-loop system was implemented using gains obtained by the ALO, the best result shown in Table 1. Figure 10 shows the responses obtained for random inputs given to the twin-rotor system.

The implemented controllers for the tail rotor using all the results shown in Table 2 were unstable, the current hypothesis for the cause of this problem is that the obtained model was not valid.

6. CONCLUSION AND FUTURE RESEARCH

The proposed PID design method based on optimization metaheuristics is intuitive and practical that offers an effective and simple way to tune the gains of the controller. In this context, the ALO approach provides good tracking performance to the PID design when validated to different reference (setpoint) signals and when compared with the performance of the other tested optimizers. For the tail rotor, however, the results show that the acquisition and identification process could be improved.

As a future research proposition, this work could be extended to compare other different intelligent approaches, such as artificial neural networks, fuzzy systems and hybrid intelligent systems, applied to the nonlinear identification, optimization and control tasks.

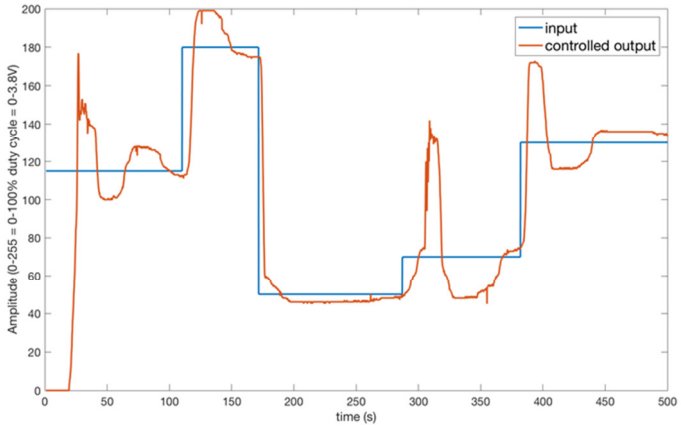


Figure 10. Best result (servo behavior) using PID tuning using ALO applied to twin-rotor system (control of the main motor).

7. ACKNOWLEDGEMENTS

The authors would like to thank National Council of Scientific and Technologic Development of Brazil - CNPq (grants: 404659/2016-0 and 303908/2015-7-PQ), and "Fundação Araucária" (grant number: 117/2014) for the financial support of this work.

8. REFERENCES

Abdelgayed, T.S.,Morsi, W.G. and Sidhu, T.S., 2017. "A new harmony search approach for optimal wavelets applied to fault classification". *IEEE Transactions on Smart Grid*, 2017 (in press).

Ahmad, S.M.,Chipperfield, A.J. and Tokhi, M.O., 2000. "Dynamic modelling and control of a 2-DOF twin rotor multi-input multi-output system". In *Proceedings of the 26th Annual Conference of the IEEE Industrial Electronics Society* (IECON), Nagoya, Japan,Vol. 2, p. 1451-1456.

Aljarah, I., Faris, H.m. and Mirjalili, S., 2017. "Optimizing connection weights in neural networks using the whale optimization algorithm". *Soft Computing*, p. 1-15 (in press).

Butt, S.S. and Aschemann, H., 2015. "Multi-variable integral sliding mode control of a two degrees of freedom helicopter". *IFAC-PapersOnLine*, Vol. 48, p. 802-807.

Hunaini, F., Robandi, I. and Sutantra, N, 2016. "Lateral and yaw motion control of the vehicle using fuzzy logic and PID being optimized by firefly algorithm". *Journal of Theoretical and Applied Information Technology*, Vol. 87, p. 16, 2016.

Levine, W.S., 1996. *The Control Handbook*. Piscataway, NJ, USA: CRC Press and IEEE Press.

Madar, J., Abonyi, J. and Szeifert, F., 2005. "Genetic programming for the identification of nonlinear input-output models". *Industrial & Engineering Chemistry Research*, Vol. 44, p. 3178-3186.

Mirjalili, S., Mirjalili, S.M. and Lewis A., 2014."Grey wolf optimizer". *Advances in Engineering Software*, Vol. 69, p. 46-61.

Mirjalili, S., Mirjalili, S.M.and Lewis A., 2015. "Multi-verse optimizer: a nature-inspired algorithm for global optimization". *Neural Computing & Applications*, Vol. 27, p. 495-513.

Ogata, K., 2012. *Engenharia de Controle Moderno*. 5th ed. São Paulo, SP: Prentice-Hall do Brasil (in Portuguese).

Pessoa, M. W., 2010. *Identificação de sistemas não-lineares utilizando modelo polinomial NARMAX e nebuloso Takagi-Sugeno-Kang*. Master thesis, Industrial and Systems Engineering Graduate Program (PPGEPS), Pontifical Catholic University of Parana (PUCPR), Curitiba, Brazil (in Portuguese).

Salcedo-Sanz, S., 2016. "Modern meta-heuristics based on nonlinear physics processes: areview of models and design procedures". *Physics Reports*, Vol. 655, p. 1-70.

Yang, X.-S., 2009. "Firefly algorithms for multimodal optimization". In *Stochastic Algorithms: Foundations and Applications* (SAGA), Lecture Notes in Computer Sciences, Vol. 5792, p. 169-178.

Yang, X.-S., 2010. *Metaheuristic Algorithms*. 2nd Ed. Frome: Luniver Press.

9. RESPONSIBILITY NOTICE

The authors are the only responsible for the printed material included in this paper.