Advances in Computer Science and its Applications
Vol. 1, No. 1, March 2012
Copyright © World Science Publisher, United States
www.worldsciencepublisher.org

State Feedback Design Aircraft Landing System with Using Differential Evolution Algorithm

S. Amir Ghoreishi[1] and Arash Ahmadivand[2]

[1] Department of Engineering, Electrical Engineering Faculty, South Tehran Branch, Islamic Azad University, Tehran, Iran
[2] Departments of Electrical Engineering, Ahar Branch, Islamic Azad University, Ahar, Iran
Email: a_ahmadivand@mailfa.org

Abstract – The main goal of this article is select the appropriate weight matrices to design the state feedback control LQR of the aircraft landing system. As we know, select the suitable weight matrices for optimal control system is not simple and so far systematic method to determine the appropriate weight matrices are not provided. In other words, there is no direct link between the weights and the optimal profile control system and select weight matrices using the trial and error method and based on designer's experiences are done. In this paper, we use Differential Evolution Algorithm technique (DE) to determine the weight matrices. Benefits of proposed method can be cited profile sustainable convergence and high computational speed. Simulation results show that comparison with Imperialist Algorithm (ICA), method of differential evolution algorithm in determining the appropriate weight matrices and optimal controller design (LQR) is very strong.

Keywords – Linear Quadratic Regulator (LQR); Differential Evolution Algorithm; Weight matrices; Imperialist Competitive Algorithm (ICA).

1. Introduction

In most of systems in order to designing and solving many issues, we need to select an answer as the optimal response from a wide range of possible answers but because of large extent answers collection, virtually all the answers can not be tested and this test should be done randomly. On the other hand, the random process must be done in such a way that will converge towards the best answer. Because linear quadratic optimal control theory is easily implemented in engineering problems and it is the base of other control theories, it has special importance [1]. Nevertheless, in certain cases that the cost function is a linear quadratic function optimal response is convergent to the linear quadratic regulator response. LQR technique has been used widely in areas such as control of induction motors and controlling the vehicle crank [1, 2]. About the issue of choosing the appropriate weight matrices in the controller design various methods have been suggested. For the first time Kalman [3] has presented a way to determine the weight matrices based on the given poles and Wang [4]. Recently, many attempts have been done to design the controller using genetic algorithms colonial competition [5, 6 and 7]. Fundamental issue is that we determine the best weight matrices that meet optimal condition monitoring system in the possible minimum time. In this article, recommended using Differential Evolution Algorithm to determine the weight matrices and will show that results meet monitoring system requirements and desired system specifications and will check the of advantages of the method over Imperialist Competitive Algorithm.

This paper is organized as follows: In Section II, we examined linear quadratic optimal control problem. In section III, introduced tools to optimize deal that is including review and introduce Differential Evolution Algorithm and Imperialist Competitive Algorithm and the laws governing these algorithms. In section IV, we examined usage of the Differential Evolution Algorithm for state feedback design and determine the appropriate weight matrices in the optimal controller. In section V, simulation results using the Differential Evolution Algorithm with the results of Imperialist Competitive Algorithm method are compared. Conclusions form the final section of the article.

2. LQR Controller

One of the state space based optimal control method is Linear Quadratic Regulator (LQR). In this section we briefly describe this method. Consider the following linear, continuous-time and controllable system:

$$\dot{x} = Ax + Bu \tag{1}$$

The following objective function is defined:

$$J = \frac{1}{2}\int_0^\infty [x^T Qx + u^T Ru]\,dt \tag{2}$$

Where, Q and R are weighting matrices and should be positive-semi-definite and positive-definite, respectively. Since system (1) is controllable, the method which is able to minimize (2) is called LQR. Considering the functional (2) in LQR the following Riccati equation should be solved:

$$PA + A^T P - PBR^{-1}B^T P + Q = 0 \tag{3}$$

By solving the above Riccati equation the positive-definite matrix P is obtained, thus the optimal gain and controller are calculated as:

$$K = R^{-1}B^T P \tag{4}$$

$$u = -Kx \tag{5}$$

Therefore, the closed-loop poles are the eigenvalues of $A - BK$. In this paper the eigenvalues of $A - BK$ are shown *as* $\lambda = \{\lambda_1, \lambda_2, \ldots, \lambda_n\}$.

In an LQR problem, the weighting matrices which are Q and R, demonstrate profound effect on the performance of controller. On the other hand, finding the best Q and R needs many computer simulation and trial and errors, which are very time-consuming. Thus using optimization methods for finding Q and R is more effective. In this paper, we use Genetic Algorithm (GA) and Particle Swarm Optimization (PSO) as two optimization tools. Firstly, in the next section we are going to describe the optimization problem for finding Q and R.

3. Optimization Tools

3.1 Deferential Evolution Algorithm

Price and Storn proposed Differential Evolution (DE) in mid 1990s [8, 9], to deal with optimization problems, defined in continuous domains. DE has similarities with both GA and PSO. DE uses information of all individuals, and differences between them, to create new solutions for optimization problem. The new solutions are created using difference and trial vectors. To create a new solution y, an old solution a is perturbed using the following rule:

$$y = a + F \otimes (b - c) \tag{6}$$

Where b and c are two individuals, randomly selected from population, and $a \neq b \neq c$. Vector F is the scaling factor, and its elements are uniformly distributed random numbers in $[F_{\min}, F_{\max}]$. Operator $\otimes$ is the element-wise multiplication operator. To create final solution z, crossover operator is applied to y and another randomly selected individual x. There are various methods of crossover. Simplest case is formulated as follows:

$$z_i = \begin{cases} y_i, & r \leq CR \ \text{ or } \ i = i_0 \\ x_i, & \text{otherwise} \end{cases} \tag{7}$$

Where x_i indicates i-th element of vector x, scalar r is a uniformly distributed random number in [0, 1], CR is the Crossover Rate parameter, and i_0 is a random integer index in the set $\{1, 2, 3, ..., n\}$. It is assumed that numbers of search space dimensions (also the number of elements of solution vectors) are equal to n. In this way, DE uses information of current population, to create individuals of the next iteration (generation). This process is carried out, until termination conditions are satisfied.

3.2. Imperialist Competitive Algorithm

Imperialist Competitive Algorithm (ICA) is proposed by Atashpaz-Gargari et al. [10, 11], and it is inspired by imperialist competition. Similar to GA, which simulates the natural evolution process to solve an optimization problem, ICA simulates the socio-political evolution to deal with optimization problems. In ICA, individual solutions are referred as (virtual) countries. Some of good countries in the initialization phase, which are named imperialists, form their own imperial. They capture their colonies from other non-imperialist (normal) countries. In every the iteration (decade) of ICA, the following operations are carried out:

3.2.1 Assimilation of Colonies

Colonies of each imperialist are assimilated to their respective imperialist. Assimilation is formulated as following:

$$x_{col}^{new} = x_{col}^{old} + \beta \mathbf{r} \otimes (x_{imp} - x_{col}^{old}) \tag{8}$$

where β is assimilation factor, and $\mathbf{r}$ is a vector, and its elements are uniformly distributed random numbers in $[0,1]$. x_{imp}, x_{col}^{old} and x_{col}^{new} are position of imperialist, old position of colony, new position of colony, respectively. In [10, 11], the new position of colony is angularly deviated. For more information about deviation, refer to [10] and [11].

3.2.2 Revolution of Colonies

Similar to mutation operator in GA, selected colonies of every imperialist are changed randomly, or revolved. Revolution is applied to a colony, with a probability of p_r.

3.2.3 Exchange with Best Colony

If after assimilation and revolution steps, there are colonies which are better than their respective imperialists, the imperialist is exchanged with its best colony. In other words, imperialist will be colony, and the best colony will be the new imperialist.

3.2.4 Imperialist Competition

Weakest imperialist among others, loses its weakest colony. One of other imperialists will capture the lost colony, randomly. The better the imperial, the more probable it will possess the colony. An imperialist without colony will collapse. It will become a colony, and captured by other imperialists. The mentioned steps are carried out, while stop conditions are not satisfied.

4. Using DE to Determine the Appropriate Weight Matrices

For finding the appropriate weight matrices, we will do as follow: First we will get the elements of weight matrices randomly in the defined limit and we will calculate the amount of controller's gain through LQR commands with Matlab software. We must pay attention that the two conditions of $\det Q \geq 0, \det R \geq 0$ should always exist [12-13]. Then we will calculate the control energy matrix and also the state variable of $x(t)$.Furthermore, we calculate the correlations of cost (J) and the total of error absolute value. For every answer of in the main population which is known as x. We will store the minimum amount of answers cost correlations and minimum amount of errors also put the condition for stopping algorithm will be when it reaches the number of pre-defined cycles.

1) Three answers from the members of main population will be chosen randomly by the names of a, b, c .The three given answers are not equal to each other or to x .

$$a \neq b \neq c \neq x \tag{9}$$

2) The new answer which is shown by y will be achieved through mutation.

$$\begin{gathered} y = (y_1, y_2, \dots, y_n) \\ y_i = a_i + \sigma_i (b_i - c_i) \end{gathered} \tag{10}$$

That σ_i is fixed coefficient with monotonous in $[0.5,1.5]$.

3) Another new answer will be achieved through cross action with probability of $P_{cr} = 0.3$ and it will be shown with x'.

$$x' = (x'_1, x'_2, x'_3, \dots, x'_n)$$

$$x'_i = \begin{cases} y_i & r \leq p_{cr} \quad or \quad i = i_0 \\ x_i & otherwise \end{cases} \tag{11}$$

In the before formula x_i is related to main answer and y_i is related to the new answer and x'_i is a combination of main answer and new answer. $i_0 \in \{i_1, i_2, \dots, i_n\}$ is the ideal index and it is chosen randomly so that at least in one case x , x' are different.

4) The answer of x' is evaluated in accordance with control index .If x' is better than x, and then we will replace x with x'. Other wise we will ignore x'.

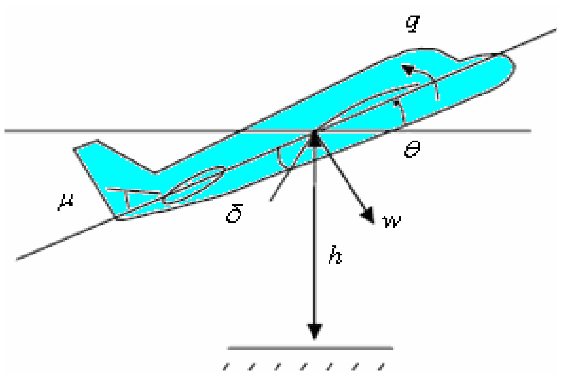

Figure 1. Aircraft Landing System

5. Simulation Results

We consider the landing system of aircraft as a complicated system with 6 sate variables [5]. The first figure shows this model. The illustration of state apace of this system is presented in formula (12).

$$\begin{bmatrix} \dot{u} \\ \dot{w} \\ q \\ \dot{\theta} \\ \dot{h} \\ \dot{e} \end{bmatrix} = \begin{bmatrix} -0.058 & 0.065 & 0 & -0.171 & 0 & 1 \\ -0.303 & -0.685 & 1.109 & 0 & 0 & 0 \\ 0.072 & -0.685 & -0.947 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & -1 & 0 & 1.133 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & -0.571 \end{bmatrix} \begin{bmatrix} u \\ w \\ q \\ \theta \\ h \\ e \end{bmatrix} + \begin{bmatrix} 0 & 0 & -0.119 \\ -0.054 & 0 & 0.074 \\ -1.117 & 0 & 0.115 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0.571 & 0 \end{bmatrix} \begin{bmatrix} \mu \\ \gamma \\ \delta \end{bmatrix} \tag{12}$$

So u is the propelling speed of the aircraft (ms^{-1}) , w is down ward speed (ms^{-1}) , q is the pitch speed to the ground ($\deg rees^{-1}$) , θ is pitch angle to the ground ($\deg rees$), h is the altitude (m) , e is propelling velocity (ms^{-2}), μ is elevator angle ($\deg rees$), γ is throttle value (ms^{-2}), δ is spoiler angle ($\deg rees$). Input of $(\mu \ \gamma \ \ \delta)^T$ must be designed in a way that the aircraft lands with illustrated way as shown in the following equation.

$$\dot{h} + 0.2h = 0 \tag{13}$$

The control index is that the amount of absolute value integral of error $IAE = \int_0^{tf} |e| dt$ is minimum. The first amounts of system state are as follow:

$$\begin{aligned} x(0) &= [u(0) \ w(0) \ q(0) \ \theta(0) \ h(0) \ e(0)]^T \\ &= [5 \quad -2.5 \quad -1 \quad -3 \quad 15 \quad 0.5]^T \end{aligned}$$

From equation (12) we have $\dot{h} = -w + 1.133\theta$. With placing this equation in equation (13) we will have:

$$\int_0^{30} |-w + 1.133\theta + 0.2h| dt = \int_0^{30} |-x_2 + 1.133x_4 + 0.2x_5| dt \tag{14}$$

In a way that $h = x_5, \theta = x_4, w = x_2$.

The design pattern is that we first choose Q, R matrices. In second step the equation of (1) and (3) will be solved through computation and the simulated results will define if the limitation of the system is solved or not. If the limitations of control system have not been considered, the weight matrices are chosen again and process goes on. Based on above-mentioned algorithm choosing suitable weight matrices are very difficult and with trial and error it takes a lot time. One of the result trial and errors is brought to you as follow:

$$Q = \begin{bmatrix} 0.618 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0.073 & 0 & -0.129 & -0.8 & 0 \\ 0 & 0 & 0.484 & 0 & 0 & 0 \\ 0 & -0.129 & 1 & 0.055 & 0.667 & 0 \\ 0 & -0.8 & 0 & 0.667 & 0.054 & 0 \\ 0 & 0 & 0 & 0 & 0.798 & 0 \end{bmatrix}$$

$$R = \begin{bmatrix} 0.621 & 0 & 0 \\ 0 & 0.421 & 0 \\ 0 & 0 & 0.143 \end{bmatrix}$$

And feedback matrix is calculated as follow:

$$K = \begin{bmatrix} -0.3477 & 0.805 & -0.8403 & -1.6266 & -0.1734 & -0.1620 \\ 1.0778 & -0.2552 & 0.1345 & 0.4379 & 0.1669 & 1.5443 \\ -1.0942 & 0.4669 & -0.0395 & -0.8032 & -0.4046 & -0.679 \end{bmatrix}$$

Table 1. Comparison Integral Amount of Error Absolute Value for Pursing Aircraft Desired Path

Trial and error	ICA	DE	Air craft Desired path
46.198	22.031	10.068	IAE

The result of aircraft landing simulation is in figure 2. The amount of absolute value integral is achieved through trial and error 46.198.

Analysis of the results achieved through implementation of DE algorithm and ICA algorithm: Parameters used in ICA algorithm are as: Population size of country 50, cross rate 30%, the minimum amount of adaptability coefficient 1.25, the cost of exploited colonies through kingdom 0.1, revolution percentage 10, search limitation $[0,20]$ and the number of iteration 50 are considered. Parameters used in DE algorithm are as: Population size of country 50, cross rate 30%, the minimum amount of scale coefficient 0.5, the maximum amount of scale coefficient 1.5, search limitation $[0,20]$ and the number of iteration 50 are considered. The amount of absolute value integral error is presented in table 1 along with implementation of all presented ways. Figures 3 and 4 show input graph for spoiler angle and elevator angle with the use of different ways. With regard to the results in table 2 we see that the maximum control force will be decreased significantly through the use of differential evolution algorithm. Figure 2 shows that by defining weight matrices through differential evolution algorithm, the aircraft will land in the predefined way. Figures 3 and 4 also show that in this case performance of control system will be improved.

Table 2. Comparison of Maximum Amount for Control input with Different Methods.

Spoiler angle	Elevator angle	Throttle	Maximum amount
7.991	19.558	17.557	Trial and error
3.556	14.895	13.893	ICA
2.795	9.535	6.358	DE

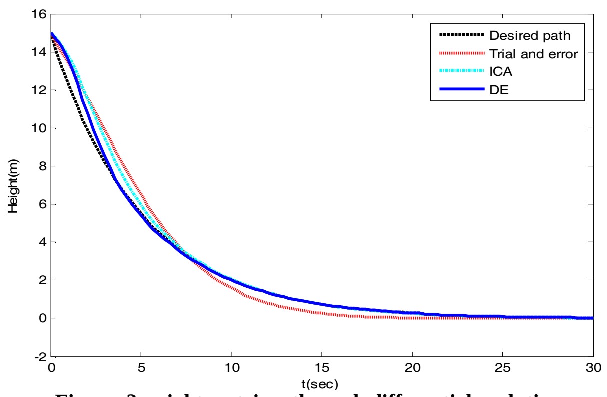

Figure. 2 weight matrices through differential evolution algorithm.

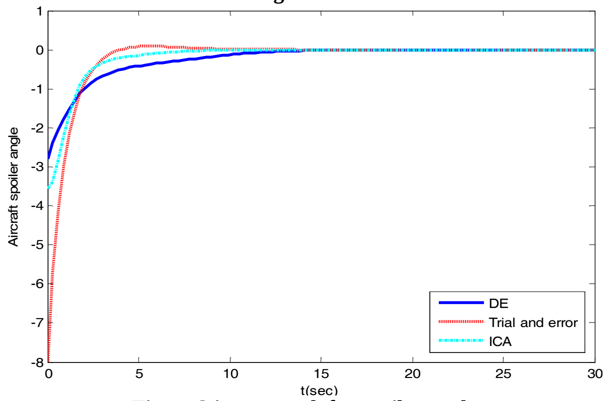

Figure.3 input graph for spoiler angle

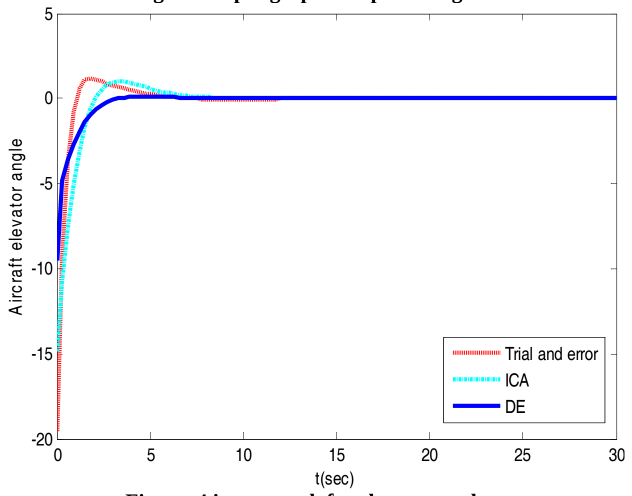

Figure. 4 input graph for elevator angle

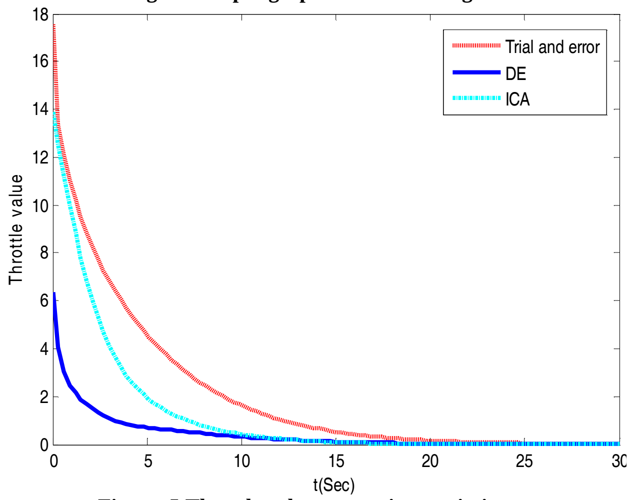

Figure. 5 Throttle value verses time variations

5. Conclusions

This paper offers a method for determining weight matrices for optimal control system design using differential evolution algorithm. Characteristics of the algorithm presented can be cited high computational speed, High success rate algorithm and faster convergence. Simulation results in comparison with previous experiments are very satisfactory and we see that responses obtained using this method has better performance.

References

[1] Athens, M., “The status of optimal control theory and applications for deterministic systems,” IEEE Trans. Automatic Control, 1966.

[2] Kirk, Donald E., Optimal control theory: an introduction, Carmel/ California, 1937.

[3] Kalman, R. E., “When is a linear control system optimal?,” J. Basci Eng. Trans., ASME-86D,1964.

[4] Wang Yaoqing, “The determination of weighting matrices in LQ optimal control system,” ACTAAutomatic Sinica, 1992.

[5] Gheng-Chung Sung, Gong Chen, “Optimal control systems design associated with geneticalgorithm,” Proceedings of 26 CACS Automatic Control Conference St. John's University, Tamsui, Taiwan, 2006.

[6] C. P. Bottura and J. V. da Fonseca Neto, “Rule based decision making unit for eigenstructureassignment via parallel genetic algorithm and LQR design,” Proceeding of the American Control Conference, Chicago, Illinois, 2000.

[7] C. P. Bottura and J. V. da Fonseca Neto, “Parallel eigenstructure assignment via LQRdesign and genetic algorithms,” Proceeding of the American Control Conference, San Diego, 1999.

[8] R. Storn and K. Price, “Differential Evolution: A Simple and Efficient Adaptive Scheme for Global Optimization over Continuous Spaces,” Technical Report TR-95-012, International Computer Science Institute, Berkeley, California, March 1995.

[9] R. Storn and K. Price, “Differential Evolution – A Fast and Efficient Heuristic for Global Optimization over Continuous Spaces,” *Journal of Global Optimization*, vol. 11, pp. 341–359, 1997.

[10] E. Atashpaz-Gargari and C. Lucas, “Imperialist Competitive Algorithm: An algorithm for optimization inspired by imperialistic competition,” *Proceedings of IEEE Congress on Evolutionary Computation 2007 (CEC2007)*, Singapore, 2007.

[11] E. Atashpaz-Gargari, F. Hashemzadeh, R. Rajabioun, and C. Lucas “Colonial Competitive Algorithm: a novel approach for PID controller design in MIMO distillation column process,” *Int. J. Intell. Comput, Cybernet.*, vol. 1, no. 3, pp. 337-355, 2008.

[12] Y. Zhang and L. Wu. “A Robust Hybrid Restarted Simulated Annealing Particle Swarm Optimization Technique”, *Advances in Computer Science and its Applications*, vol. 1, no. 1, pp. 5-8, 2012.

[13] A. Ahmadivand, S. Farhangi, “A Comparative Study of IS-IS and IGRP Protocols for Real-Time Application Based on OPNET”, *Advances in Computer Science and its Applications*, vol. 1, no. 1, pp. 9-15, 2012.