Received: September 17, 2020. Revised: October 19, 2020. 157

International Journal of
Intelligent Engineering & Systems

http://www.inass.org/

An Improvement in LQR Controller Design based on Modified Chaotic Particle Swarm Optimization and Model Order Reduction

Hadeel N. Abdullah[1]*

[1]*Electrical Engineering Department, University of Technology, Iraq*
* Corresponding author's Email: Hadeel.n.abdullah@uotechnology.edu.iq

Abstract: The diverse engineering and scientific applications are stated through complex and high-order systems. The significant difficulties of these systems are the complications of modeling, analyzing, and controlling. It is easier to examine simpler models for more physical insights than more complex models and result in lower-ordering controllers that are easier to implement. The model order reduction (MOR) was used to simplify the computational difficulty of such complications and was later developed intensively for use with increasingly CDS. In this paper, a new Modified Chaos Particle Swarm Optimization (MCPSO) technique is employed to get a reduced-order model of a large scale system and design a Linear Quadratic Regulator (LQR) based controller. The mod uses the combination of advantages of basic PSO algorithms and chaotic algorithms. It becomes an excellent algorithm with fast convergence, few control parameters, simple execution, and avoidance of local extremes. In addition to combining the chaotic algorithm, CPSO also improved the weight parameter w, adjusting it to the dynamic attenuation direction. First, efficient reduced-order model parameters are obtained for original higher-order systems based on the MCPSO. Then linear quadratic regulator (LQR (controller parameters optimized for the reduced-order model. The goodness of the proposed method is evaluated through a numerical example. The experimental results indicate that the proposed technique's reduced order model provides an excellent close approximation to the original system.

Keywords: Model order reduction, Modified PSO, Linear quadratic regulator, Chaos optimization.

1. Introduction

The cost and complexity of the control unit increase as the order of the system was increases. This problem can be overcome if the low-order model is available to the original high-order system. If a controller can be designed using the lower order model, the original high-order system will be improved when placed in the closed-loop. To reduce cost and time in design and simplify implementation, reduced-order models (ROM) are very suitable for engineers in analysis, synthesis, and simulation. Different techniques are useful in the literature for MOR [1–3].

Several methods commonly used in MOR do not guarantee the stability of the ROM. Through a post-processing step, ROM is controlled to ensure stability while reinforce/maintaining its accuracy using the limiting nonlinear lease-square minimization problem.

Particle Swarm Optimization (PSO) algorithm is a bionic optimization algorithm based on bird and fish swarm foraging process [4]. The PSO algorithm has the following features: (1) fast convergence speed, the algorithm can get search outcomes fast; (2) few control parameters are required, the algorithm is direct and straightforward to implement; (3) certain parallelism, PSO operations are applied to a group of particles. Despite these advantages, the PSO algorithm also has the same problem as the genetic algorithm: it is simple to place down into the optimal local value, which is very unfavorable for the multi-peak function to solve the maximum global value. These optimization problems are very common in real life. Therefore, when the PSO algorithm is applied, a certain degree of improvement needs to be

made to avoid its early phenomenon and fall into the optimal local value.

Various investigations have been carried out, and different methods have been proposed to simplify the higher-order transfer function [5-6]. Each method has its advantages and limitations. The most important concerns among the disadvantages are the tedious computational procedures or shifting and maintaining stability in the reduced model.

Chaos optimization (CO) algorithm is a global optimization method, which employs numerical sequences generated using chaotic maps. Chaos is a random motion state obtained from certain equations [7]. The CO algorithm uses chaos equations to iteratively generate random sequences for random traversal search. The characteristics of the chaotic algorithm are: (1) randomness, the chaotic variables generated by the algorithm are as messy as random variables on the surface; (2) traversability, the random sequence generated by the algorithm can traverse all points in the search area without repeating; (3) Regularity, the algorithm is controlled by the determined iterative Eq. (4) It is very sensitive to the initial value, and slight changes in the initial conditions will cause huge changes in the system behavior . In all of the hybrid approaches, CO Algorithm takes a lot of computational effort to ultimately reach the optimal solution. This is due to the fact that the ability to search for local CO Algorithms is generally poor. The movement steps of chaotic variables between two successive iterations are usually large, which leads to solutions that jump out into the search space [8].

The Chaos Particle Swarm Optimization (CPSO) algorithm combines the fast convergence of the PSO algorithm and the traversal randomness of the CO algorithm [9]. In the area near the optimal solution selected by each generation of the PSO algorithm, the chaotic algorithm is used to further search, prevent it from falling into the optimal local value, thereby improving the PSO algorithm's and becoming an efficient optimization algorithm.

In [10], a new hybrid model combining ANN and CPSO has been proposed to improve forecast accuracy. The results proved that the proposed model works better for fine particles compared to coarse particles. Zhu, Zheng, and Ma, in [11], presented an evolutionary approach to solving optimal power system problem of electric vehicle. In [12], proposed a new algorithm to improve the chaotic CPSO, to solve combinatorial optimization problems. Empirical results show that CS-PSO has better variety, frequency, and efficacy than healthy lifestyle recommendations (HLR-PSO).

One of the advantages of using the LQR is that it facilitates the design and increases case variables by estimating the situation. The main advantage of the LQR control relative to pole placement is that instead of determining where similar values should be placed, a range of performance weight may be identified that may have a more attractive appeal. The result is stable control guaranteed [13,14].

Control aspects of large scale systems (models with very high order) are a major concern in the field of control systems. The designed controller's order must be close to the large- scale system order, or even more in most cases. As the controller's order increases the control aspects of the system, it becomes even more complicated. Many model order reduction techniques that reduce the higher-order system's order without losing the predominant characteristics. A linear quadratic regulator based design is an optimization tool to derive an optimal controller by minimizing the cost function based on the two weighting matrices Q and R, which weigh the state vector and the system input. The main feature of this paper aims to:

- Apply model order reduction on a large scale system based on a new combination between the advantages of basic PSO and chaotic algorithms. Also, made improvements to the weight coefficient w, of the proposed Chaos Particle Swarm Optimization (CPSO)
- Design a Linear Quadratic Regulator (LQR) based controller to analyze the performance indices in the time and frequency domains.
- The step and frequency responses of the system with the LQR controller are simulated in MatLab.
- A single-input-single-output system (SISO) is considered, due to the LQR controller's compatibility and state-space equations. Also, this study extended to multi-input multi-output (MIMO) systems.

The rest of the paper deals with section 2 explain the Problem Formulation of the model order reduction. Section 3 is associated with the proposed MCPSO technique. Section 3 is also associated with the proposed design of the optimal LQR controller's proposed design based on the reduced-order model. Section 4 is associated with Numerical Example and Simulation Results, and section 5 with a conclusion.

2. Problem formulation

2.1 Reduced-order model

Consider $G_n(s)$ be the SISO transfer function of the linear time-invariant system of order 'n' represented by the form [2]:

$$G_n(s)=\frac{N_n(s)}{D_n(s)}=\frac{a_0+a_1s+a_2s^2+...+a_{n-1}s^{n-1}}{b_0+b_1s+b_2s^2+...+b_ns^n} \quad (1)$$

Where *a, b: 0 ≤ i ≤ n-1* are known scalar constants. The objective is to find the r^{th} *(r < n)* order ROM $G_r(s)$ represented by the form:

$$G_r(s)=\frac{N_r(s)}{D_r(s)}=\frac{e_0+e_1s+e_2s^2+...+e_{r-1}s^{r-1}}{f_0+f_1s+f_2s^2+...+b_rs^r} \quad (2)$$

Where *e, f: 0 ≤ i ≤ n-1* are unknown scalar constants. The ISE between the responses of $G_r(s)$ and $G_n(s)$ systems are calculated to measure the goodness of the ROM: the lower the ISE, the closer $G_r(s)$ is to $G_n(s)$. ISE is given by [15]:

$$ISE=\sum_{i=0}^{n}\left[y(t_i)-y_r(t_i)\right]^2 \quad (3)$$

Where $y(t_i)$ and $y_r(t_i)$, are the responses of the original and reduced-order systems, respectively.
Consider $G_n(s)$ be the transfer function of a higher-order MIMO system. It is given by [2]:

$$G_n(s)=\frac{1}{D_n(s)}\begin{bmatrix} a_{11}(s) & a_{12}(s) & ... & a_{1p}(s) \\ a_{21}(s) & a_{22}(s) & ... & a_{2p}(s) \\ ... & ... & ... & ... \\ a_{m1}(s) & a_{m2}(s) & ... & a_{mp}(s) \end{bmatrix} \quad (4)$$

The general form of $g_i(s)$ from $Gn(s)$ can be taken as:

$$g_{ij}(s)=\frac{a_{ij}(s)}{D_n(s)} \quad (5)$$

Where *i=1,2, ..., p and j=1.2. ..., m*
The objective is to find the ($r^n<n$) ROM $G_r(s)$ represented in the form:

$$G_r(s)=\frac{1}{D_r(s)}\begin{bmatrix} e_{11}(s) & e_{12}(s) & ... & e_{1p}(s) \\ e_{21}(s) & e_{22}(s) & ... & e_{2p}(s) \\ ... & ... & ... & ... \\ e_{m1}(s) & e_{m2}(s) & ... & e_{mp}(s) \end{bmatrix} \quad (6)$$

The general form of $R_{jj}(s)$ from $G_r(s)$ is taken as:

$$R_{ij}(s)=\frac{e_{ij}(s)}{D_r(s)} \quad (7)$$

2.2 Particle swarm optimization

The MPSO algorithm is proposed under the inspiration of the bird flock foraging process. The primary data object in PSO is "particles". Particles have a position and a speed, just like a bird searching for the optimal solution. Multiple particles form a bird swarm-like particle swarm structure. Like other evolutionary algorithms, the PSO algorithm first initializes the particle swarm, searches for the particle swarm, updates, evolves to the next generation, repeats continuously, and finally finds the optimal solution. The difference is that the PSO algorithm has evolved toward an optimal global solution and an individual optimal solution during the evolution of a new generation of particle swarms, which is why it can quickly converge [4].

Two fundamental quantities are needed to describe the state of a continually moving particle: (1) the current position *X*, which records the position coordinates of the current particle; (2) the current velocity *V*, which depends on the distance and direction of the last movement of the particle.
Besides, each particle has a fitness, which is used to measure the quality of the solution. It is a function of the optimization function *f(x)*, which can be written as *fitness = φ [f(x)]*. Here, we can simply make *fitness = f(x)*, determined by the current position *x* [15].

Also, for each particle in the particle swarm, its optimal solution *pbest* will be recorded, indicating that the particle has found the optimal solution until the current algebra; and for the current particle swarm, a globally optimal solution *gbest* will be recorded, indicating that the optimal global solution has been found so far, which is the final output. Obviously, for a particle swarm of size *N*, there must be an optimal solution of N individuals, denoted as *pbest_i*, *i = 1,2, ..., N*, where the fitness is the best (maximum or minimum, determined according to needs), namely It's *gbest*.

When the particle swarm evolves to the next generation, each particle updates itself by tracking the two "optimal solutions" *pbes*t and *gbes*t. The update formula is as follows [2]:

$$V(t+1)=wV(t)+c_1r_1(pbest-X(t))+c_2r_2(gbest-X(t)) \quad (8)$$

$$X(t+1)=X(t)+V(t+1) \quad (9)$$

Received: September 17, 2020. Revised: October 19, 2020.

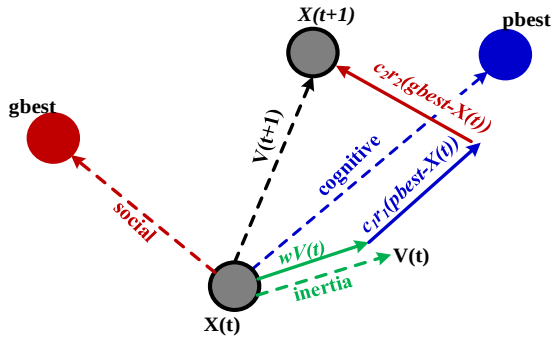


Figure. 1 Optimaization procedure of the PSO

Where *t* represents the t-th generation; *w* is the inertia weight factors, which is the proportional coefficient of the current speed *V (t)*, taking a random number between [0,1]; r_1 and r_2 are both random between [0,1] Number, c_1 and c_2 are learning factors that affect how fast the particle swarm follows the optimal solution. *X (t + 1)* represents the position of the next generation, which is obtained by adding the current position *X (t)* to the next generation speed *V(t+1)*. It can be seen that the key to particle updating is to find *V(t+1)*, and the updated schematic diagram is shown in Fig. 1

We can see that *V (t + 1)* is composed of three parts: the first part is related to the current velocity of the particle *V (t)*, indicating that the particle tends to expand in the search space; the second part is the "cognitive part," which is The result of particles absorbing self-experience; the third part is the "social part", which is the result of particles absorbing social experience.

When *V* is too large, the change of *X* update once will be huge. It is likely to skip the optimal global solution during the search, which is not conducive to the PSO algorithm and may even lead to local extreme values. Therefore, it is necessary to set a maximum value V_{max} for *V*, when $0 < V(t+1) < V_{max}$, $V = V(t+1)$, and when $V(t+1) > V_{max}$, $V = V_{max}$, so Limit *V* to the range of [0, V_{max}] to ensure the accuracy of the search.

The problem of optimization can ultimately be reduced to finding the maximum and minimum values. Let the dimension of the optimized solution be *m*, that is, $X = (x_1, x_2, \ldots, x_m)$, the optimization function is $f(X) = f(x1, x2, \ldots, xm)$, $xi \in [a, b]$, optimize The goal is to find the solution x_m that minimizes *f(X)*, then the PSO algorithm can be summarized as follows:

Step 1: Initialization

① Set the particle swarm size N, the maximum algebra M, c_1 and c_2, and other parameters;

② Initialize the current position *X(0)* and velocity *V(0)* of *N* particles. That is, for each particle position component x_i is initialized to a certain number between [a, b], and the velocity component is initialized to a certain number between $[0, \sqrt{V_m}/m]$.

③ Initialize pbest to the initial position *X(0)*, substitute and calculate its fitness *f(X)*, and use the particle with the smallest fitness as *gbest.*

Step 2: The search process

While (Algebra ≤ M)

{

① Update the position *X* and velocity *V* of each particle according to equations (8 and 9) to get a new particle group;

② Calculate the fitness *f(X)* of each particle;

③ Update the individual optimal value of each particle: if *f(X)* <*f(pbest)*, update *pbest* = *X*, otherwise do not update

④ Global optimal value update: select the *pbest* value with the smallest fitness as *gbest*;

⑤ Update algebra: *t* = *t* + *1*

}

Step 3: Result output: output *gbest* and *f(gbest)*, and the solution is completed.

2.3 Chaos optimization

Chaos is a random motion state obtained by determining the equation. It is a common phenomenon in nonlinear systems. Its behavior is complex and similar to random so that it can be used for random search. Chaotic variables have the following characteristics [16]:

Pseudo-randomness: The chaotic variables on the surface look as random as random variables. If it is a two-dimensional chaotic variable, its value is the point on the plane that looks disordered;

Ergodicity: chaotic variables can traverse all states in space without repeating; that is to say, the values of chaotic variables in the evolution process will never be repeated;

Regularity: Although the chaotic variable appears to be chaotic on the surface, it is obtained by a specific iterative equation and has an inherent regularity, which is determined by the chaotic equation;

Sensitivity to initial values: Two initial values with a minimal difference, after several iterations of chaos, the output results will produce massive changes.

Using the characteristics of pseudo-randomness and ergodicity of chaotic variables, an excellent global search function can be achieved. It is created

by the chaotic iteration equation, which reflects the characteristics of the chaotic system. By providing initial values for a chaotic variable, a set of random sequences with ergodicity and pseudo-randomness can be generated.

The logistic map is a first-order nonlinear difference equation that appears widely in the economic, social, and biological sciences. The relative simplicity of the logistics map makes it a widely used entry point in considering the concept of chaos. In general, Logic mapping is used to generate pseudo-random sequences, as shown in formula (10):

$$Z:\alpha_{n+1}=\mu\alpha_n\left(1-\alpha_n\right) \tag{10}$$

Where Z is a chaotic variable, α_n is the primary value of the chaotic variable; μ is the control parameter, when $\mu = 4$, the chaotic variable Z is in the state of full chaos, with chaos; when the chaotic variable Z is assigned an initial value α_0 (Where $\alpha_0 \neq 0.25, 0.5, 0.75$), through the iteration of the Logic equation, a sequence Z: $\alpha_0, \alpha_1, \alpha_2, \ldots, \alpha_m, \ldots,$ can be generated. Since the chaotic variables are realized as a sequence, you can use the random variables as a sequence with ergodicity within the chaotic range. Iteratively traverses the chaotic range without repeating it. When another initial value β_0 is assigned to the chaotic variable Z, another chaotic variable will be generated, no matter how close α_0 and β_0 are, after several iterations, the trajectories of the two chaotic variables will differ significantly, which is an embodiment of the sensitivity of the chaotic system to the initial value.

There are two main ways to use the Logic chaotic algorithm:

(1) From an initial value between [0,1] through iteration of the Logic equation, a chaotic, random sequence between [0,1] Z: $\alpha_0, \alpha_1, \alpha_2, \ldots,$ is generated through linear mapping (see Formula 11), extending chaos to the value range [a, b] of the optimization variable X, to traverse the value range of the optimization variable.

$$Z \to X: \quad \mathrm{X} = \alpha + (\beta - \alpha)Z \tag{11}$$

(2) Use the Logic equation to generate a chaotic, random sequence Z between [0,1], and then use carrier mapping (see formula 12) to introduce chaos into the area around *gbest*, thereby implementing local chaos search.

$$Z \to Y: \quad X = gbest + SR \cdot (Z - 0.5) \tag{12}$$

Where SR is the search radius, which is used to control the range of local chaos search.

3. The proposed methodology

3.1 Proposed modified chaos-particle swarm optimization (MCPSO)

MCPSO combines the advantages of basic PSO and chaotic algorithms, and it is an excellent algorithm with fast convergence, few control parameters, simple implementation, and avoiding local extremum.

A modified inertia weight factor was introduced to overcome the local minimum trapping defect, usually associated with the inertial weight operator. Chaos search optimizes particle locations, favors finding solutions quickly in problem space, and avoids the possibility of early convergence.

A random inertia weight is used for emotional chaos to ensure a balance between exploitation and exploration. Low inertia weight favors exploitation, while high in inertia weight favors exploration. The constant inertia weight affects the rate of convergence of the algorithm and often leads to early convergence. The messy search optimization was used in all cases due to its high dynamic properties, which ensure particle diversity.

In addition to combining the chaotic algorithm, CPSO also improved the weight coefficient w, set to a dynamic attenuation trend, the update formula of w is shown in formula (13).

$$\left.\begin{array}{l} \textit{If } \textit{fitness}(i) \le \textit{the average of the fitness values} \\ \textit{then} \\ \quad w = w_{\mathrm{m}} + \dfrac{(w_M - w_{\mathrm{m}})}{\textit{fitness}(i) - \mathrm{m}\ \textit{fitness}} \\ \textit{else} \\ \quad w = w_M \end{array}\right\} \tag{13}$$

Where w_m is the lower limit of w, and w_M is the upper limit of w. In the early stage, because the search is in the early stage and requires a large-scale search, the dependence on the speed V is greater. As evolution continues to mature, the search point is continuously close to the optimal value position. At this time, the search range needs to be narrowed. The dependence of the speed V is reduced. Such a dynamic adaptive setting is conducive to the algorithms rapid convergence of the algorithm and finds the optimal value faster.

For the m-dimensional optimization problem, $f(X) = f(x_1, x_2, \ldots, x_m)$, $x_i \in [\alpha, \beta]$, the proposed MCPSO algorithm can be summarized as follows:

Step 1: Chaos initialization population:

An m random numbers are generated between [0,1], and the initial value of each component of the m-dimensional chaotic variable $Z=(Z_1, Z_2, \ldots, Z_m)$ is generated by $2N$ iterations to generate $2N$ an m-dimensional sequence with chaotic properties. Through $X=\alpha(\beta-\alpha)Z$ mapping, it is mapped to the optimization variable; that is, $2N$ initial positions are obtained, and N of which are well adapted are selected as initial particles, then N is randomly generated within the maximum speed range. An initial velocity as the particle velocity;

Step 2: Global search

Search with the PSO algorithm to search for the global optimal solution *pbest* of each generation;

Step 3: Local search

Use the method in step (1) to generate an m-dimensional chaotic sequence of length L, and then use $X=gbest+SR(Z-0.5)$ to obtain search points with chaotic characteristics near *pbest*. The points are substituted into the optimization function for testing and compared with $f(gbest)$, respectively. *If* $f(X)<f(gbest)$ is satisfied, X can be updated to *gbest*;

Step 4: t = t + 1, if the maximum algebra is not reached, jump back to step 2 for further search; otherwise, execute step 5;

Step 5: Output

The searched *gbest* and $f(gbest)$ are output as the optimization result.

The flow chart of the MCPSO algorithm is shown in Fig. 2.

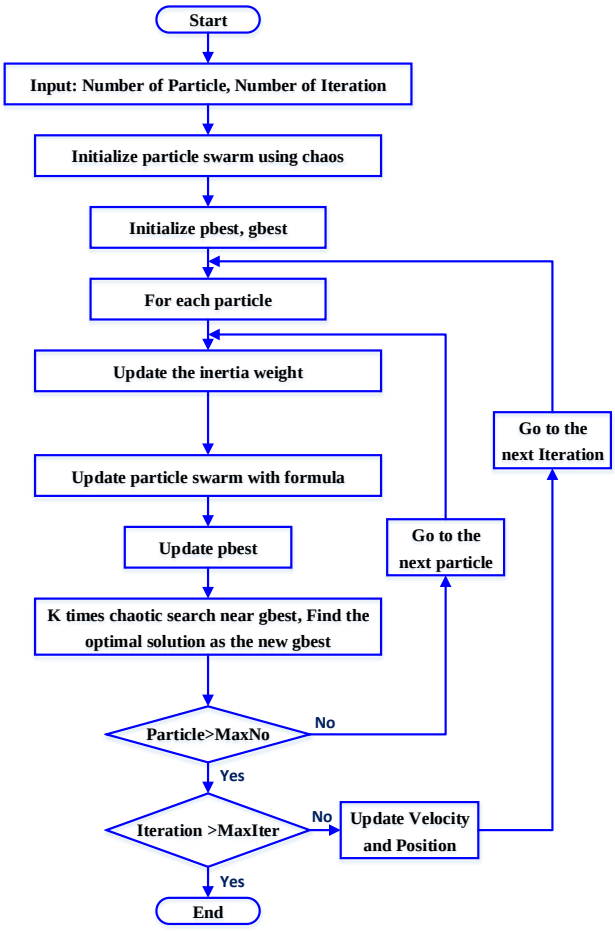


Figure. 2 Basic flow chart of the MCPSO

3.2 Proposed LQR Controller Design

Consider the optimal regulator problem shown in Fig. 3. The system equation is given by:

$$\left.\begin{aligned}\dot{x}&=Ax+Bu\\ y&=Cx+Du\end{aligned}\right\} \tag{14}$$

For the controlled system described by (14), the cost function to be minimized is given by:

$$J=\int_0^{\infty}\left(x^TQ\,x+u^T\,Ru\right)dt \tag{15}$$

Where u is not constrained, Q is required to be an asymmetric, positive, semi-definite matrix, and R is required to be an asymmetric, positive-definite matrix.

The optimal control is obtained as:

$$u=K1x+K2y_i=-R^{-1}B^TPx \tag{16}$$

where P is the solution of the Riccati equation:

$$A^TP+PA-PBR^{-1}+Q=0 \tag{17}$$

In most cases, these weights, Q and R, are chosen with the designer's expertise regarding understanding the process states. The formulation can be easily optimized by using some global optimization algorithms to search for the weighting matrices. In this paper, the MCPSO algorithm's second use is to adjust the LQR weighting matrices optimally. Similarly to practical applications, we define the $Q=diag[Q_1, Q_2, \ldots, Q_n]$ and $R=diag[R_1, R_2, \ldots, R_m]$ as diagonal matrices to alleviate the curse of dimensionality (n and m are dimensions of the state and control vectors, respectively).

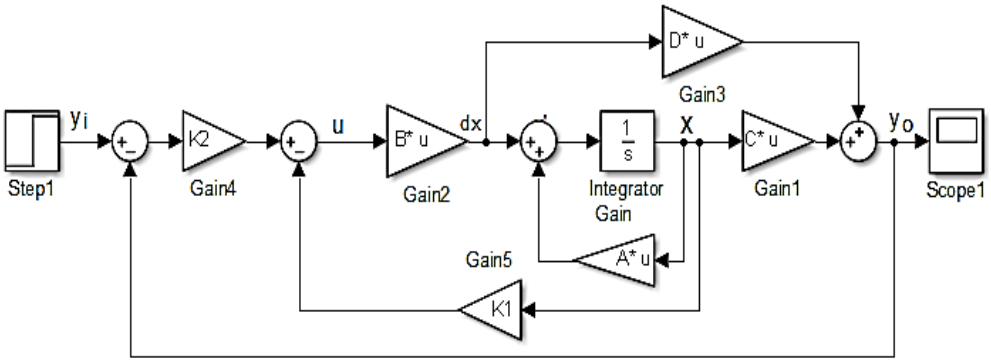


Figure. 3 MATLAB/Simulink model of the LQR controller

4. Numerical example and simulation results

In this section, the systems under consideration, and the proposed controllers are modeled and simulated in the MATLAB/Simulink environment. The proposed MCPSO algorithm performance is assessed with constant values of the initial parameters such as Swarm size=50, a Maximum number of generations=100, Acceleration factors (c_1=1.2, c_2=0.8), Inertia weight (w_m-w_M) = 0.4-0.9.

Example 1: Consider an eighth-order SISO system in Shamash [17, 18]

$$G(S)=\frac{35s^7+1086s^6+13285s^5+82402s^4+}{s^8+21s^7+220s^6+1558s^5+7669s^4+} \quad (18)$$

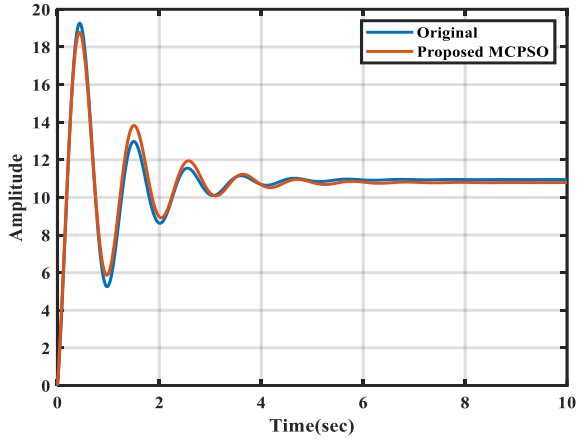
$$\frac{278376s^3+511812s^2+482964s+194480}{24469s^3+46350s^2+45952S+17760}$$

By applying the MCPSO algorithm, the ROM is:

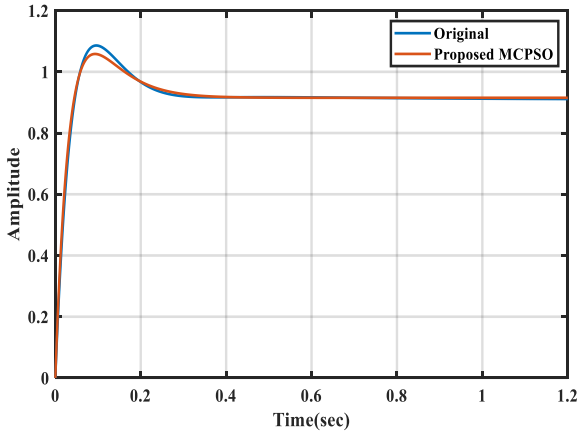
$$G_2(s)=\frac{40.05S+381.3}{S^2+1.809S+35.34} \quad (19)$$

The step responses of the higher-order model (HOM) and ROMs are shown in Fig. 4. The ISE and root mean square error (RMS) are calculated to compare this method with other known methods available in the literature, as shown in Table 1.

The convergence during the evolution of the MCPSO algorithm is shown in Fig. 5. A comparison of the fitness value, using the same number of iteration, for the techniques proposed in this work indicated by the blue line for MCPSO and in the red line for bacterial foraging-particle swarm optimization (BF-PSO) techniques proposed in [15], and in the green line for the modified Particle swarm optimization (MPSO) techniques proposed in [2].



(a)



(b)

Figure. 4 Step responses of the original model and reduced-order models for Example 1: (a): open-loop and (b) closed-loop

It can be seen from the figure that as the evolutionary algebra progresses, the fitness of the optimal global solution continues to decrease, and it stabilizes at the minimum value of 0.297 after ten generations for MCPSO. It can be seen that MCPSO can quickly converge, thus verifying that MCPSO is a more efficient and feasible algorithm.

Table 1. Comparison of error-index values with existing methods for example 1

Method	Reduced model	RMS error	ISE
Proposed MCPSO	$\frac{40.05S+381.3}{S^2+1.809S+35.34}$	0.298	0.889
Ref. [17]	$\frac{39.517S+388.959}{S^2+1.8377S+35.5199}$	0.338	1.145
Ref. [18]	$\frac{35S+401.2}{S^2+1.436S+36.63}$	0.5	2.507

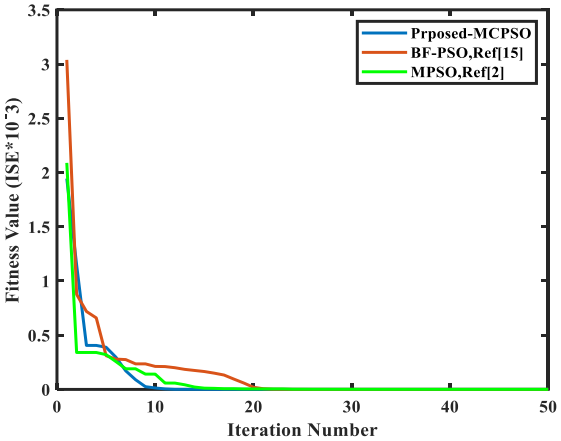


Figure. 5 Evolution processes of the MCPSO strategies for example 1

Applying the LQR controllers to the reduced model, the values of *A*, *B*, *C*, and *D* are:

$$A=\begin{bmatrix}-1.809 & -4.4175\\ 8 & 0\end{bmatrix},\quad B=\begin{bmatrix}8\\0\end{bmatrix},$$
$$c=[5.006\ \ 5.958],\quad D=[0] \tag{20}$$

The parameters of the LQR controller are tuned by using the same error minimization technique employing MCPSO. The optimized LQR controller parameters are:

$$Q=\begin{bmatrix}25.576 & 0\\ 0 & 35.6298\end{bmatrix},\quad R=[1.0126] \tag{21}$$

The optimal feedback gain matrix is:

$$K1=[5.783785\ \ 5.505185],$$
$$K2=[216.0917] \tag{22}$$

The transient response of the controller of the reduced-order system is shown in Fig. 6. To prove the

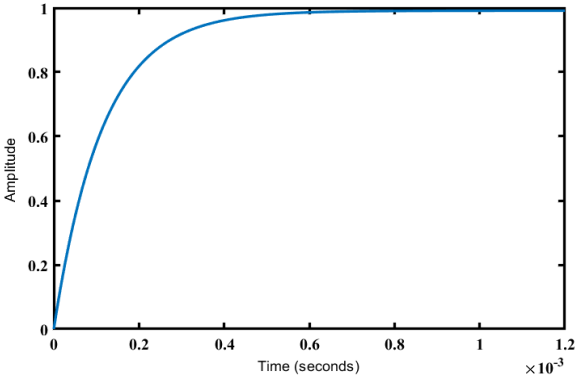


Figure. 6 Step responses of reduced-order models for example 1 using the LQR controller.

Table 2. The performance characteristics of the system in example 1.

Characteristic	**HOM**	**ROM**	**LQR**
Rise time	0.034	0.033	2.525×10^{-4}
Settling time	0.247	0.274	4.49×10^{-4}
Overshoot %)	18.5	15.6	0
Steady state	1.086	1.093	1

observation of the controller, system performance data are taken and listed in Table 2.

Example 2: Consider a sixth-order two-input two-output system transfer function matrix given in [15, 19, 20 and 21]:

$$G_6(s)=\begin{bmatrix}\frac{2(s+5)}{(s+1)(s+10)} & \frac{(s+4)}{(s+2)(s+5)}\\ \frac{(s+10)}{(s+1)(s+20)} & \frac{(s+6)}{(s+2)(s+3)}\end{bmatrix}, \tag{23}$$

$$G_6(s)=\frac{1}{D(s)}\begin{bmatrix}a_{11}(s) & a_{12}(s)\\ a_{21}(s) & a_{22}(s)\end{bmatrix}$$

By applying the MCPSO algorithm, the ROM is:

$$G_2(s)=\frac{\begin{bmatrix}(1.315s+3) & (1.033s+1.2)\\ (0.5776s+1.5) & (1.8s+3)\end{bmatrix}}{(s+3)(s+1)} \tag{24}$$

The step responses of the HOM and the ROM are shown in Fig. 7. Table 3 shows a comparison between the proposed ROM and different ROMs.
Applying the LQR controllers to the reduced model, the values of *A*, *B*, *C*, and *D* are:

$$A=\begin{bmatrix}-4 & -1.5 & 0 & 0\\ 2 & 0 & 0 & 0\\ 0 & 0 & -1.5 & -1.5\\ 0 & 0 & 2 & 0\end{bmatrix},$$

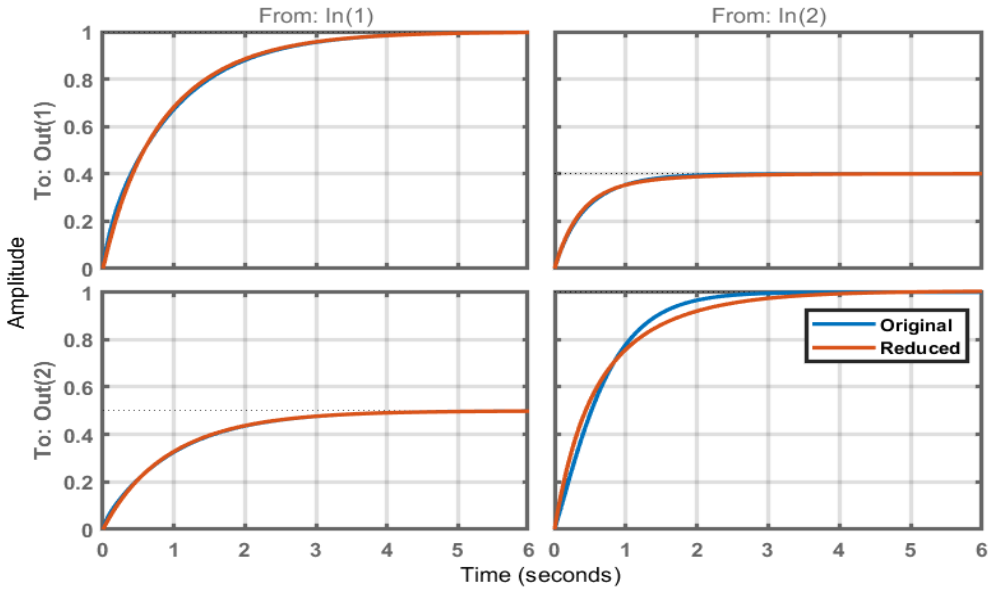
$$B'=\begin{bmatrix}2 & 0 & 0 & 0\\ 0 & 0 & 2 & 0\end{bmatrix}, D=\begin{bmatrix}0 & 0\\ 0 & 0\end{bmatrix}, \quad (25)$$

$$C=\begin{bmatrix}0.6585 & 0.7495 & 0.5155 & 0.3005\\ 0.2891 & 0.3748 & 0.8905 & 0.7535\end{bmatrix}$$

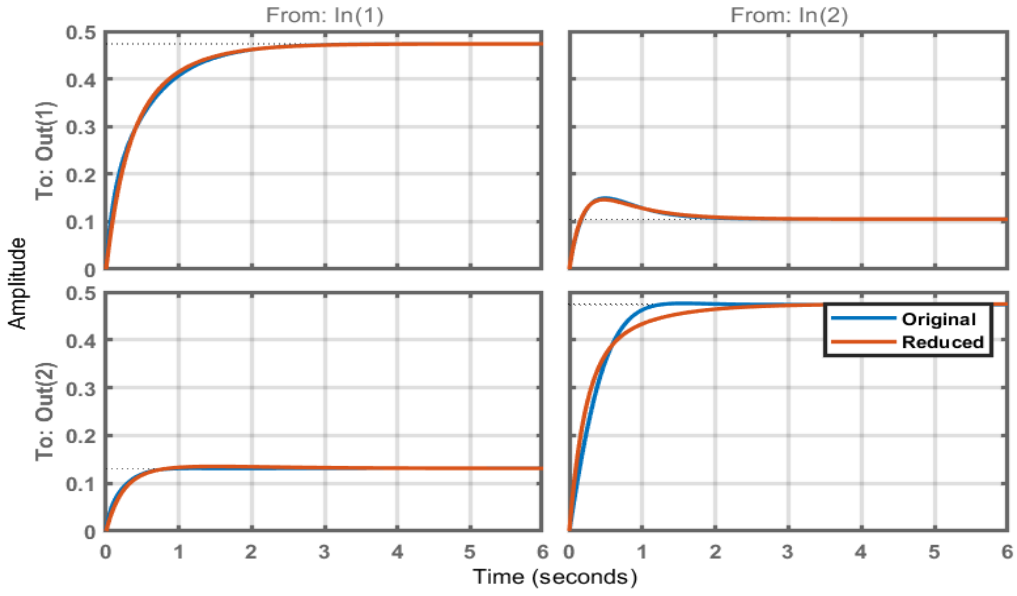
By employing MCPSO, the optimized LQR controller parameters are:

$$R=\begin{bmatrix}1000 & 0\\ 0 & 100\end{bmatrix},$$

$$Q=\begin{bmatrix}0.5172 & 0 & 0 & 0\\ 0 & 0.7022 & 0 & 0\\ 0 & 0 & 1.0587 & 0\\ 0 & 0 & 0 & 0.6581\end{bmatrix} \quad (26)$$



(a)



(b)

Figure. 7 Step responses of the original model and reduced-order models for example 2: (a) open-loop and (b) closed-loop

The optimal feedback gain matrix is: LQR upgraded parameters.

$$K1=\begin{bmatrix} 0.0004 & 0.0005 & 0 & 0 \\ 0 & 0 & 0.0048 & 0.0044 \end{bmatrix}, \quad (27)$$

$$K2=65.938$$

The transient response of the controller of the reduced-order system is shown in Fig.8. The performance data are taken and are listed in Table 4.

5. Conclusion

It can be concluded from the analysis of the numerical example that the MCPSO algorithm is an efficient optimization algorithm. It has fast convergence, simple implementation, and can overcome the local extremum problem. This MCPSO was used to reduce the high-order systems model. It notes that the ROM obtained based on the proposed MCPSO technology provides exceptional parataxis to the original system. Also, it is presented a comparative analysis of the ROM obtained by the proposed technology and that obtained by other known methods available in the literature in terms of error values means ISE and RMS, as shown in Tables 1 and 3. LQR is designed based on the MCPSO and reduced system ranking for a more convenient control method. LQR control ensures stability and adequate performance at all operating points.

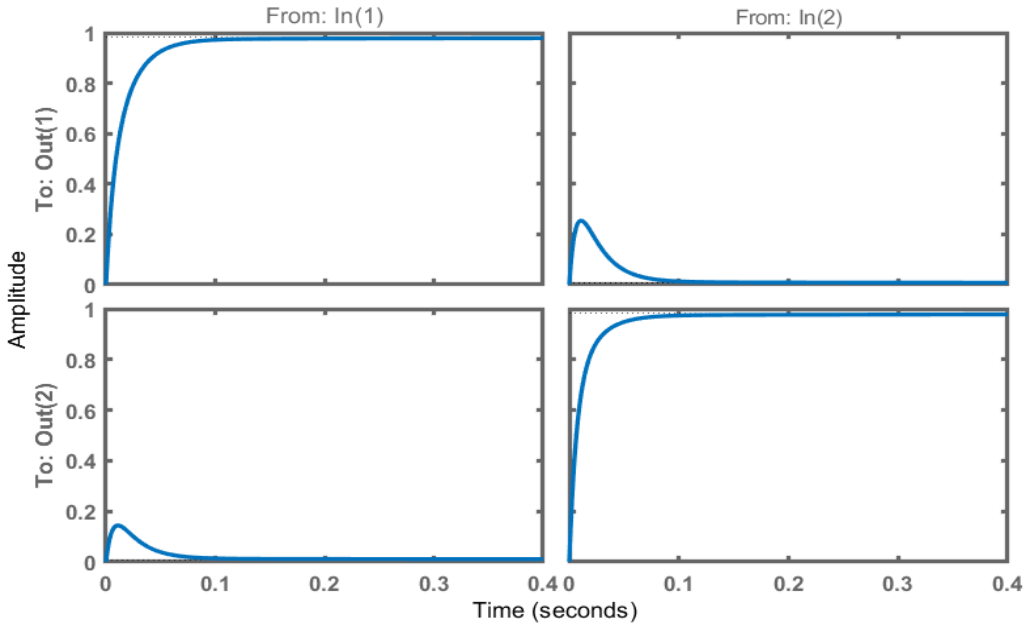


Figure. 8 The response of the reduced-order model with the LQR controller for example 2

Table 3. Comparison of error-index values with existing methods for example 2

Method	Reduced Model ($G_2(s)$)	RMS Error	ISE (1.0e-03 *)
Proposed MCPSO	$\begin{matrix} 1.315s+3 & 1.033s+1.2 \\ 0.5776s+1.5 & 1.8s+3 \end{matrix} / (s^2+4s+3)$	0.0064 0.0021 0.0028 0.0257	0.0415 0.0044 0.0081 0.6663
Ref. [15]	$\begin{matrix} 1.317s+2.998 & 1.031s+1.202 \\ 0.5782s+1.499 & 1.781s+3.014 \end{matrix} / (s^2+4s+3)$	0.0063 0.0021 0.0026 0.0245	0.0403 0.0043 0.0070 0.6048
Ref. [19]	$\begin{matrix} 0.9655s+2.068 & 0.855s+0.8272 \\ 0.5517s+1.034 & 1.689s+2.068 \end{matrix} / (s^2+3.068s+2.068)$	0.0240 0.0026 0.0027 0.0199	0.5822 0.0068 0.0072 0.4018
Ref. [20]	$\begin{matrix} 1.0546s+3.65079 & 1.07782s+1.4603 \\ 0.43458s+1.82539 & 1.9035s+3.65079 \end{matrix} / (s^2+4.3374s+3.65079)$	0.0141 0.0065 0.0033 0.0246	0.2017 0.0423 0.0110 0.6103
Ref. [21]	$\begin{matrix} 0.9098s+0.7091 & 0.4916s+0.2836 \\ 0.4373s+0.3545 & 1.0753s+0.7091 \end{matrix} / (s^2+1.548s+0.7091)$	0.0275 0.0107 0.0243 0.0283	0.7648 0.1149 0.5949 0.8117

Table 4. The performance characteristics of the system in example 2

Characteristic		HOM	ROM	LQR
R11	Rise time	1.177	1.089	0.036
	Settling time	2.156	2.144	0.074
	Overshoot (%)	0	0	0
	Steady state	0.473	0.473	1
R12	Rise time	0.128	0.124	9.1×10^{-5}
	Settling time	2.084	2.361	0.105
	Overshoot (%)	41.82	38.94	3.39×10^{3}
	Steady state	0.149	0.146	0
R21	Rise time	0.464	0.483	0.0002
	Settling time	0.767	2.324	0.158
	Overshoot (%)	0	3.024	1.48×10^{3}
	Steady state	0.131	0.135	0
R22	Rise time	0.677	0.88	0.0266
	Settling time	1.020	2.09	0.0632
	Overshoot (%)	0.557	0	0
	Steady state	0.476	0.474	1

MCPSO-based technology in the area of reduced demand model remains unexploited. Hence, our future research will aim to investigate further possible applications in some of the more complex and higher-order systems, especially in the field of electronic medical systems

Conflicts of Interest

We confirm that there is no conflict of interest in this work.

References

[1] Y. Tang, H. Hu, and Q. Tian, "Model order reduction based on successively local linearizations for flexible multibody dynamics", *International Journal for Numerical Methods in Engineering*, Vol. 118, No. 3, pp.159-180, 2019.

[2] H. Abdullah, "Reduction of Large-Scale Linear Dynamic SISO and MIMO Systems Using Modified Particle Swarm Optimization Algorithm", In: *Proc. of 11th Conf. on Industrial Electronics and Applications (ICIEA)*, pp. 66-171, 2016 Hefei, China.

[3] A. A. Sadda and K. Iqbal, "Reduced-order Modeling Using Genetic-Fuzzy algorithm", In: *Proc. of IEEE International Conf. on Systems, Man and Cybernetics*, pp. 4796 – 4800, 2009, San Antonio, TX, USA, 11-14.

[4] M. Elbes, S. Alzubi, and T. Kanan, A. Al-Fuqaha, and B. Hawashin, "A survey on particle swarm optimization with emphasis on engineering and network applications", *Evolutionary Intelligence*, Vol. 12, No. 2, pp.113-129, 2019.

[5] G. Vasu, M. Sivakumar, and M. Ramalingaraju, "A novel model reduction approach for linear time-invariant systems via enhanced PSO-DV algorithm and improved MPPA method", In: *Proc. of the Institution of Mechanical Engineers, Part I: Journal of Systems and Control Engineering*, Vol. 234, No. 2, pp. 240-256, 2020.

[6] A. Sikander and P. Thakur, "Reduced-order modeling of linear time-invariant system using modified cuckoo search algorithm", *Soft Computing*, Vol. 22, No. 10, pp. 3449-3459, 2018.

[7] C. Rim, S. Piao, G. Li, and U. Pak, "A niching chaos optimization algorithm for multimodal optimization", *Soft Computing*, Vol. 22, No. 2, pp. 621-633, 2018.

[8] Y. Hu, F. Zhu, L. Zhang, Y, Lui, and Z. Wang, "Scheduling of manufacturers based on chaos optimization algorithm in cloud manufacturing", *Robotics and Computer-Integrated Manufacturing*, Vol. 58, pp. 13-20, 2019.

[9] E. Wang, C. Jia, G. Tong, P. Qu, X. Lan, and T. Pang, "Fault detection and isolation in GPS receiver autonomous integrity monitoring based on chaos particle swarm optimization-particle filter algorithm", *Advances in Space Research*, Vo. 61, No. 5, pp. 1260-1272, 2018.

[10] H.-D. He, W.-Z. Lu, and Y. Xue, "Prediction of particulate matter at street level using artificial neural networks coupling with chaotic particle swarm optimization algorithm", *Building and Environment*, Vol. 78, pp. 111–117, 2014.

[11] T. Zhu, H. Zheng, and Z. Ma, "A chaotic particle swarm optimization algorithm for solving optimal power system problem of electric vehicle", *Advances in Mechanical Engineering*, Vol. 11, No. 3, pp. 1-9, 2019.

[12] X. Xu, H. Rong, M. Trovati, M. Liptrott, and N. Bessis, "CS-PSO: chaotic particle swarm optimization algorithm for solving combinatorial optimization problems", *Soft Computing*, Vol. 22, No. 3, pp. 783-795, 2018.

[13] D. Mazumdar, D. Sinha, S. Panja, and D. K. Dhak, "Design of LQR controller for solar tracking system", In: *Proc. of 2015 IEEE International Conf. on Electrical, Computer and Communication Technologies (ICECCT)*, Coimbatore, India, pp. 1–5, 2015.

[14] H. Abdullah, H. Sun, and M. Abd, "Design LQG/LTR Controller for Higher Order Systems Based on the Reduction Model", In: *Proc. of 2016 IEEE PES Asia-Pacific Power and Energy Engineering Conf. (APPEEC)*, pp. 2276-2281, Xi'an, China, 2016.

[15] H. Abdullah, "A hybrid bacterial foraging and modified particle swarm optimization for model order reduction", *International Journal of Electrical and Computer Engineering (IJECE)*, Vol. 9, No. 2, pp. 1100–1109, 2019.

[16] J. Feng, J. Zhang, X. Zhu, and W. Lian, "A novel chaos optimization algorithm", *Multimedia Tools and Applications*, Vol. 76, No. 16, pp.17405-17436, 2017.

[17] G. Vasu, K.V.S. Santosh, and G. Sandeep, "Reduction of large scale linear dynamic SISO and MIMO systems using differential evolution optimization algorithm", In: *Proc. of IEEE Students' Conf. on Electrical, Electronics and Computer Science, SCEECS,* Bhopal, India, pp. 1-6, 2012.

[18] T. Manigandan, N. Devarajan, and S. N. Svanandam, "Design of PID controller using reduced order model", *Acad. Open Internet Journal*, Vol. 15, pp. 1–15, 2005.

[19] S. Tiwari and G. Kaur, "Improved reduced-order modeling using clustering method with dominant pole retention", *IETE Journal of research*, Vol. 66, No. 1, pp. 42-52, 2020.

[20] A. Sikander and R. Prasad, "A new technique for reduced-order modeling of linear time-invariant system", *IETE Journal of Research*, Vol. 63, No. 3, pp. 316-324, 2017.

[21] A. Prajapati and R. Prasad, "Order reduction of linear dynamic systems by improved Routh approximation method", *IETE Journal of Research*, Vol. 65, No. 5, pp. 702-715, 2019.