Evolutionary computation based parameter estimation algorithm for state space models

Shrey Verma[1,3] shreyverma21@iitk.ac.in
Ankush Sharma[2] ansharma@iitk.ac.in
Binh Tran[3] b.tran@latrobe.edu.au
Damminda Alahakoon[3] d.alahakoon@latrobe.edu.au

***Abstract*—State-space models (SSMs) provide a compact and interpretable framework for representing dynamic systems governed by differential equations. However, as system complexity increases, particularly in single-input, multiple-output (SIMO) configurations, traditional parameter estimation methods face challenges due to the denser state and parameter matrices. These issues often lead to reduced accuracy and limited scalability. This paper presents an evolutionary state space parameter estimator (ESSPE) based on differential evolution (DE) to address these limitations. The proposed method estimates the parameters of canonical SSMs directly from input–output data without requiring prior assumptions about the model. A new fitness function is introduced to simultaneously solve multiple coupled differential equations within the SSM framework. Simulation studies on systems with 6 and 12 parameters demonstrate that ESSPE achieves estimation errors below 0.1% while maintaining robustness across varying data lengths. The results confirm that ESSPE provides improved accuracy, scalability, and interpretability compared to conventional recursive and iterative approaches, making it a reliable tool for parameter identification in complex dynamic systems.**

***Index Terms*—Evolutionary algorithm, differential evolution, parameter estimation, particle swarm optimization, state space model**

I. Introduction

The rise of machine learning (ML) and artificial intelligence (AI) algorithms makes training models easier. Moreover these models often lack transparency. Being "black boxes," many algorithms don't reveal the underlying physics of the system they represent. This can hinder in-depth analysis as well as human acceptance. Due to low interpretability and difficulty in understanding, it is difficult to understand how component values influence the system's behavior. To address this, there's a growing need for 'white box' modeling. This approach leverages real-world data to train models based on established physical equations. This provides a clearer picture of the system's inner workings and allows for more insightful analysis. This is where state space models come in a physics-based approach to describe the behavior of most of the dynamic systems [1].

State space models (SSMs), derived from the state-variable formulation of differential equations, are fundamental tools for modeling and analyzing dynamic systems [2]. Their origins can be traced to early Kalman filter research [3], where they were primarily developed for engineering applications such as tracking aircraft and missiles in noisy environments. Over time, SSMs have gained widespread importance in time-series analysis and forecasting [4]. Despite significant advancements, key challenges remain, particularly in calibrating SSMs for real-world applications. In their original engineering context, model parameters were typically assumed to be known, and early studies focused on analyzing system behavior under this assumption [5]. However, in many practical systems such as electrical rotating machines [6], transformers [7], and mechanical systems [8] parameters are often unavailable due to proprietary or accessibility constraints. Consequently, data-driven approaches have become essential for estimating unknown SSM parameters directly from measured system data [9].

Parameter estimation methods are generally categorized into recursive and iterative approaches [10]. Recursive methods enable online updates with low computational cost but often suffer performance degradation for complex models. Iterative methods provide better convergence and accuracy for such systems, albeit with higher memory demands and batch processing requirements. The selection of an appropriate method depends on application-specific accuracy and computational constraints. These techniques are widely used for both parameter estimation and system equation solving [11]. Feng [12] proposed a recursive least squares (RLS) method for canonical SSMs, later extended to multivariable systems with delayed states [13]. Xu et al. [14] introduced a stochastic gradient–based approach applicable to both delayed and non-delayed SSMs. Zhuang et al. [15] developed an iterative least-squares algorithm for single-input single-output systems, which was further improved by integrating a Kalman filter [16]. Cue et al. [17] proposed a Kalman filtering–based hierarchical generalized stochastic gradient method for joint state and parameter estimation in observable canonical SSMs. More recently, hybrid machine learning approaches have emerged, such as the Kalman filter–linear neural network framework by Yang and Shi [18], which enhances self-learning and captures nonlinear system characteristics.

The reviewed literature shows that most SSM-based identification methods rely on transforming an n^{th} -order differential equation into a set of state equations with a single output equation. While effective for low-order systems, this formulation becomes increasingly difficult to apply to large

[1]Sustainable Energy Engineering, Indian Institute of Technology Kanpur, India.
[2]Electrical Engineering, Indian Institute of Technology Kanpur, India.
[3]Centre for Data Analytics and Cognition, Latrobe University Melbourne, Australia.

and complex electrical systems. As the number of state and output equations grows, conventional estimation methods [12] often face scalability issues and reduced accuracy in parameter identification. To address these limitations, evolutionary algorithms based on differential evolution (DE) are investigated for solving the parameter estimation problem. Among meta-heuristic optimization techniques, DE is widely recognized for its robustness and effectiveness in handling nonlinear, multivariable, and multimodal optimization problems. DE has been successfully applied to estimate parameters in photovoltaic generation systems [19], where strong nonlinearities are present, and in three-phase induction motors [20], particularly when nameplate data is unavailable or machines are aged. Furthermore, DE has demonstrated effectiveness in identifying parameters of complex models such as Hammerstein–Wiener systems [21], Bayesian networks [22], and nonlinear Muskingum models [6].

DE is a promising approach for addressing challenges in SSM parameter estimation for several reasons. First, like other evolutionary algorithms, DE does not require prior assumptions about the model structure [23]. Second, DE has been extensively developed within the meta-heuristics and evolutionary computation communities, with continuous improvements in solution representation and fitness evaluation [24], enabling it to effectively handle constrained and highly complex optimization problems. Finally, as a population-based optimization method, DE demonstrates strong capability in solving large-scale, nonlinear problems with high efficiency. Its inherent parallelizability allows the learning process to be significantly accelerated using multiple processors, making DE particularly suitable for applications that demand accurate and interpretable models within limited computational time.

The main contributions of this paper are:

- A new parameter estimator algorithm for SSMs that is more efficient than past literature due to the state matrix being considered less sparse compared to previous matrices.
- The first evolutionary state space parameter estimator (ESSPE) algorithm.
- A new fitness function to solve n numbers of differential equations of SSMs.
- Analyses of accuracy and interpretability to show the efficacy of the proposed algorithm.

The rest of the paper is organised as follows. The next section II describes the background of SSM with its mathematical representation and related work. Section III describes the evolutionary algorithm mathematics, fitness function, and algorithm used. Section IV describes experimental design. The results and analyses are shown in section V. Conclusions and future directions are discussed in section VI.

II. Background

This section delves into the mathematical foundations of SSMs. We will explore the core concepts of state variables, state equations, observation equations, and the state space representation. Following this foundation, we will examine relevant research that leverages SSMs for various applications.

A. State space models

A state space model is a mathematical framework used to describe a dynamic system by outlining its internal state, the rules governing its progression over time, and the measurements taken from the system. This model comprises two types of equations:

1) State equations, one for each state variable, representing the progression of a system's internal, hidden state over time. The state variables contain the crucial information that defines the system's present condition and outline how it shifts from one state to the next.
2) Observation equations describing how the system's internal state is linked to the available or recorded observations or measurements. They explain the relationship between these observations and the hidden state, accounting for noise or uncertainties in the process.

SSMs are widely used in the analysis of electrical circuits, particularly in applications like DC-DC converters, including buck, boost, and buck-boost converters. In these systems, the key state variables typically represent the voltage across capacitors and the current through inductors. Since these converters generally contain one capacitor and one inductor, two state equations one for the capacitor voltage and one for the inductor current are used to model the system's dynamics. The observed outputs are usually the voltages across these components, while a single voltage input governs the converter's overall behavior. [25]–[28].

In another example, modeling the mass-spring-damper systems often relies on state variables that represent the behavior of the energy storage elements, like springs and dampers. When we use the state space method to analyze these systems, we typically use two or more state equations. Each equation describes the dynamics of a specific energy storage element. The observed outputs of these systems are usually the displacements (movements) of the masses. Finally, an external force acting at various points controls the overall behavior of the system [29].

Consider a state space system for Single Input Multiple Output (SIMO) presented in canonical form,

$$x(t+1) = Ax(t) + Bu(t) \tag{1}$$

$$y(t) = Cx(t) + r(t) \tag{2}$$

Equations (1) and (2) are called as the state equation and the observation equation, respectively. States are defined as minimum set of variables required to completely describe the future behavior of the system and components in the system that can store and release energy over time, such as capacitors (storing electrical energy) or inductors (storing magnetic energy). In an n^{th} order system, i.e. the numer of states is n, the state vector $x(t) := [x_1(t), x_2(t), \ldots, x_n(t)]^T \in \mathbb{R}^n$, and $u(t)$ is the system input. And the observation function, $y(t) := [y_1(t), y_2(t), \ldots, y_n(t)]^T \in \mathbb{R}^n$ is the system output

and $r(t) \in \mathbb{R}^n$ is a random noise with zero mean. The matrix $A \in \mathbb{R}^{n\times n}$ is the state matrix or system parameter matrix, $B \in \mathbb{R}^{n\times 1}$ is the input matrix or control matrix, $C := I_{n\times n} \in \mathbb{R}^{n\times n}$ is the output matrix or observation matrix. These matrices are defined as follows,

$$A := \begin{bmatrix} a_{11} & a_{12} & \dots & a_{1n} \\ a_{21} & a_{22} & \dots & a_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ a_{(n-1)1} & a_{(n-1)2} & \dots & a_{(n-1)(n-1)} \\ a_{n1} & a_{n2} & \dots & a_{nn} \end{bmatrix} \in \mathbb{R}^{n\times n}$$

$$B := \begin{bmatrix} b_1 \\ b_2 \\ \vdots \\ b_{(n-1)} \\ b_n \end{bmatrix} \in \mathbb{R}^{n\times 1}, C := \begin{bmatrix} 1 & 0 & \dots & 0 \\ 0 & 1 & \dots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \dots & 1 \end{bmatrix} := I_{n\times n}$$

The parameters $a_{ij} \in \mathbb{R}$ and $b_i \in \mathbb{R}$ are to be identified based on the observation data $\{u(t), y(t) : t = 1, 2, 3 \dots T\}$.

From (1) (2), we have

$$\begin{bmatrix} x_1(t+1) \\ x_2(t+1) \\ \vdots \\ x_{n-1}(t+1) \\ x_n(t+1) \end{bmatrix} = A \times \begin{bmatrix} x_1(t) \\ x_2(t) \\ \vdots \\ x_{n-1}(t) \\ x_n(t) \end{bmatrix} + B \times u(t), \quad (3)$$

$$\begin{bmatrix} y_1(t) \\ y_2(t) \\ \vdots \\ y_{n-1}(t) \\ y_n(t) \end{bmatrix} = C \times \begin{bmatrix} x_1(t) \\ x_2(t) \\ \vdots \\ x_{n-1}(t) \\ x_n(t) \end{bmatrix} + r(t) \quad (4)$$

which can be written as:

$$x_i(t+1) = \sum_{j=1}^{n} a_{ij}x_j(t) + b_i u(t), \ i = 1, 2, \dots, n \quad (5)$$

$$y_i(t) = Cx_i(t) + r(t), \ i = 1, 2, \dots, n \quad (6)$$

B. Differential evolution

DE is a population-based evolutionary algorithm designed for optimizing continuous-valued functions. Owing to its stochastic search capability and effectiveness, DE has been successfully applied to a broad range of global optimization problems, including scheduling, engineering design [30], and constrained optimization [31]. DE operates on a population of candidate solutions, where each individual is a D-dimensional parameter vector and D denotes the number of decision variables. The initial population is randomly generated within the feasible solution space. Through iterative application of mutation, crossover, and selection operators, the population evolves toward optimal solutions until a predefined stopping criterion—such as a maximum number of iterations or convergence threshold—is satisfied. The overall DE procedure is summarized as follows:

Step 1. Population initialization: The initial population consists of N_P individuals, each represented as

$$\theta_m^{(0)} = \left(\theta_{m,1}^{(0)}, \theta_{m,2}^{(0)}, \dots, \theta_{m,D}^{(0)}\right)^T, \quad m = 1, 2, \dots, N_P.$$

These individuals are randomly generated within the solution space. The overall population can be expressed as

$$\theta^{(0)} = \{\theta_1^{(0)}, \theta_2^{(0)}, \dots, \theta_{N_P}^{(0)}\},$$

where N_P denotes the population size.

Step 2. Mutation: For each individual in the current population,

$$\theta_m^{(t)} = \left(\theta_{m,1}^{(t)}, \theta_{m,2}^{(t)}, \dots, \theta_{m,D}^{(t)}\right)^T \in \theta^{(t)}, \quad t = 1, 2, \dots, \tilde{T},$$

a mutant vector is generated as

$$v_m^{(t)} = \left(v_{m,1}^{(t)}, v_{m,2}^{(t)}, \dots, v_{m,D}^{(t)}\right)^T.$$

Here, t represents the current generation, $\tilde{T}$ is the maximum number of generations, and

$$\theta^{(t)} = \{\theta_1^{(t)}, \theta_2^{(t)}, \dots, \theta_{N_P}^{(t)}\}$$

denotes the population at the t^{th} generation.

Different variants of Differential Evolution (DE) emerge from diverse strategies utilizing different mutation operators, which influence the generation of mutant and trial vectors. The most well-known variant is "DE/rand/1/bin." In this notation, "rand" signifies that the base vector $(\theta_{r,0}^{(t)})$ is chosen randomly, "1" indicates the number of vector pairs (i.e., vector differences) to be calculated, and "bin" refers to the application of binomial recombination.

$$DE/rand/1/bin : v_m^{(t)} = \theta_{best1}^{(t)} + F(\theta_{r1}^{(t)} - \theta_{r2}^{(t)}) \quad (7)$$

The indices $r1$, $r2$ are two mutually distinct integers within the range $[1, N_P]$, which differ from the index m and from each other. These indices are randomly generated anew for each mutant vector. The scale factor F is a positive control parameter used for scaling the difference vector.

Step 3. Crossover: Each individual $\theta_m^{(t)}$ will undergo crossover with the mutated individual $v_m^{(t)}$ in Step 2 and produce trial individual. $w_m^{(t)} = (w_{m,1}^{(t)}, w_{m,2}^{(t)}, \dots, w_{m,D}^{(t)})^T$. In the DE family of algorithms, the most commonly adopted crossover operator is the binomial operator. It can be described in the following way:

$$w_{m,j}^{(t)} = \begin{cases} v_{m,j}^{(t)}, & \text{if } rand() \le CR \text{ or } j_r = j, \\ \theta_{m,j}^{(t)}, & \text{if } rand() > CR \text{ and } j_r \ne j, \end{cases}$$

$$\text{where } m = 1, 2, \dots, N_P \text{ and } j = 1, 2, \dots, D. \quad (8)$$

Where, $rand()$ generates a uniformly distributed random number between $(0, 1)$, and j_r represents the index of a randomly chosen dimension and the crossover rate $CR \in (0, 1)$.

Step 4. Selection: The individual with the highest function fitness value of the mutated trial individual $w_m^{(t)}$ and the target

vector individual $\theta_m^{(t)}$ is chosen to move on to the next generation. This greedy method can be explained as follows:

$$\theta_m^{(t+1)} = \begin{cases} w_m^{(t)}, f(w_m^{(t)}) < f(\theta_m^{(t)}) \\ \theta_m^{(t)}, f(w_m^{(t)}) \geq f(\theta_m^{(t)}) \end{cases} \quad n = 1, 2, \ldots, N_P \quad (9)$$

Step 5. Stopping Criteria: The algorithm ends when the termination criteria are met. If not, increment t by 1 and go back to step 2.

III. Evolutionary state space parameter estimator

The aim of this study is to achieve both accuracy and scalability in parameter estimation of dynamic models. To achieve this goal, we introduce a novel algorithm called evolutionary state space parameter estimator (ESSPE) that employs an evolutionary based DE algorithm for parameter estimation in SSMs.

As mentioned in Section II, each DE individual is a potential solution, which is a vector of all parameters to be estimated. For state space models, these parameters are the elements of matrix A *and* B in (1) and (2). Therefore, ESSPE's individual representation will be:

$$\begin{aligned} \theta_m &= [\,\theta_1,\, \theta_2,\, \ldots,\, \theta_m\,]^T \\ &= [\,a_{11},\, a_{12},\, \ldots,\, a_{1n},\, a_{21},\, a_{22},\, \ldots,\, a_{2n},\, \ldots, \\ &\quad a_{n1},\, a_{n2},\, \ldots,\, a_{nn},\, b_1,\, b_2,\, \ldots,\, b_n\,]^T. \end{aligned} \quad (10)$$

where $m = 1, 2, 3 \ldots, N_P$, and $N_P = n \times \ n \ + \ n$ (the size of matrices A and B).

To evaluate each individual, θ_m is transformed back to matrices $A_{new} \in \mathbb{R}^{n \times n}$ and $B_{new} \in \mathbb{R}^{n \times 1}$, which are then used to calculate the future states $\psi(t)$ and output values $\phi(t)$ as follows:

$$\psi(t+1) = A_{new}\psi(t) + B_{new}u(t), \quad (11)$$

$$\phi(t) = C\psi(t) + r(t) \quad (12)$$

The difference between $\phi(t)$ and the ground truth $y(t)$ is considered as an error, which should be minimised. Therefore, the fitness function of ESSPE is defined as:

$$min\ f(\theta) := \sum_{i=1}^{n} \left(y_i(t) - \phi_i(t)\right)^2 \quad (13)$$

where, $y_i(t)$ is the true value and the $\phi_i(t)$ is the predicted value, calculated based on the predicted parameter (θ_m), (11) and (12).

The algorithm 1 shows the pseudocode of ESSPE, using a given fitness function (f), lower and upper bound $(lb\ and\ ub)$ of each decision variable, the size of the population (Pop_size), the number of iterations (N_{iter}), the scaling factor (F) for mutation, the probability of crossover CR, and the input data set $(u(t), y_n(t)\ , and\ C)$. The initial step involves generating a random population, followed by evaluating the fitness of each member of this population. This population is then evolved through a number of iterations N_{iter}. In each iteration, each individual θ_m is mutated and crossed over with another randomly selected individual to generate a new offspring (the donor vector), which can be used to replace θ_m if it is better than θ_m based on (9).

Algorithm 1 Evolutionary parameter estimator for SSMs

Input : $f,\ lb, ub, Pop\_size, N_{iter}, F, CR, u(t), y_n(t), C$
Output: The best individual
1 Initialize a random population $\{\theta_m\}$
2 Evaluate fitness of each individual in $\{\theta_m\}$ using f

for $t = 1\ to\ N_{iter}$ **do**
  for $m = 1\ to\ Pop\_size$ **do** $v_m \leftarrow$ Generate a donor vector for θ_m using mutation technique in (7)
$w_m \leftarrow$ Perform crossover to generate trial vector for θ_m using (8)

  for $m = 1\ to\ Pop\_size$ **do** Bound w_m
$f_{w_m} \leftarrow$ Evaluate fitness of w_m
$\theta_m \leftarrow$ Perform greedy selection using f_{w_m} and f_{θ_m} based on (9)

IV. Experimental setup

This section provides an overview of the datasets used to evaluate the performance of the proposed method. It also explains the parameter settings applied to all methods during the experiment. The proposed ESSPE algorithm is unique in its own kind, as it utilises SIMO system, whose state equation and observation equation are equal in number. Due to its uniqueness, this problem is remain unaddressed in the past literature work. Therefore, there is a lack of standard benchmark or experiment to compare the efficacy.

A. Datasets

To evaluate the effectiveness of the proposed method, it was run on two example canonical models (CM) of standard state space with predetermined sets of 6 and 12 parameters, respectively as shown below:

Canonical model 1 (n = 2):

$$\begin{bmatrix} x_{1,CM1}(t+1) \\ x_{2,CM1}(t+1) \end{bmatrix} = \begin{bmatrix} 121.56756 & -500.65781 \\ 4545.45457 & -1515.15186 \end{bmatrix} \begin{bmatrix} x_{1,CM1}(t) \\ x_{2,CM1}(t) \end{bmatrix} + \begin{bmatrix} 200.56497 \\ 605.36728 \end{bmatrix} u(t) \quad (14)$$

$$\begin{bmatrix} y(t)_{1,CM1} \\ y(t)_{2,CM1} \end{bmatrix} = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} x(t)_{1,CM1} \\ x(t)_{2,CM1} \end{bmatrix} + r(t) \quad (15)$$

Canonical model 2 (n = 3) :

$$\begin{bmatrix} x_{1,CM2}(t+1) \\ x_{2,CM2}(t+1) \\ x_{3,CM2}(t+1) \end{bmatrix} = \begin{bmatrix} -343.11591 & 655.82417 & -998.49869 \\ -708.63773 & -778.85473 & 463.26079 \\ 283.92008 & 128.75817 & -127.41719 \end{bmatrix} \begin{bmatrix} x_{1,CM2}(t) \\ x_{2,CM2}(t) \\ x_{3,CM2}(t) \end{bmatrix} + \begin{bmatrix} 406.51599 \\ 694.43693 \\ -231.89567 \end{bmatrix} u(t) \quad (16)$$

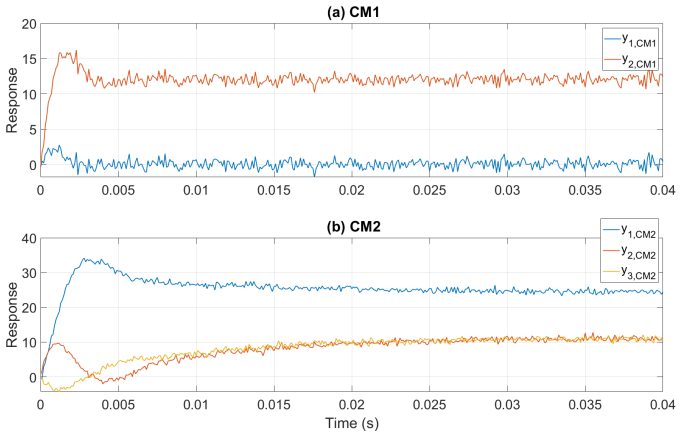

Fig. 1: System output responses (a) CM1 and (b) CM2

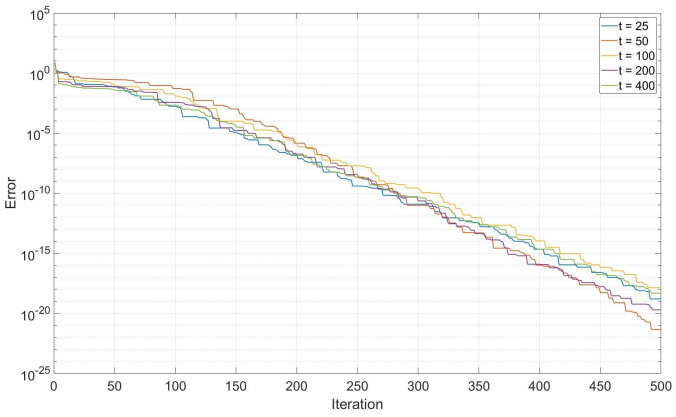

Fig. 2: Parameter estimation error versus N_{iter} for CM1

$$\begin{bmatrix} y(t)_{1,CM2} \\ y(t)_{2,CM2} \\ y(t)_{3,CM2} \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x(t)_{1,CM2} \\ x(t)_{2,CM2} \\ x(t)_{3,CM2} \end{bmatrix} + r(t) \quad (17)$$

The parameters that need to be identified are:

$$\begin{aligned} \theta_{m,CM1} &= [\, a_{11},\, a_{12},\, a_{21},\, a_{22},\, b_1,\, b_2 \,] \\ &= [\, 121.56756,\ -500.65781,\ 4545.45457, \\ &\quad -1515.15186,\ 200.56497,\ 605.36728 \,] \end{aligned} \quad (18)$$

$$\begin{aligned} \theta_{m,CM2} &= [\, a_{11},\, a_{12},\, a_{13},\, a_{21},\, a_{22},\, a_{23},\, a_{31},\, a_{32},\, a_{33},\, b_1,\, b_2,\, b_3 \,] \\ &= [\, -343.11591,\ 655.82417,\ -998.49869,\ -708.63773, \\ &\quad -778.85473,\ 463.26079,\ 283.92008, 128.75817, \\ &\quad -127.41719,\ 406.51599,\ 694.43693,\ -231.89567 \,] \end{aligned} \quad (19)$$

The input $u(t)$, to the simulation is modeled as an independent and persistently exciting sequence with zero mean and unit variance. While $r(t)$ is modelled as white noise with zero mean and variance $\sigma^2 = 0.30$. Assuming that all the states are observable at output $y(t)$. The proposed ESSPE algorithm is applied to estimate all the parameters based on a dataset generated in MATLAB code environment with different data length set. The maximum data length is generated for 0.04 seconds with a step interval of 0.0001 seconds. Different data lengths, N_t, are set at 25, 50, 100, 200, and 400 to test the efficacy of the ESSPE algorithm. Figure 1 illustrates the system outputs responses of CM1 and CM2, respectively. Each representing the time-domain response of distinct state-space system equations. Both responses are perturbed by white noise, making the outputs fluctuate around a baseline behavior.

B. Hyperparameter Settings

The hyperparameter settings for the DE are described in Table I. Since the size of the search space is directly proportional to the dimensionality of the problem, which is the number of parameters (N_P), we set the population size of DE proportional to N_P. With twice the number of parameters, the second problem is much more complex than the first one; therefore, it requires a longer evolutionary process to converge. Therefore, the number of iterations was set to 500 and 3000 for the two problems, respectively. The experiments were run on a PC with the following configurations: Intel Xenon CPU E5-2697 @2.3GHz, RAM 128GB, operating system Window Server 2019 Standard, Python 3.7. Other parameters such as the mutation factor and the crossover rate remain the same as in [32].

TABLE I: DE Hyperparameter settings.

Hyperparameters	Settings
Population Size	$10 \times N_P$
Mutation Strategy	$best/1/bin$
Mutation Factor	[0.5, 1]
Seed	1
Crossover Rate	0.7
Maximum iterations	500 & 3000

V. Result and Analysis

To evaluate the performance of the proposed ESSPE method, the mean square error (MSE) of the true value and predicted value is calculated. To show the effectiveness of ESSPE, this section compares the results estimated by ESSPE for CM1 and CM2 with different data lengths N_t used during the training process. Table II and table IIIa, IIIb show the variation in estimated parameter of SSM for CM1 and CM2 with the change in the data length N_t provided for 500 and 3000 iterations. For CM1 model the parameter estimated are exactly same to the true value for different data length data set provided to the ESSPE algorithm. So, the error reached to 0%, but the increase in data length the execution time increase, as more no of data points need to be optimised. For the CM2 model, the parameter estimation process yielded an average error difference of approximately 0.94 %. It was observed that the percentage error in the estimated parameters increased as the data length either expanded or contracted.

Fig. 2 and 3 demonstrate the variation in fitness function error as a function of iterations for CM1 and CM2, respectively.

TABLE II: The parameter estimated for different data length t for CM1.

N_t	a_{11}	a_{12}	a_{21}	a_{22}	b_1	b_2	$\delta(\%)$	Time (sec)
25	121.56756	-500.65781	4545.45456	-1515.15186	200.56496	605.36728	0	27
50	121.56755	-500.65780	4545.45457	-1515.15185	200.56496	605.36727	0	32
100	121.56756	-500.65780	4545.45458	-1515.15185	200.56496	605.36728	0	58
200	121.56756	-500.65781	4545.45457	-1515.15186	200.56497	605.36728	0	73
400	121.56755	-500.65781	4545.45455	-1515.15185	200.56497	605.36728	0	126
True values	121.56756	-500.65781	4545.45457	-1515.15186	200.56497	605.36728		

TABLE III: The parameter estimated for different data lengths t for CM2.

(a) Estimated parameters for a_{ij}

N_t	a_{11}	a_{12}	a_{13}	a_{21}	a_{22}	a_{23}	$\delta(\%)$	Time (sec)
25	-343.10056	656.08839	-997.77575	-708.66457	-779.15002	462.38169	0.37458	288
50	-343.11595	655.82327	-998.50013	-708.63782	-778.85421	463.26170	0.00015	355
100	-343.11591	655.82417	-998.49869	-708.63773	-778.85473	463.26079	0	444
200	-343.11592	655.82410	-998.49864	-778.85469	-778.85469	463.26075	0.000005	794
400	-343.11082	655.81849	-998.48780	-778.86848	-778.86848	463.26618	0.0015	1511
True values	-343.11591	655.82417	-998.49869	-708.63773	-778.85473	463.26079		

(b) Estimated parameters for a_{ij}, b_i

N_t	a_{31}	a_{32}	a_{33}	b_1	b_2	b_3	$\delta(\%)$	Time (sec)
25	283.99457	130.23313	-123.65039	406.51127	694.44090	-231.93417	0.37458	288
50	283.92011	128.75770	-127.41834	406.51613	694.43695	-231.89567	0.00015	355
100	283.92008	128.75817	-127.41719	406.51599	694.43693	-231.89567	0	444
200	283.92010	128.75817	-127.41720	406.51601	694.43695	-231.89568	0.000005	794
400	283.26151	128.76168	-127.42051	406.50640	694.44026	-231.90070	0.0015	1511
True values	283.92008	128.75817	-127.41719	406.51599	694.43693	-231.89567		

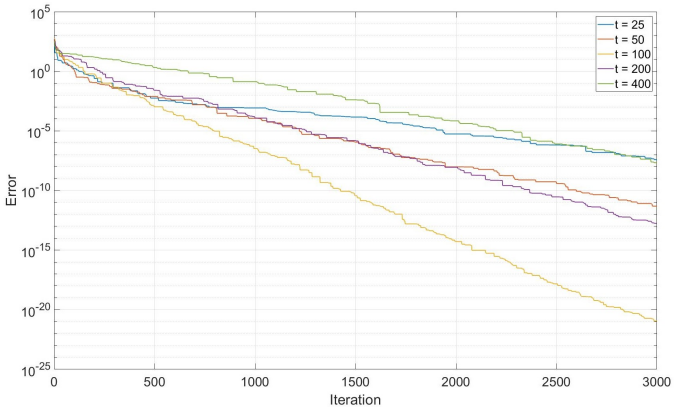

Fig. 3: Parameter estimation error versus N_{iter} for CM2

Fig. 2 shows that, for both the initial and final iterations, the fitness function errors are nearly identical for each data length set. The convergence of the estimated values to the true values provided to ESSPE is directly influenced by the variation in data length. As the data length increases or decreases, the accuracy of the estimation improves, gradually aligning the estimated values closer to the true values. This suggests a strong relationship between data quantity and the precision of the ESSPE estimation process, for smaller SSM.

In contrast, fig. 3 highlights that the final iteration errors are more distinct from each other, yet a closer analysis of the error trend reveals a consistent reduction in error with an increasing number of iterations. This pattern suggests that continuing the iterative process can further decrease the error, ultimately leading to convergence toward the true parameter values. Thus, both the amount of data and the number of iterations play crucial roles in refining the accuracy of the estimated parameters.

VI. Conclusion

This paper presents an evolutionary state-space parameter estimation (ESSPE) algorithm for identifying parameters of linear systems in canonical state-space form using input–output data. The proposed method employs differential evolution to achieve high estimation accuracy and robustness across varying data lengths. Simulation results demonstrate estimation errors below 0.1% for both 2×2 and 3×3 state-space models.

The results show that parameter estimation errors remain consistently low for all tested data lengths, with ESSPE exhibiting stable convergence at error levels of approximately $\leq 0.1\%$,. While longer data records require increased computational effort to reach the same accuracy, a clear relationship is observed between data availability and estimation precision.

ESSPE demonstrates strong scalability and robustness, maintaining high accuracy despite variations in data length and iteration count. Compared to conventional estimation techniques, whose performance often degrades with changes in

data size, ESSPE consistently delivers superior accuracy and adaptability. This combination of precision, scalability, and robustness makes ESSPE particularly well suited for accurate modeling of dynamic systems operating under diverse and data-constrained conditions.

ACKNOWLEDGMENT

This work is supported by the Anusandhan National Research Foundation (ANRF), Government of India: under MAHA EV-Mission with lead Institute IIT Kanpur for the project "Grid Readiness for EV: Enablers and Technological developments (GREET)", Sanction order no. ANRF/MAHA/2024/000115.

REFERENCES

[1] N. Lobontiu, "Chapter 8 - state-space modeling," in *System Dynamics for Engineering Students (Second Edition)*, second edition ed., N. Lobontiu, Ed. Boston: Academic Press, 2018, pp. 379–427. [Online]. Available: https://www.sciencedirect.com/science/article/pii/B9780128045596000087

[2] Y. Gu and F. Ding, "Parameter estimation for a multivariable state space system with d-step state-delay," *Journal of the Franklin Institute*, vol. 350, no. 4, pp. 724–736, 2013.

[3] R. E. Kalman, "A New Approach to Linear Filtering and Prediction Problems," *Journal of Basic Engineering*, vol. 82, no. 1, pp. 35–45, 03 1960. [Online]. Available: https://doi.org/10.1115/1.3662552

[4] N. Kantas, A. Doucet, S. S. Singh, J. Maciejowski, and N. Chopin, "On Particle Methods for Parameter Estimation in State-Space Models," *Statistical Science*, vol. 30, no. 3, pp. 328 – 351, 2015. [Online]. Available: https://doi.org/10.1214/14-STS511

[5] P. Mellodge, "Chapter 2 - system modeling," in *A Practical Approach to Dynamical Systems for Engineers*, P. Mellodge, Ed. Woodhead Publishing, 2016, pp. 17–145. [Online]. Available: https://www.sciencedirect.com/science/article/pii/B9780081002025000024

[6] D.-M. Xu, L. Qiu, and S.-Y. Chen, "Estimation of nonlinear muskingum model parameter using differential evolution," *Journal of Hydrologic Engineering*, vol. 17, no. 2, pp. 348–353, 2012.

[7] M. P. Ćalasan, A. Jovanović, V. Rubežić, D. Mujičić, and A. Deriszadeh, "Notes on parameter estimation for single-phase transformer," *IEEE Transactions on Industry Applications*, vol. 56, no. 4, pp. 3710–3718, 2020.

[8] S. Greś, K. Tatsis, V. Dertimanis, and E. Chatzi, "Parameter-state estimation for mechanical systems with small model errors," *IFAC-PapersOnLine*, vol. 56, no. 2, pp. 10 264–10 269, 2023.

[9] X. Zhang and F. Ding, "Adaptive parameter estimation for a general dynamical system with unknown states," *International Journal of Robust and Nonlinear Control*, vol. 30, no. 4, pp. 1351–1372, 2020.

[10] S. S. Haykin, *Adaptive filter theory*. Pearson Education India, 2002.

[11] X. Ma and F. Ding, "Recursive and iterative least squares parameter estimation algorithms for observability canonical state space systems," *Journal of the Franklin Institute*, vol. 352, no. 1, pp. 248–258, 2015.

[12] F. Ding, "Combined state and least squares parameter estimation algorithms for dynamic systems," *Applied Mathematical Modelling*, vol. 38, no. 1, pp. 403–412, 2014.

[13] Y. Gu and F. Ding, "Parameter estimation for a multivariable state space system with d-step state-delay," *Journal of the Franklin Institute*, vol. 350, no. 4, pp. 724–736, 2013.

[14] L. Xu, F. Ding, Y. Gu, A. Alsaedi, and T. Hayat, "A multi-innovation state and parameter estimation algorithm for a state space system with d-step state-delay," *Signal Processing*, vol. 140, pp. 97–103, 2017.

[15] L. Zhuang, F. Pan, and F. Ding, "Parameter and state estimation algorithm for single-input single-output linear systems using the canonical state space models," *Applied Mathematical Modelling*, vol. 36, no. 8, pp. 3454–3463, 2012.

[16] F. Ding, X. Liu, and X. Ma, "Kalman state filtering based least squares iterative parameter estimation for observer canonical state space systems using decomposition," *Journal of Computational and Applied Mathematics*, vol. 301, pp. 135–143, 2016.

[17] T. Cui, F. Ding, A. Alsaedi, and T. Hayat, "Recursive parameter and state estimation methods for observability canonical state-space models exploiting the hierarchical identification principle," *IET Control Theory & Applications*, vol. 13, no. 16, pp. 2538–2545, 2019. [Online]. Available: https://ietresearch.onlinelibrary.wiley.com/doi/abs/10.1049/iet-cta.2018.6333

[18] Y. Yang and Y. Shi, "State and parameter estimation algorithm for state space model based on linear neural network and kalman filter," in *2019 IEEE International Conference on Mechatronics and Automation (ICMA)*. IEEE, 2019, pp. 2314–2318.

[19] S. Gao, K. Wang, S. Tao, T. Jin, H. Dai, and J. Cheng, "A state-of-the-art differential evolution algorithm for parameter estimation of solar photovoltaic models," *Energy Conversion and Management*, vol. 230, p. 113784, 2021.

[20] J. J. Guedes, M. F. Castoldi, A. Goedtel, C. M. Agulhari, and D. S. Sanches, "Parameters estimation of three-phase induction motors using differential evolution," *Electric Power Systems Research*, vol. 154, pp. 204–212, 2018.

[21] A. Mehmood and M. A. Z. Raja, "Novel design of weighted differential evolution for parameter estimation of hammerstein-wiener systems," *Journal of Advanced Research*, vol. 43, pp. 123–136, 2023.

[22] A. Platas-Lopez, E. Mezura-Montes, N. Cruz-Ramirez, and A. Guerra-Hernandez, "Discriminative learning of bayesian network parameters by differential evolution," *Applied Mathematical Modelling*, vol. 93, pp. 244–256, 2021.

[23] V. Kachitvichyanukul, "Comparison of three evolutionary algorithms: Ga, pso, and de," *Industrial Engineering and Management Systems*, vol. 11, no. 3, pp. 215–223, 2012.

[24] A. V. Kononova, F. Caraffini, and T. Bäck, "Differential evolution outside the box," *Information Sciences*, vol. 581, pp. 587–604, 2021. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0020025521009956

[25] R. H. Tan and L. Y. Hoo, "Dc-dc converter modeling and simulation using state space approach," in *2015 IEEE Conference on Energy Conversion (CENCON)*. IEEE, 2015, pp. 42–47.

[26] M. S. M. Sarif, T. X. Pei, and A. Annuar, "Modeling, design and control of bidirectional dc-dc converter using state-space average model," in *2018 IEEE symposium on computer applications & industrial electronics (ISCAIE)*. IEEE, 2018, pp. 416–421.

[27] G. Herbst, "A building-block approach to state-space modeling of dc-dc converter systems," *J*, vol. 2, no. 3, pp. 247–267, 2019.

[28] X. Zhou and Q. He, "Modeling and simulation of buck-boost converter with voltage feedback control," in *MATEC web of conferences*, vol. 31. EDP Sciences, 2015, p. 10006.

[29] P. Sivák and D. Hroncová, "State-space model of a mechanical system in matlab/simulink," *Procedia engineering*, vol. 48, pp. 629–635, 2012.

[30] V. V. De Melo and G. L. Carosio, "Investigating multi-view differential evolution for solving constrained engineering design problems," *Expert Systems with Applications*, vol. 40, no. 9, pp. 3370–3377, 2013.

[31] J.-G. Juang and S.-T. Yu, "Disturbance encountered landing system design based on sliding mode control with evolutionary computation and cerebellar model articulation controller," *Applied Mathematical Modelling*, vol. 39, no. 19, pp. 5862–5881, 2015.

[32] R. Storn and K. Price, "Differential evolution–a simple and efficient heuristic for global optimization over continuous spaces," *Journal of global optimization*, vol. 11, pp. 341–359, 1997.