Pramana – J. Phys. (2022) 96:3
https://doi.org/10.1007/s12043-021-02255-1

Parameter estimation of chaotic dynamical systems using LS-based cost functions on the state space

ALI MOUSAZADEH and YASSER SHEKOFTEH*

Faculty of Computer Science and Engineering, Shahid Beheshti University, Tehran, Iran
*Corresponding author. E-mail: y_shekofteh@sbu.ac.ir

MS received 4 July 2021; accepted 23 August 2021

Abstract. In this paper, two cost functions are suggested for the parameter estimation of chaotic dynamical systems based on the least squares and state space. These cost functions, named ordinary least squares and total least squares, are orders of magnitude faster and more efficient than the previously suggested cost functions in the literature. They handle chaotic systems exceptionally well, contain no internal system model, work with noisy data, are easy to implement and fast to optimise. The proposed cost functions are tested on three dynamical systems: the logistic map, the discrete nonlinear map that models neurodegenerative diseases and the Hénon map. The experimental results are then compared with other cost functions.

Keywords. Parameter estimation; dynamical systems; chaotic systems; least squares; state space.

PACS Nos 05.45.-a; 07.05.Kf; 87.00

1. Introduction

The potential for chaos exists in numerous dynamical systems in nature. Examples of such systems are discrete maps describing the dynamics of the human brain under illnesses [1], differential equations modelling the interaction between insulin and glucose [2], neurons [3,4], heart rate [5], or vocal folds models of the speech production system [6–8]. Estimating the systems' parameters can give us insight into their behaviour and therefore is of great interest [9]. Traditionally, optimisation techniques along with the simple mean squared error (MSE) cost function are used for parameter estimation (PE) of dynamical systems [10]. However, this method faces significant limitations when applied to chaotic systems due to their sensitivity to initial condition [11,12] and choice of parameters [13]. Attempts have been made to mitigate these limitations. 'Return map fingerprint' (RMF) was first suggested in [14] and improved in [15] which uses the nearest-neighbour distance between points in a state space, a suitable domain for analysing the chaotic systems rather than the time domain, as a measure of difference and minimises it in order to estimate the parameters of the system. Also, cost functions based on 'Gaussian mixture models' (GMM) and hidden Markov model (HMM) were then suggested in [16–18] which estimate the density of points in the state space using Gaussian components and minimises the negative log-likelihood of data.

Inspired by the GMM cost function, a cost function based on the 'self-organising map' (SOM) was recently suggested [19] to further improve the accuracy of the previously mentioned cost functions by combining nearest-neighbour algorithms with a probabilistic approach. A detailed comparison of these cost functions is provided in [20]. Although sensitivity to the initial condition has been reduced for PE, there is still much room for improvement. The SOM-based cost function, which is currently the fastest and most accurate method for PE of discrete dynamical chaotic systems, is a model-based approach, which means the appropriate model and its corresponding hyperparameters must be selected beforehand for PE to be possible. Furthermore, none of the previously suggested methods in the literature are gradient-based, which means they have to rely on optimisation techniques such as particle swarm optimisation [21–25]. This makes optimisation and PE cumbersome as there are many hyperparameters to tune and many decisions regarding the optimisation algorithm's explore-exploit trade-offs. This becomes even a bigger issue if multiple parameters of the system are unknown simultaneously due to the vastness of higher-dimensional space [26]. In addition, all the previously suggested cost functions need to sequentially generate

Published online: 18 December 2021

samples to measure the difference from the real data, which makes it impossible to speed up their evaluation through parallelism. This paper aims to address these problems by introducing cost functions that estimate the parameters of the dynamical systems by minimising the distance between each term and its predicted value based on previous time steps. These methods are examined over the state space of the dynamical systems.

This paper is structured as follows: first, the state-space-based ordinary least squares (OLS) cost function and its noise-robust variant, namely total least squares (TLS) cost function are introduced. The next sections provide experimental results to investigate the efficiency and accuracy of the proposed cost functions. Section 6 explores the effects of varying starting points in optimising the mentioned cost functions. Finally, §7 presents the conclusions of this paper.

2. Proposed cost functions

This section introduces the ordinary and total least squares (LS) cost functions as the proposed cost functions of the paper. Consider a discrete dynamical system of the form $X_{i+1} = F(\Theta, X_i)$ where $X_i = (x_1, x_2, \ldots, x_d)$ is a d-dimensional vector and $\Theta = (\theta_1, \theta_2, \ldots, \theta_p)$ is the unknown parameter of the model. The goal is to find the value of Θ corresponding to a set of real system measurements. The measurements may or may not include noise. That is, it cannot be known beforehand whether or not any noise is present or if there is, no information is available of its significance. First, let us consider a case where it can somehow be determined that the amount of noise is negligible. In such cases, for an arbitrary guess, Θ of the unknown parameters, the OLS cost function is

$$J(\Theta) = \frac{1}{nd}\sum_{i=1}^{n}\|F(\Theta, X_{i-1}) - X_i\|^2, \tag{1}$$

where $\|Z\|^2 = \text{trace}(ZZ^T)$ is the Frobenius norm of the difference vector and n is the number of data samples minus the initial condition, which we denote as X_0. Note that this is different from what is usually known as the mean squared error (MSE) cost function within the time domain, which is

$$J(\hat{X}(\Theta)) = \frac{1}{nd}\sum_{i=1}^{n}\|\hat{X}_i - X_i\|^2, \tag{2}$$

where $\hat{X}_i$ is the ith sample generated using Θ.

As mentioned, the MSE cost function has been thoroughly investigated in the literature. Its weaknesses are known, the most problematic one being its sensitivity to both the initial condition and the choice for Θ. If the dynamical system is chaotic, then a small change in Θ will lead to the rapid divergence of the samples. This, in turn, means that even in the close neighbourhood of the true value for Θ, the cost will remain high on long time intervals. This makes gradient-based approaches impossible, leaving only methods such as particle swarm optimisation, which can only be used at short time intervals to avoid the system's chaotic nature. These issues become even more bothersome as the number of unknown parameters increases due to the curse of dimensionality and the difficulties of optimisation in higher dimensions without taking advantage of the gradient.

However, the cost function of (1) only depends on Θ. It is not calculated using another set of samples in the time domain, taking advantage of the relation among the samples from the real system. Therefore, the dependence on the initial condition and Θ is avoided. The relation between the initial condition and its following time-step is a single point on the state space which, despite being considered in (1) has little to no effect on the value of the cost function, let alone causing massive changes similar to the MSE cost function.

Due to the absence of noise, (1) depends only on Θ. Depending on the function F, this is either a linear or a nonlinear ordinary least squares problem which can be solved by calculating the gradient of J with respect to Θ and then using methods such as steepest descent. The gradient is calculated as follows:

$$\frac{\partial J}{\partial \Theta} = \frac{2}{nd}\sum_{i=1}^{n}(F(\Theta, X_{i-1}) - X_i)\frac{\partial F}{\partial \Theta}, \tag{3}$$

where $\partial F/\partial\Theta$ is the Jacobian of the dynamical system at X_i with respect to Θ. However, when noise is present, the cost function depends on Θ and $\hat{X}$, the true values of the sampled system, since they are also unknown. In such cases, the total least squares (TLS) [27–29] cost function is

$$J(\Theta, \hat{X}) = \frac{1}{nd}\sum_{i=1}^{n}(\|\hat{X}_{i-1} - X_{i-1}\|^2 + \|F(\Theta, \hat{X}_{i-1}) - X_i\|^2). \tag{4}$$

By taking every possible source of error into account, the number of unknown variables has increased significantly. In fact, every sample of the data is unknown and will be estimated, along with the model's unknown parameters. Therefore, the gradient of J with respect to $\hat{X}$ must also be calculated as follows:

$$\frac{\partial J}{\partial \hat{X}_j} = \frac{2}{nd}\left(\hat{X}_j - X_j + (F(\Theta, \hat{X}_j) - X_{j+1})\frac{\partial F}{\partial \hat{X}_j}\right). \tag{5}$$

Table 1. MAPE values of the GMM and SOM cost functions for PE of the logistic map for $r = 3.76$ using 500 samples [20].

Mean absolute percentage error (MAPE)		
# of parameters (GMM, SOM)	GMM	SOM
80 (14, 4×4)	8.79×10^{-5}	2.47×10^{-4}
125 (22, 5×5)	3.70×10^{-5}	3.63×10^{-5}
180 (31, 6×6)	3.08×10^{-5}	2.10×10^{-5}
245 (42, 7×7)	2.86×10^{-5}	3.19×10^{-6}
320 (55, 8×8)	1.67×10^{-5}	2.04×10^{-6}
405 (70, 9×9)	1.29×10^{-5}	1.67×10^{-6}
500 (86, 10×10)	6.90×10^{-6}	6.72×10^{-7}

While calculating the gradients using programming languages such as MATLAB, it is essential to avoid loops and implement the program using vectorisation, because loops significantly increase the run-time. It is easy to see that both TLS and OLS cost functions can be evaluated in $O(nd)$ operations, which is faster than the fastest current cost function in the literature, namely SOM.

It should be noted that neither of the two cost functions suggested for parameter estimation of chaotic dynamical systems in this paper are new in any sense. This paper's point is that PE of such systems can be solved rather efficiently if the samples from the dynamical system are looked at locally and in the state space, rather than globally and the temporal dimension. In this way, the task turns into a classic regression problem, which, as will be demonstrated in this paper, can be solved using classic optimisation methods, thus eliminating the need for model-based cost functions such as SOM and GMM.

In the following, we examine the performance of the proposed methods on three dynamical systems: the logistic map, the discrete nonlinear map that models neurodegenerative diseases and the Hénon map. Here metrics named mean absolute percentage error (MAPE) and run-time are used to show performance of the PE method. The MAPE is calculated in the following way:

$$\mathrm{MAPE}(r) = \frac{1}{k}\sum_{i=1}^{k}\left|\frac{r-\hat{r}_i}{r}\right|, \tag{6}$$

where $\hat{r}_i$ is the estimate of the unknown parameter r in the ith run of the algorithm and k is the total number of runs of the algorithm.

3. The logistic map

In this section, we evaluate the performance of our proposed method on the logistic map [30] as a general dynamical system and a popular benchmark:

$$x_{i+1} = r x_i (1 - x_i), \tag{7}$$

where $r \in [0, 4]$ represents the growth rate and $x \in [0, 1]$ represents the ratio of the current population to the maximum potential population. This is a popular benchmark and has been thoroughly studied in the literature. Furthermore, it is chaotic for many values of $r \in [3.56, 4]$.

We use the logistic map to test the performance of OLS and TLS methods. Here, the steepest descent method is used to minimise the suggested cost function as a simple method. Also, a tolerance value must be selected (0.0001% in this case). If the relative change in the cost function falls below this threshold, the algorithm is terminated.

For case without noise, the OLS cost function for the logistic map is as follows:

$$J(r) = \frac{1}{n}\sum_{i=1}^{n}\| r x_{i-1}(1 - x_{i-1}) - x_i \|^2 \tag{8}$$

and the respective derivative is

$$\frac{\partial J}{\partial r} = \frac{2}{n}\sum_{i=1}^{n}(r x_{i-1}(1 - x_{i-1}) - x_i)(x_{i-1}(1 - x_{i-1})). \tag{9}$$

As the logistic map is linear with respect to its unknown parameter r and the measurements are without noise, the solution to this problem is equivalent to a simple linear regression.

The obtained results of the experiments are demonstrated in tables 1 and 2. Results are averaged over $k = 100$ runs on an Intel Core i7-4720HQ CPU. The average run-time of the OLS cost function for parameter estimation of the logistic map for $r = 3.76$ using 500 samples is 0.0032 s and the average MAPE is 8.2676×10^{-16}. By comparing these results with tables 1 and 2 we see that in the absence of noise, performing an OLS regression in the state space is several orders of magnitude faster and more accurate than the previously suggested SOM and GMM cost functions.

It is also possible to use the TLS cost function in the absence of noise. However, this would be unnecessary and a waste of computational resources because the OLS cost function converges more quickly. In the next step, we evaluate the TLS cost function in the presence of noise. We test the accuracy and run-time of the method for signal-to-noise ratio (SNR) values ranging from 10

Table 2. Run-time values of the GMM and SOM cost functions for PE of the logistic map for $r = 3.76$ using 500 samples [20].

Performance summary		
# of parameters (GMM, SOM)	GMM	SOM
80 (14, 4×4)	1.52 s	1.03 s
125 (22, 5×5)	2.50 s	1.28 s
180 (31, 6×6)	2.93 s	1.54 s
245 (42, 7×7)	3.85 s	1.79 s
320 (55, 8×8)	4.89 s	2.04 s
405 (70, 9×9)	5.43 s	2.48 s
500 (86, 10×10)	5.60 s	2.81 s

to 40 dB. The results are presented in table 3 and figure 1 demonstrates a reconstructed state space of the logistic map using the TLS cost function under noisy conditions. It is also possible to use the OLS cost function with noisy samples. However, as the OLS cost function assumes that noise cannot contribute significantly to the error, its use leads to less accurate results. We can see from table 3 that the TLS cost function, despite taking more time to converge than OLS, is still high-speed and converges within a fraction of a second.

Compared to the previously suggested cost functions such as SOM and GMM, the suggested cost functions of this paper do not depend on any hyperparameter such as the number of components. Furthermore, as both OLS and TLS cost functions do not need to generate samples in the time domain sequentially, a high degree of parallelism is possible using hardware or GPU-based implementations. Investigation of different optimisation techniques and the effect of parallelism is beyond the scope of this paper.

OLS and TLS, being gradient-based, can converge significantly faster and lead to much more accurate estimations of the unknown parameters than the previously suggested cost functions. They can also handle higher-dimensional problems much more easily.

4. Modelling neurodegenerative diseases

In this section, we investigate the efficiency of our proposed method on a benchmark from the field of biomedical engineering. The discrete map suggested in [1] can be used to model the behaviour of the brain of a person suffering from neurodegenerative diseases such as Parkinson's. The system consists of the following coupled equations:

$$\begin{cases} x_{n+1} = B\tanh(w_1 y_n) - A\tanh(w_2 y_n), \\ y_n = S_2\tanh(-S_1 x_n) + S_3, \end{cases} \tag{10}$$

where x is the input of basal ganglia and y is its output used as an input of the thalamus. It is noted in [1] that the parameters S_1, S_2 and S_3 in (10) play vital roles in determining the type of illness. For other parameters in eq. (10), values given in ref. [1] have been used.

The map of (10) has the potential for chaos, making it a good benchmark for our purposes. It is also nonlinear and much more complicated than the logistic map. Therefore, we fix the other unknown parameters and try to estimate these using samples of the map. We first consider the case where there is no measurement noise, which means that the OLS cost function is the better choice for PE. As only y depends on S_1, S_2 and S_3, there is no need to include x in the cost function. Therefore, we have

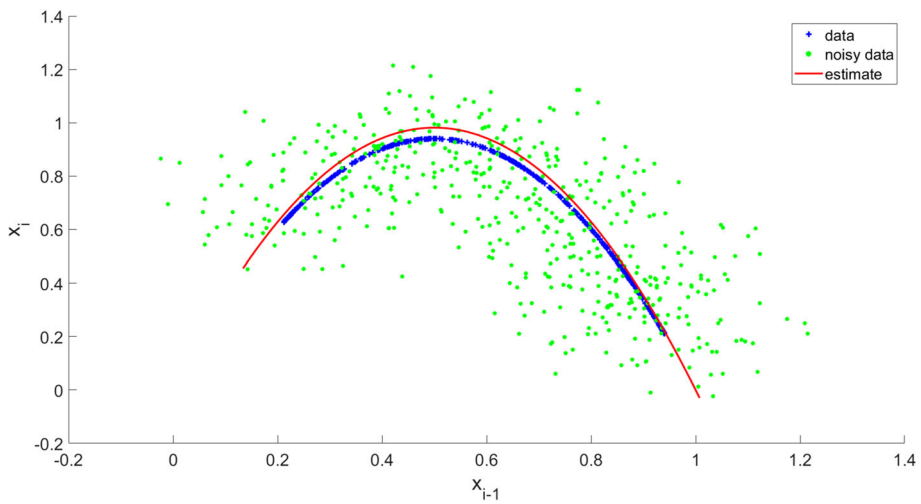

Figure 1. Reconstructed state space of the logistic map based on noisy data for SNR = 15 dB and $r = 3.76$ using the TLS cost function.

Table 3. MAPE values for PE of the logistic map for $r = 3.76$ using 500 samples plus white Gaussian noise using the TLS cost function.

	SNR (dB)				
Parameter	10	15	20	30	40
r	0.139	0.026	0.008	0.002	6.7×10^{-4}

Results are averaged over 100 runs on an Intel Core i7-4720HQ CPU. Average runtime = 0.025 s.

$$J(S_1, S_2, S_3) = \frac{1}{n}\sum_{i=1}^{n} \| S_2 \tanh(-S_1 x_i) + S_3 - y_i \|^2. \tag{11}$$

The gradients of the cost function with respect to the unknown parameters are:

$$\frac{\partial J}{\partial S_1} = \frac{2}{n}\sum_{i=1}^{n}(S_2 \tanh(-S_1 x_i) + S_3 - y_i) \times (-S_2 x_i \operatorname{sech}^2(-S_1 x_i)) \tag{12}$$

$$\frac{\partial J}{\partial S_2} = \frac{2}{n}\sum_{i=1}^{n}(S_2 \tanh(-S_1 x_i) + S_3 - y_i) \times (-\tanh(-S_1 x_i)) \tag{13}$$

$$\frac{\partial J}{\partial S_3} = \frac{2}{n}\sum_{i=1}^{n}(S_2 \tanh(-S_1 x_i) + S_3 - y_i). \tag{14}$$

To make the task of optimisation easier and more efficient, we use MATLAB's *fminunc* function to minimise the cost and set the optimisation algorithm to quasi-Newton. The results of the OLS cost function are presented in table 4. Also, the results of the previously suggested cost function in the literature, the SOM-based cost function, are presented in table 5. We can see from these tables that the accuracy of the OLS cost function is far beyond the capabilities of the previously suggested SOM-based cost functions.

Next we investigate the effect of noise. We use the TLS cost function as it is more accurate under noise. The cost function is

$$J(S_1, S_2, S_3, \hat{X}) = \frac{1}{n}\sum_{i=1}^{n}(\|\hat{x}_i - x_i\|^2 + \|S_2 \tanh(-S_1 \hat{x}_i) + S_3 - y_i\|^2) \tag{15}$$

and the gradient with respect to $\hat{x}_i$ is

$$\frac{\partial J}{\partial \hat{x}_i} = \frac{2}{n}(\hat{x}_i - x_i + (S_2 \tanh(-S_1 \hat{x}_i) + S_3 - y_i) \times (-S_2 \hat{x}_i \operatorname{sech}^2(-S_1 \hat{x}_i))). \tag{16}$$

Similar to the case without noise, we optimise the cost function using MATLAB's *fminunc* function and set the algorithm to quasi-Newton. Table 6 demonstrates the MAPE values and figure 2 depicts its state space. As expected, the accuracy increases as noise levels decrease. Furthermore, we see that run-time also decreases as noise becomes less prominent. This is because heavier distortions of data, on average, make it more difficult for the algorithm to converge on the right values for the measured samples.

In table 6, we can see that the run-time has reached the order of seconds on this benchmark, which is to be expected because we delegated the task of optimisation

Table 4. MAPE values of the OLS cost function for PE of (10) for $B = 5.821$, $A = 18$, $w_1 = 1.487$, $w_2 = 0.2223$, $S_1 = 0.5$, $S_2 = 1.5$, $S_3 = 1$ using 1000 samples.

OLS (without noise)	
Parameter	MAPE
S_1	1.11×10^{-16}
S_2	0
S_3	0

Average run-time over 100 runs = 0.0054 s.

Table 5. MAPE values of an SOM-based cost function with 15 neurons trained on 1000 samples generated by (10) [20].

Parameter estimation results			
#	S_1	S_2	S_3
Parameter value	0.50	1.50	1.00
Estimated value	0.5007	1.4987	1.00
MAPE	1.50×10^{-3}	8.64×10^{-4}	≈ 0

Parameter estimation is done using the PSO algorithm.

Table 6. MAPE values of the TLS cost function for PE of (10) for $S_1 = 0.5$, $S_2 = 1.5$, $S_3 = 1$ using 1000 samples under noise.

	SNR (dB)			
Parameter	15	20	30	40
S_1	0.0850	0.0510	0.0129	0.0037
S_2	0.0612	0.0332	0.0093	0.0026
S_3	0.0076	0.0050	0.0015	4.359×10^{-4}
Runtime (s)	2.07	1.85	1.44	1.41

Results are averaged over 100 runs.

to *fminunc*, which is much more sophisticated and accurate than the simple steepest descent. It is recommended to avoid the simple steepest descent when the dynamical system is nonlinear in its unknown parameters. The interactions between the parameters can make the steepest descent inefficient.

In this particular system, when SNR decreases below 15 dB, the state space becomes heavily distorted. As shown in figure 2, the state space is similar to a linear function except for some slight curvatures. Under extreme noise, this slight difference might disappear, which in turn causes a linear function to be a better estimate of the state space than a tanh function. Incidentally, S_1 and S_2 can be used to make the state space similar to a linear function near places where most of the data points exist, as shown in figure 3. This causes the algorithm to make wildly inaccurate estimates of S_1 and S_2. This, however, is not a weakness of the algorithm itself but happens due to the nonlinear nature of the dynamical system and the interactions between its parameters. Therefore, one should be careful about making deductions about the parameters of the nonlinear system when the measured data are heavily corrupted by noise.

5. Hénon map

To compare the TLS and OLS cost functions with the ones suggested in the literature, we test them on another common discrete benchmark known as the Hénon map:

$$\begin{cases} x_{n+1} = 1 - Cx_n^2 + y_n \\ y_{n+1} = Bx_n \end{cases} \tag{17}$$

which produces a chaotic attractor for $B = 0.2$ and $C = 1.4$. The OLS cost function for this dynamical system is as follows:

$$J(B, C) = \frac{1}{2n}\sum_{i=1}^{n}(\|1 - Cx_{i-1}^2 + y_{i-1} - x_i\|^2 + \|Bx_{i-1} - y_i\|^2). \tag{18}$$

Note that $d = 2$ because the unknown parameters involve two dimensions of the system. The derivative with respect to the unknown parameters is

$$\frac{\partial J}{\partial B} = \frac{1}{n}\sum_{i=1}^{n}(Bx_{i-1} - y_i)(x_{i-1}) \tag{19}$$

$$\frac{\partial J}{\partial C} = \frac{1}{n}\sum_{i=1}^{n}(1 - Cx_{i-1}^2 + y_{i-1} - x_i)(x_{i-1}^2). \tag{20}$$

Additionally, the TLS cost function for the Hénon map is

$$J(B, C, \hat{X}) = \frac{1}{2n}\sum_{i=1}^{n}(\|\hat{x}_i - x_i\|^2 + \|\hat{y}_i - y_i\|^2 + \|1 - C\hat{x}_i^2 + \hat{y}_i - x_i\|^2 + \|B\hat{x}_i - y_i\|^2) \tag{21}$$

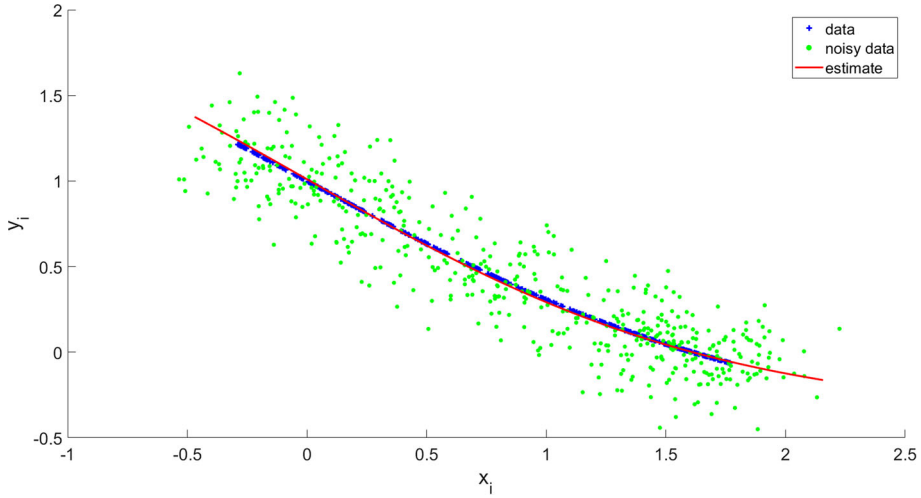

Figure 2. The state space of (10) based on noisy data for SNR = 15 dB and $S_1 = 0.5$, $S_2 = 1.5$, $S_3 = 1$ using the TLS cost function.

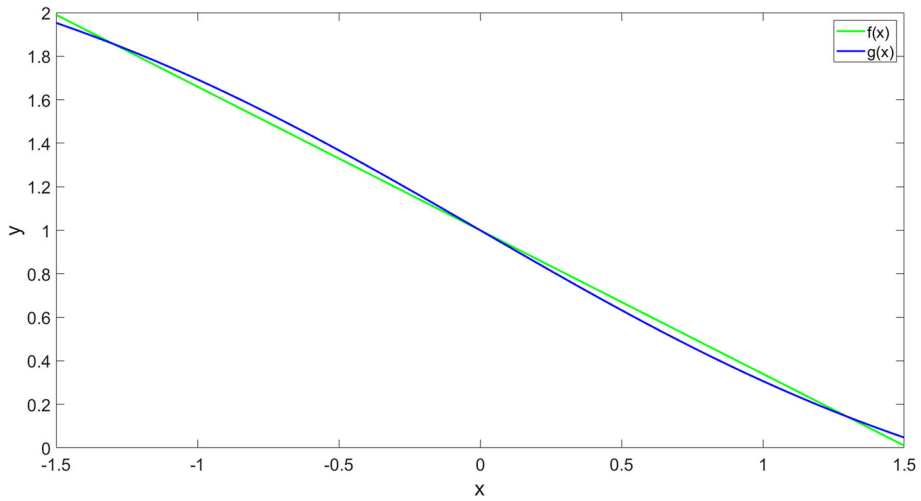

Figure 3. An example of two very different choices of parameters for (10) which lead to similar functions on a specific domain. $f(x) = 660\tanh(-0.001x) + 1$, $g(x) = 1.5\tanh(-0.5x) + 1$.

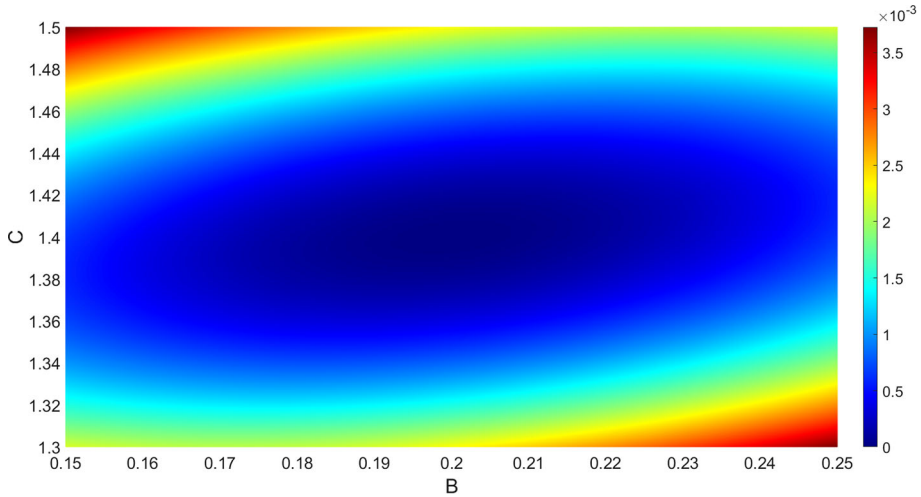

Figure 4. The OLS cost function for parameter estimation of the Hénon map.

and the respective derivatives are

$$\frac{\partial J}{\partial \hat{x}_i} = \frac{1}{n}\sum_{i=1}^{n}(\hat{x}_i - x_i + (1 - C\hat{x}_i^2 + \hat{y}_i - x_i)(-2C\hat{x}_i) + (B\hat{x}_i - y_i)B) \tag{22}$$

$$\frac{\partial J}{\partial \hat{y}_i} = \frac{1}{n}\sum_{i=1}^{n}(\hat{y}_i - y_i + (1 - C\hat{x}_i^2 + \hat{y}_i - x_i)). \tag{23}$$

Our experiments show that the OLS cost function estimates the parameters of the Hénon map in 2.5 ms averaged over 100 runs for $B = 0.2, C = 1.4$ using 500 samples, with the unbeatable error of zero percent. This and the convex shape of the cost function in figure 4 are expected because (17) is linear with respect to its unknown parameters and the measurements were not corrupted by noise. The set-up for the task of parameter estimation was chosen to be similar to [20] for a fair comparison. A detailed comparison of the performance of different cost functions on the Hénon map is provided in our previous work [20] and will not be repeated here.

Tables 7 and 8 compare the performance of the TLS and OLS cost functions under noise. It is interesting to note that the gains of using the TLS cost function for a more accurate estimate outweigh the cost of a higher run-time the most when the number of samples is low and the amount of noise is high. As the noise level decreases and the number of samples becomes more, the OLS and TLS cost functions will have comparable accuracy. However, the TLS cost function will be much slower to minimise. Therefore, as a heuristic, it is better to use the OLS cost function when the number of samples is large or the noise level is negligible. On the other hand, the TLS cost function performs best when the noise level is high and the number of samples is low. Additionally, it can be seen that higher noise levels

Table 7. MAPE and run-time values of the TLS cost function for PE of (17) for $B = 0.2$, $C = 1.4$ under several levels of noise.

	SNR (dB)				
Parameter	10	15	20	30	40
$B, n = 1000$	0.1301	0.0487	0.0216	0.0047	0.0016
$C, n = 1000$	0.0183	0.0107	0.0052	0.0017	0.0005
Run-time (s)	5.691	4.466	3.637	3.375	3.498
$B, n = 50$	0.2959	0.1347	0.0744	0.0262	0.0078
$C, n = 50$	0.0742	0.0378	0.0240	0.0074	0.0024
Run-time (s)	0.0094	0.0092	0.0083	0.0084	0.0083

Results are averaged over 100 runs.

Table 8. MAPE values of the OLS cost function for PE of (17) for $B = 0.2$, $C = 1.4$ under several levels of noise.

	SNR (dB)				
Parameter	10	15	20	30	40
$B, n = 1000$	0.3480	0.1351	0.0509	0.0077	0.0021
$C, n = 1000$	0.2417	0.0927	0.0313	0.0031	0.0006
Run-time (s)	0.0019	0.0020	0.0020	0.0020	0.0019
$B, n = 50$	0.4989	0.2298	0.1129	0.0308	0.0111
$C, n = 50$	0.2291	0.0893	0.0350	0.0084	0.0025
Run-time (s)	0.0018	0.0018	0.0018	0.0019	0.0019

Results are averaged over 100 runs.

can increase run-time. However, there is no clear-cut relation between run-time and noise level. The run-time for the OLS cost function is significantly reduced through an efficient vectorised implementation of the dynamical system. We see that even for a larger number of samples, the run-time is rather low in the order of magnitude. In the TLS cost function, we see in table 7 that when the number of samples is large, the run-time has a somewhat exponential relation with SNR, which plateaus when the noise level becomes negligible.

6. Effect of the starting point on convergence

In the absence of noise, the convergence of the OLS cost function is mostly affected by the number of samples, the number of unknown parameters of the model and the function which characterises the state space. It is clear that for functions such as the logistic map, which are linear with respect to their unknown parameters, the optimisation of the OLS cost function is rather trivial and there is no clear relation between the convergence rate and the starting point of the optimisation regardless of noise. We ran the OLS optimisation task for the logistic map using the quasi-Newton optimisation method to confirm this. We changed the starting point for r from 0 to 4 by 0.01 steps. For each starting point, random noise of 20 SNR was added to the data in 30 different runs. Then, the MAPE and average run-time values were investigated. As expected, no relation of statistical significance was apparent. The result is the same for the OLS cost function without noise. The complications in optimisation can arise only in the case of state-space functions that are nonlinear or not well-behaved. If the function can generate sub-optimal solutions, then the OLS cost function is no longer convex. As an example, consider any recursive map of the form $X_{n+1} = (\theta_1/\theta_0) f(X_n)$. In such cases, the nonlinear combination of the unknown parameters creates infinite solutions. Therefore, the starting point can significantly affect convergence because the optimisation might reach a solution which is different from the one we sought.

In almost every case, one can set the starting point of the OLS optimisation to the zero vector without significant impact on convergence. If the function is reasonably well behaved and singularities are present and rare, one way to avoid them is to manually set the starting point to a non-singularity. If certain values are physically impossible yet permissible in theory for the unknown parameters, one can take advantage of conditional optimisation to avoid those regions entirely.

Things are slightly more complicated in the case of the TLS cost function. As non-linearity is essential for chaotic behaviour and many other systems of interest are nonlinear with respect to their variables, every TLS

optimisation problem in this context can lead to complications when combined with noisy data. This is illustrated in §4 in dealing with highly corrupted samples of a nonlinear system. While certain starting points in such cases might lead to false solutions, our experiments show there is little to be gained in the run-time by changing the starting point. Since the number of unknown variables is much more in the TLS cost function, to test the effect of starting point on run-time, two separate tests were conducted. First, only the starting unknown parameter was changed for the logistic map. Similar to the OLS optimisation, the effect on run-time was negligible. Next, several methods were tested for initialising the unknown samples: zero initialisation, random uniform, random normal and initialising the unknown samples to the measured samples. These tests were conducted on the logistic map and eq. (10) 30 times for each method under an SNR of 20. Our tests showed that the difference between these methods of initialisation in convergence time is statistically negligible. As a rule of the thumb, one can set the unknown samples to the measured ones and let the algorithm adjust them accordingly.

7. Conclusion

In this paper, two state space based least-squares cost functions were proposed: Ordinary least squares (OLS) and its noise-robust variant, the total least squares (TLS). They can be optimised using explicit gradients, which causes them to have superior accuracy and faster run-time compared to all the previously suggested cost functions in the literature. They have no internal model of the data, making them easy to implement because they do not require a training phase, unlike SOM- and GMM-based cost functions. Furthermore, they do not sequentially generate samples, making high levels of parallelism possible that can make them even faster and create the opportunity for processing huge amounts of data.

References

[1] P S Shabestari, Z Rostami, V-T Pham, F E Alsaadi and T Hayat, *Commun. Theor. Phys.* **71**(**10**), 1241 (2019)

[2] P S Shabestari, S Panahi, B Hatef, S Jafari and J C Sprott, *Chaos Solitons Fractals* **112**, 44 (2018)

[3] J L Hindmarsh and R Rose, *Proc. R. Soc. London Ser. B* **221**(**1222**), 87 (1984)

[4] H Korn and P Faure, *Comptes rendus biologies* **326**(**9**), 787 (2003)

[5] M G Signorini, F Marchetti, A Cirigioni and S Cerutti, *Proceedings of the 19th Annual International Conference of the IEEE Engineering in Medicine and Biology Society.'Magnificent Milestones and Emerging Opportunities in Medical Engineering' (Cat. No. 97CH36136)* Vol. 3, pp. 1402–1405 (1997)

[6] I Tokuda and H Herzel, *Chaos* **15**(**1**), 013702 (2005)

[7] Y Zhang, C Tao and J J Jiang, *Chaos* **16**(**2**), 023118 (2006)

[8] Y Shekofteh, S Gharibzadeh and F Almasganj, *J. Med. Signals Sensors* **3**(**3**), 185 (2013)

[9] G Xu, Y Shekofteh, A Akgül, C Li and S Panahi, *Entropy* **20**(**2**), 86 (2018)

[10] G Quaranta, W Lacarbonara and S F Masri, *Nonlinear Dynam.* **99**, 1 (2020)

[11] S Jafari, S R H Golpayegani, A H Jafari and S Gharibzadeh, *Int. J. General Systems* **41**(**3**), 329 (2012)

[12] S Jafari, S M R H Golpayegani and A Daliri, *Int. J. Comput. Math.* **90**(**5**), 903 (2013)

[13] R C Hilborn *et al*, *Chaos and nonlinear dynamics: An introduction for scientists and engineers* (Oxford University Press, 2000)

[14] S Jafari, J C Sprott, V-T Pham, S M R H Golpayegani and A H Jafari, *Int. J. Bifurc. Chaos* **24**(**10**), 1450134 (2014)

[15] Y Peng, K Sun and S He, *Int. J. Bifurc. Chaos* **30**(**04**), 2050058 (2020)

[16] Y Shekofteh, S Jafari, J C Sprott, S M R H Golpayegani and F Almasganj, *Commun. Nonlinear Sci. Numer. Simul.* **20**(**2**), 469 (2015)

[17] Y Shekofteh, S Jafari, K Rajagopal and V-T Pham, *Circuits Systems Signal Processing* **38**(**5**), 2039 (2019)

[18] Y Shekofteh, S Jafari and K Rajagopal, *Soft Computing* **23**(**13**), 4765 (2019)

[19] A Mousazadeh and Y Shekofteh, *2019 27th Iranian Conference on Electrical Engineering (ICEE)* (2019), pp. 1024–1029

[20] A Mousazadeh and Y Shekofteh, *Eng. Appl. Artificial Intelligence* **94**, 103817 (2020)

[21] Q He, L Wang and B Liu, *Chaos Solitons Fractals* **34**(**2**), 654 (2007)

[22] Y Tang and X Guan, *Chaos Solitons Fractals* **40**(**3**), 1391 (2009)

[23] K Yang, K Maginu and H Nomura, *Int. J. Comput. Math.* **86**(**12**), 2225 (2009)

[24] H Modares, A Alfi and M-M Fateh, *Expert Syst. Appl.* **37**(**5**), 3714 (2010)

[25] S Mukhopadhyay and S Banerjee, *Expert Syst. Appl.* **39**(**1**), 917 (2012)

[26] S Chen, J Montgomery and A Bolufé-Röhler, *Appl. Intelligence* **42**(**3**), 514 (2015)

[27] I Markovsky and S Van Huffel, *Signal Processing* **87**(**10**), 2283 (2007)

[28] M Pešta, *WDS'08 Proceedings of Contributed Papers: Part I–Mathematics and Computer Sciences* (2008) pp. 88–93

[29] R L Branham Jr, *New Astron. Rev.* **45**(**11–12**), 649 (2001)

[30] R M May, *Nature* **261**(**5560**), 459 (1976)