OPTI 2014
An International Conference on
Engineering and Applied Sciences Optimization
M. Papadrakakis, M.G. Karlaftis, N.D. Lagaros (eds.)
Kos Island, Greece, 4-6, June 2014

DIFFERENTIAL EVOLUTION - SIMULATED ANNEALING (DESA) ALGORITHM FOR FITTING AUTOREGRESSIVE MODELS TO DATA

Rizavel C. Addawe[1] **and Joselito C. Magadia**[2]

[1]University of the Philippines Baguio
Gov. Pack Rd., Baguio City, Philippines
e-mail: rizavel.addawe@upd.edu.ph

[2] University of the Philippines Diliman
Quezon City, Philippines
e-mail: joselito.magadia@upd.edu.ph

Keywords: Differential Evolution, Simulated Annealing, Hybrid Optimization Algorithm, Autoregressive Process, AR(1) Model, Maximum Likelihood Estimation.

Abstract. *In this paper, we propose Differential Evolution - Simulated Annealing (DESA), a hybrid optimization algorithm for fitting autoregressive models to data. The addition of a new strategy based on parabolic estimation to Differential Evolution (DE) algorithm and the incorporation of the Simulated Annealing (SA) algorithm for a selection strategy makes DESA a robust optimization algorithm. The proposed hybrid algorithm obtained acceptable solutions particularly for AR(1) processes with unknown drift and additive outliers. Experiments on the parameter estimation of autoregressive models showed that the proposed hybrid algorithm, DESA has shown reliability in finding global minimum of the reference problem sets. Moreover, we compared the performance of DESA algorithm with those of DE, SA, maximum likelihood estimator (MLE) and ordinary least squares (OLS) on the fitting problem. Simulation results have shown that the proposed algorithm, DESA, provides MSE lower than those of MLE and/or OLS for almost all situations. Using 10-minute average wind speed data, DESA also obtained a better model fit on the actual series.*

1 INTRODUCTION

Because outliers are known to bias estimated model parameters [1, 2], it is therefore important to have procedures that will deal with such outlier effects. Outliers in time series were first studied by Fox [3], who proposed a classification of time series outliers to type I and type II outliers based on autoregressive model. These two types have later been renamed as additive outliers (AO) and innovational (or innovations) outliers (IO). An AO only affects a single observation, which is either larger or smaller in value than expected. After this disturbance, the series returns to its normal path as if nothing had happened. In contrast, an IO affects not only the particular observation but also subsequent observations. A time series that does not contain any outliers is called an outlier-free series. Usually, only AOs and IOs are considered in literature, but an influential article by Tsay (1988) defines three other types of outliers as well, namely level shifts, transient changes and variance changes [4]. Time series model identification with the presence of outliers are traditionally based on the estimated autocorrelations and partial autocorrelations, and will in the presence of outliers therefore be misleading, unless outliers are taken into account. These problems on model estimation of time series with outliers have been discussed in various literatures [5, 6, 7]. Methods such as least squares and maximum likelihood (ML) methods are both sensitive to the presence of outliers, especially AOs, whereas various robust estimators can handle some of the problems caused by outliers [5, 8]. This paper, presents a hybrid optimization algorithm, differential evolution and simulated annealing in fitting autoregressive models to data, particularly for the first-order autoregressive process with unknown drift and additive outliers.

Optimization algorithms inspired by the process of natural selection have been in use since the 1950s [9], and are often referred to as evolutionary algorithms (EAs). Genetic algorithm (GA) is one such method, and was invented by John Holland in the 1960s [10]. Genetic algorithms apply logical operations, usually in bit strings of fixed or variable length, in order to perform crossover, mutation, and selection on a population. Over the course of successive generations, the members of the population are more likely to represent a minimum of an objective function.

EAs have become very popular as function optimizers, because they are simple, easy to implement, and exhibit good and stable performance for wide range of functions. Evolutionary computation has become an important problem solving methodology among many researchers working in the area of computational intelligence [11]. It has been widely accepted for solving several important practical applications in engineering, science, technology, business and commerce. However, global optimization of high dimentional systems remain a challenging area of study. Researchers have to find new and extended methods which are even able to handle complex problems within huge search domains [12]. These complex systems often require the application of two or more optimization methods, since many of the known algorithms cannot be adjusted to the solution of large problems. Thus, a promising approach for the development of these methods is the intelligent combination of several powerful optimization methods by maintaining or even enhancing the advantageous characteristics of each method. However, the intelligent combination and application of these methods remains a difficult endeavor. We have to find a technique which automates the combination and interaction of several optimization algorithms [13].

Recently a number of evolutionary computing techniques such as particle swarm optimization (PSO) [14], Differential Evolution (DE) [15], Bacterial Foraging Optimization (BFO) [16] and Cat Swarm Optimization (CSO) [17] have been successfully applied to many fields. From

these algorithms, DE is found to be a simple and useful alternative to GA and has been observed to perform better for various applications such as parameter identification [18], image processing [19, 20], data clustering [21], optimal designing [22], scheduling [23] and stock market prediction [24].

In this paper, the search capabilities of DE and SA has been intermixed in an attempt to increase success rate and improve solutions obtained without lost of robustness. In our investigations, two separate experiments were done. First, we applied DE, SA and DESA algorithms on the problem of fitting AR(1) models to simulated AR(1) processes with unknown drift and additive outliers. The parameter estimates from these three algorithms were compared with estimates using maximum likelihood estimators (MLEs) and ordinary least squares (OLS) method. Next, we demostrated DESA algorithm for fitting autoregressive models to the wind speed data.

The paper is organized as follows: In the next section we briefly discuss the algorithms DE, SA and DESA algorithms. Section 3 reports experiments with the AR(1) processes with unknown drift and additive outliers using MLE, OLS, DE, SA and DESA. Application of DESA algorithm to actual data is shown in Section 4. Finally, Section 5 summarizes findings and concludes performance of the proposed DESA algorithm.

2 THE ALGORITHMS

Evolutionary algorithms are stochastic search methods that mimic the metaphor of natural biological evolution. They operate on a population of potential solutions applying the principle of survival of the fittest to produce better and better approximations to a solution. At each generation, a new set of approximations is created by the process of selecting individuals according to their level of fitness in the problem domain and breeding them together using operators borrowed from natural genetics [13]. This process leads to the evolution of population of individuals that are better suited to their environment than the individuals that they were created from, just as in natural adaptation. These algorithms model natural processes, such as *selection*, *recombination*, *mutation*, *migration*, *locality* and *neighborhood* [11].

Differential Evolution (DE), is a robust global optimizer, able to find excellent solutions to design problems, even in a complicated landscape. Unfortunately, it is slow to converge, requiring more function evaluations in several applications [15]. Simulated Annealing (SA) on the other hand is a local search algorithm able to avoid becoming trapped at local minima. While this technique is unlikely to find the optimum solution, it can often find a very good solution, even in the presence of noisy data [26, 27]. In this paper, the search abilities of SA for selecting the fittest candidate solutions were used as input into the crossover stage of the DE framework, thus the development of our proposed hybdrid algorithm, DESA [28].

2.1 Differential evolution (DE)

There are several variants of DE [11]. However, the particular variant used in this study is the *DE/rand/1/bin* scheme. The function to be optimized, f, is of the form: $f(x)\ \Re^n \rightarrow \Re$. Minimize the value of the *objective function* $f(X)$, by optimizing the value of its parameters: $\mathbf{X} = (x_1, \ldots, x_D); \mathbf{X} \in \Re^D$, where $\mathbf{X}$ denotes a vector composed of D objective function parameters. The parameters of the objective function are also subject to lower and upper boundary constraints, $x^{(L)}$ and $x^{(U)}$, respectively: $x_j^{(L)} \leq x_j \leq x_j^{(U)};\ j = 1, 2, ..., D$.

DE operates on a population of fixed size NP, not with a single solution for the optimization problem. Population P of generation G contains n_{pop} solution vectors called individuals of the

population. Each vector represents potential solution for the optimization problem: $P_G = \mathbf{X}_{i,G}$, where $i = 1, 2, \ldots, NP$; and, $G = 1, 2, \ldots, G_{max}$.

The population P of generation G contains n_{pop} individuals each containing n_{param} parameters. A member of the population may be denoted by $\mathbf{X}_{i,G}$, where i is an index to the population and G is the generation to which the population belongs. Thus, $P_G = x_{i,G} = x_{j,i,G}$, where $i = 1, 2, \ldots, NP$; and, $j = 1, 2, \ldots, D$.

In order to establish a starting point for optimum seeking, the population must be initialized. Often the only available knowlegde about the location of the optimum solution are the boundaries of the problem variables. Thus, to initialize the population P_0, is to seed it within the given boundary contraints. To seed the initial problem: $P_G = 0 : P_0 = X_{i,j,0} = r_{i,j}(x_j^{(U)} - x_j^{(L)}) + x_j^{(L)}$ where $i = 1, 2, \ldots, n_{pop}$; $j = 1, 2, \ldots, n_{param}$, and r denotes to a uniformly distributed random value within the range [0.0,1.0], chosen for each j.

From the first generation forward, the population of the following generation P_{G+1} is created in the following on the basis of the current population P_G. First, a trial population for the subsequent generation, P'_{G+1}, is generated as follows:

$$x'_{i,j,G+1} = \begin{cases} x_{i,j,G} + F(x_{Ai,j,G} - x_{Bi,j,G}) \\ \qquad if \;\; r_{i,j} \leq C_r \vee j = D_i \\ \\ \qquad\qquad x_{i,j,G} \qquad\qquad otherwise, \end{cases} \tag{1}$$

where $i = 1, 2, \ldots, n_{pop}$; $D = 1, 2, \ldots, n_{param}$; $A = 1, 2, \ldots, n_{pop}$; $B = 1, 2, \ldots, n_{pop}$; $C = 1, 2, \ldots, n_{pop}$; $A_i \neq B_i \neq C_i \neq i$; $C_r \in [0, 1]$; $F \in [0, 2]$; and $r \in [0, 1]$.

A, B, and C are randomly chosen indeces referring to three randomly chosen individuals of population. These indeces are mutually different from each other and also different from the running index i. New, random values for A, B, C are assigned for each of index i (for each individual). A new value for random number r is assigned for each value of index j (for each chromosome). The index D refers to a randomly chosen chromosome of each individual vector $\mathbf{X}'_{G+1}$ differs from each counterpart in the previous generation, $\mathbf{X}_G$. A new random (integer) value is assigned to D for each value of index i (for each individual). F, CR and NP are DE fixed control parameters in the search process. F is a real valued factor in range $[0.0, 2.0]$ that controls the amplification of differential variations and CR is a real-valued crossover factor in range $[0.0, 1.0]$ controlling the probability to choose mutated value for x instead of its current value. Generally, both F and CR affect the convergence velocity and robustness of the search process. Their optimal values are dependent both on the objective function, $f(X)$, characteristics and on the population size NP [30].

The selection scheme of DE also differs from the other EAs. On the basis of the current population, P_G, and the temporary population, P'_{G+1}, the population of the next generation, P_{G+1} is created as follows:

$$X_{i,G+1} = \begin{cases} X'_{i,G} \quad if \qquad f_{cost}(X'_{i,G+1}) \leq f_{cost}(X_{i,G}) \\ X_{i,G} \qquad\qquad otherwise \end{cases} \tag{2}$$

Therefore, every individual of the trial population is compared with its counterpart in the current population. The one with the lower value of the cost function $f_{cost}(\mathbf{X})$ will survive to

the population of the next generation. Thus, every individual of the next generation is as good or better than its counterpart in the current generation. The interesting feature concerning DE's selection scheme is that a trial vector is not compared against all the individuals in the current population, but only against one individual, against its counterpart in the current population [30, 31].

2.2 Simulated annealing (SA)

SA are based on an analogy with a physical phenomenon of the annealing process of metals. When metals are cooled slowly enough, nature is able to find the minimum energy crystalline structure by redistribution of the atoms as they lose mobility. If the temperature is decreased too quickly, a liquid metal rather end up in an amorphous state with a higher energy and not in a pure crystal. The algorithm to find minimum temperature can be used in the search for a minimum in a more general system. SA algorithms randomly generate at each iteration a candidate point and through a random mechanism controlled by a parameter called temperature, they decide whether to move to candidate point or to stay in the current one at the next iteration [32].

The objective function in mathematical optimization is analogous to energy in physical system, and the global minimum to the minimum energy state. An annealing schedule describes the temperature parameter T, and gives the rules for lowering it as the search progresses. The annealing schedule decreases T for each successive population. The algorithm employs a random search, which not only accepts decreases in the objective function $f(x)$, but also some increases. The later are accepted with a probability $e^{-(\Delta f)T}$, where Δf, is the increase in f, and T is given by the annealing schedule. Thus, the control space is randomly examined among the local minima of depth less than about T. As T is lowered, the number of such minima qualifying for frequent visits is gradually reduced. The major advantage of this algorithm over the other methods is the ability to avoid becoming trapped at local minima [26].

SA algorithm starts with an initial guess of the parameters and a temperature value T. The amounts of the optimization parameters are randomly varied proportional to T, and the objective function are calculated for each variation. The change in the objective function, may be denoted by ΔE. If $\Delta E < 0$, we have a set of parameters that results in a better value of the objective function; then, this set is retained and forms the basis of the next variation. However, a set of parameters resulting in a worse or an increased objective function might be retained depending on how large ΔE is and where this lies on a known probability distribution, $P(\Delta E)$. A random number between 0 and 1 is drawn, and if this is less than $P(\Delta E)$; then, the set of parameters are retained. For some fixed number of times, this process is repeated.

After which, the value of T is decreased by an acceptably small amount and the whole set of iterations is done again. Smaller value of T gives shorter range of varying the parameters. After a number of iterations and when T is sufficiently small, a near optimal set of parameters will have been found. Simulated annealing surely identifies the global minimum from the several local minima of the objective function. However, attaining convergence is intense and computationally long [32].

The difference between a simple random walk and a simulated annealing algorithm occurs mainly in the selection of a new solution. The generic random walk looks for a smaller objective function value, and steps in a direction in a parameter space corresponding to a decrease in E. Simulated annealing allows increase in E with a known probability, given by Boltzmann distribution: $P(\Delta E) = e^{(\frac{-\Delta E}{T})}$, where $P(\Delta E)$ is a probability distribution giving the probability of making an energy increasing step having magnitude ΔE. This is used in simulated annealing

algorithm in the form of acceptance criterion, returning true if the ΔE will be accepted and the step in parameter space made, and false otherwise. To do this, simply draw a random number between 0 and 1 (a probability) and see if it is less than $P(\Delta E)$; return true if so [33].

2.3 Differential evolution - simulated annealing (DESA) algorithm

The DESA algorithm introduced in this paper is a combination of the DE and SA algorithms. The combination is done by incorporting an SA-like selection criteria in a DE framework to form the DESA algorithm. Thus, the DESA algorithm is characterized by **self-organization**, **mutation**, **crossover**, and **SA-like selection** scheme of the strategy parameters. The SA-like selection of DESA is given by the equation:

$$\Phi_{i,G+1} = \begin{cases} \Phi'_{i,G} & if \quad R(\Phi'_{i,G+1}) \leq R(\Phi_{i,G}) \\ or & e^{-[\frac{R(\Phi'_{i,G+1})-R(\Phi_{i,G})}{T_{reduct}}]} > rand[0,1] \\ \Phi_{i,G} & otherwise \end{cases} \tag{3}$$

where Φ is the parameter vector and R is the fitness function. If the trial vector has the lower value of the cost function $R(\Phi)$, it will survive to the population of the next generation. But, if the trial vector has the higher value of the cost function, then it is subjected to the *Metropolis Criterion* as done in SA. The interesting points in this DESA's selection scheme are that a trial vector is not compared against every individual, but only against one individual in the current population; and that besides accepting improvements in cost, it also to a limited extent accepts deteriorations in cost of the objective function [31].

3 AUTOREGRESSIVE PROCESS

Assume that the observed series $X_1, X_2, \ldots, X_T$ is generated by a univariate autoregressive process of order p, i.e.,

$$X_t - \mu = \sum_{k=1}^{p} \phi_k (X_{t-k} - \mu) = a_t, \tag{4}$$

where $\{a_t\}$ is a normal white noise process with mean zero and variance σ^2. Besides σ^2 the model also contains the parameters $\phi_1, \phi_2, \ldots, \phi_p$ and μ to be estimated on the basis of observation. It is assumed that the AR(p) process represents a stationary model. This requirement is satisfied if the roots of the equation

$$1 - \sum_{k=1}^{p} \phi_k B^k = 0 \tag{5}$$

lie outside the unit circle. In a stationary case, $\mu = E\{X_t\}$, i.e., the mean of $\{X_t\}$.

Technically, we can therefore formulate the AR(p) model as follows:

$$X_t = \phi_1 X_{t-1} + \phi_2 X_{t-2} + \ldots, \phi_p X_{t-p} + \epsilon_t \tag{6}$$

where X_t denotes the time series and ϵ_t indicates a white-noise process. The value of p is called the order of the AR model. If $p = \infty$, then the process is called infinite AR process. So, an autoregressive model corresponds simply to a linear regression of the current value of the

series against one or more prior values of the series, and therefore be analyzed among other methods with the standard linear least squares technique, where the resulting estimation of the parameters, Φ, has a straight forward interpretation.

Hence, for AR(p) process, the p number of parameters, $\phi_1, \phi_2, \ldots, \phi_p$ can be estimated to minimize the sum of the squared differences between the data points and the AR(p) process. There are many optimization techniques that will perform efficiently [34]. However, it is difficult to fit an autoregressive process to a set of data [35].

The objective is to minimize the sum of the squared differences between an estimated AR(p) process, $\hat{X}_t$, and the given series X_t for $t = 1, 2, \ldots, T$. The order of the process, p, is not known in advance. The value of the AR(p) process at a value x_t can be thought of as a function

$$\hat{X}_T(\Phi, x) = \sum_{k=1}^{p} \phi_k X_{t-k}, \tag{7}$$

where $\hat{X}_T$ is an AR(p) process with parameters $\Phi = \phi_1, \phi_2, \ldots, \phi_p$. The objective function can then be expressed as

$$R(\Phi) = \sum_{i=1}^{T} (X_i - \hat{X}_i)^2. \tag{8}$$

3.1 Gaussian AR(1) process with an unknown drift and additive outliers

Suppose an outlier-free time series $\{X_t; t = 1, 2, \ldots\}$ follows an $AR(1)$ model:

$$X_t = \mu + \phi(X_{t-1} - \mu) + \epsilon_t \tag{9}$$

where μ is the population mean, ϕ is an autoregressive parameter, $\phi \in (-1, 1)$, ϵ_t are unobservable independent errors and identically $N(0, \sigma_\epsilon^2)$. For $|\phi| = 1$, the AR(1) model is called the random walk model, otherwise it is called a stationary AR(1) process when $|\phi| < 1$. For ϕ close to 1.0 or near a non-stationary process, the mean and variance of the model change over time.

Let the observed time series be denoted by $\{Y_t\}$. In the simple case when $\{X_t\}$ has a single additive outlier at time point $T(1 < T < n)$, model 9 can be modified as to

$$Y_t = X_t + \delta I_t^{(T)} \tag{10}$$

where δ represents the magnitude of the additive outlier effect and $I_t^{(T)}$ is an indicator variable such that

$$I_t^{(T)} = \begin{cases} 1, & \text{if } t = T; \\ 0, & \text{if } t \neq T. \end{cases} \tag{11}$$

3.2 Experiments on fitting AR(1) processes with unknown drift and additive outliers using MLE, DE, SA and DESA

We examine the performance of the DE, SA, and DESA for the Gaussian AR(1) process with unknown drift and additive outliers, with particular emphasis on comparisons with MLE and OLS estimates. Data are generated from an AR(1) process with an unknown drift and additive outliers. The following parameter values were used: $(\mu, \sigma_\epsilon^2) = (0, 1); \phi = 0.1, 0.2, \ldots, 0.9; \sigma^2 = 1 - \phi^2$; sample sizes, $n = 25, 50, 100, 250$; the magnitude of additive outliers effect, $\delta = 3\sigma_\epsilon$

and $5\sigma_\epsilon$; percentage additive outliers, $p = 5\%$ and 10%; and, numbers of runs M=10000. In addition, the additive outliers occured randomly. All simulations were performed using programs written in the R statistical software, DEoptim package [39] and GenSA package [40]. An example of AR(1) process with 100 observations with 10% random additive outliers of magnitude 5 times the standard deviation is shown in Figure 1.

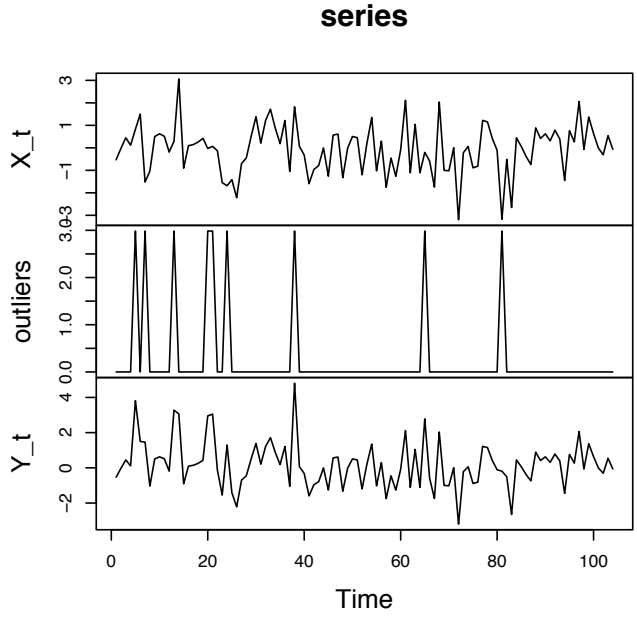

Figure 1: Graph of autoregressive process with unknown drift and additive outliers: $Y_t = X_t + \delta I_t^{(T)}$ with $T = 100$, $\phi = 0.1$, p = 0.1, and $\delta = 5$.

3.3 Parameter estimates for the AR(1) with AOs

Tables 1-2 and Figures 2-5 show the estimated MSEs of all estimators, $\hat{\phi}_{MLE}$, $\hat{\phi}_{OLS}$, $\hat{\phi}_{DE}$, $\hat{\phi}_{SA}$, and $\hat{\phi}_{DESA}$. As can be seen from Tables 1-2 and Figures 2-5, the MSE of $\hat{\phi}_{MLE}$ and $\hat{\phi}_{OLS}$ are larger than the MSEs of the other estimators especially when ϕ is close to 1.0 and for small sample sizes. These values decrease as sample sizes get larger. The value for $\hat{\phi}_{MLE}$ performs well for $n \geq 50$. On the other hand, the hybrid estimator, $\hat{\phi}_{DESA}$, provides the lowest MSE in all scenarios except when $\phi = 0.1$ and small sample sizes (n=25 and 50). Additionally, the $\hat{\phi}_{DESA}$ performs very well with respect to the other four estimators. The proposed estimator, $\hat{\phi}_{DESA}$, dominates all estimators since the MSE of the proposed estimator is the lowest for almost all cases. For the rest, the MSE of $\hat{\phi}_{SA}$ is less than that of $\hat{\phi}_{DE}$ and $\hat{\phi}_{MLE}$ and $\hat{\phi}_{OLS}$ for almost all situations. The $\hat{\phi}_{SA}$ often ranks the second best following the proposed estimator. Furthemore, the MSEs showed in Table 1 are less than those reported in Table 2 because time series data of Table 1 have less outliers.

		$\delta = 3\sigma_\epsilon$					$\delta = 5\sigma_\epsilon$				
n	ϕ	MLE	OLS	DE	SA	DESA	MLE	OLS	DE	SA	DESA
25 -	.1	0.0727	0.0830	0.0364	0.0187	**0.0184**	0.1157	0.0391	0.0340	0.0343	**0.0170**
	.2	0.0717	0.0673	0.0389	**0.0233**	0.0237	0.0981	0.0235	0.0407	0.0409	**0.0243**
	.3	0.0785	0.0502	0.0438	0.0368	**0.0327**	0.0935	**0.0144**	0.0532	0.0539	0.0387
	.4	0.0892	**0.0392**	0.0496	0.0521	0.0423	0.1001	**0.0185**	0.0688	0.0687	0.0574
	.5	0.1010	**0.0401**	0.0555	0.0687	0.0513	0.1152	**0.0308**	0.0869	0.0866	0.0778
	.6	0.1133	**0.0500**	0.0614	0.0861	0.0588	0.0834	**0.0506**	0.1056	0.1060	0.0986
	.7	0.1187	0.0739	0.0647	0.0943	**0.0625**	0.1780	**0.0739**	0.1198	0.1210	0.1145
	.8	0.1265	0.1062	0.0606	0.0922	**0.0597**	0.2190	0.1084	0.1224	0.1245	**0.1193**
	.9	0.1289	0.1589	**0.0481**	0.0876	**0.0481**	0.2384	0.1639	0.1032	0.1031	**0.0989**
50	.1	0.0390	0.0391	0.0186	0.0190	**0.0111**	0.0106	0.0267	0.0186	0.0187	**0.0103**
	.2	0.0346	0.0235	0.0209	0.0206	**0.0161**	**0.0158**	0.0307	0.0249	0.0246	0.0180
	.3	0.0365	**0.0144**	0.0252	0.0253	0.0226	**0.0225**	0.0244	0.0368	0.0368	0.0321
	.4	0.0424	**0.0185**	0.0302	0.0301	0.0295	**0.0292**	0.0300	0.0524	0.0521	0.0492
	.5	0.0494	**0.0308**	0.0356	0.0349	0.0354	**0.0352**	0.0471	0.0689	0.0687	0.0665
	.6	0.0541	0.0506	**0.0394**	0.0398	**0.0394**	**0.0392**	0.0770	0.0857	0.0861	0.0837
	.7	0.0568	0.0740	0.0398	**0.0395**	0.0401	**0.0397**	0.1178	0.0950	0.0943	0.0937
	.8	0.0533	0.1084	0.0369	0.0362	**0.0356**	**0.0368**	0.1730	0.0921	0.0924	0.0907
	.9	0.0462	0.1639	0.0252	**0.0253**	0.0256	**0.0252**	0.2479	0.0683	0.0682	0.0665
100	.1	0.0206	0.0159	**0.0100**	**0.0100**	0.0066	0.0460	0.0196	0.0103	0.0102	**0.0065**
	.2	0.0146	**0.0070**	0.0118	0.0118	0.0106	0.0261	**0.0112**	0.0164	0.0164	0.0138
	.3	0.0169	**0.0092**	0.0154	0.0155	0.0151	0.0227	**0.0145**	0.0276	0.0277	0.0261
	.4	0.0235	0.0197	0.0197	**0.0196**	**0.0196**	0.0357	**0.0321**	0.0423	0.0421	0.0416
	.5	0.0298	0.0345	0.0240	**0.0238**	0.0240	0.0594	0.0617	0.0584	**0.0579**	0.0583
	.6	0.0345	0.0507	0.0274	0.0274	**0.0271**	0.0830	0.0995	0.0735	0.0732	**0.0725**
	.7	0.0348	0.0684	0.0271	0.0271	**0.0270**	0.0958	0.1456	0.0801	0.0795	**0.0791**
	.8	0.0309	0.0925	**0.0228**	0.0230	0.0231	0.0926	0.2052	0.0734	0.0735	**0.0731**
	.9	0.0218	0.1340	0.0141	**0.0140**	0.0142	0.0685	0.2785	0.0481	**0.0478**	0.0486
250	.1	0.0076	0.0037	0.0041	0.0041	**0.0035**	0.0274	0.0055	0.0046	0.0046	**0.0038**
	.2	0.0053	**0.0040**	0.0060	0.0060	0.0060	0.0106	0.0065	0.0107	**0.0064**	0.0106
	.3	0.0115	0.0112	**0.0093**	**0.0093**	0.0095	0.0196	0.0211	0.0217	**0.0123**	0.0219
	.4	0.0171	0.0200	**0.0116**	0.0134	0.0137	0.0366	0.0461	0.0362	**0.0212**	0.0363
	.5	0.0213	0.0294	**0.0173**	**0.0173**	0.0175	0.0622	0.0773	0.0518	**0.0309**	0.0519
	.6	0.0236	0.0406	**0.0163**	0.0199	0.0199	0.0652	0.1135	0.0652	**0.0391**	0.0648
	.7	0.0231	0.0545	**0.0196**	0.0198	**0.0196**	0.0809	0.1557	0.0704	**0.0420**	0.0699
	.8	0.0187	0.0721	0.0156	0.0157	**0.0155**	0.0719	0.2061	0.0621	**0.0361**	0.0622
	.9	0.0106	0.0988	0.0081	**0.0080**	0.0082	0.0667	0.2732	0.0351	**0.0196**	0.0354

Table 1: The estimated MSEs of $\hat{\phi}_{MLE}$, $\hat{\phi}_{OLS}$, $\hat{\phi}_{DE}$, $\hat{\phi}_{SA}$, and $\hat{\phi}_{DESA}$ when percentage of additive outliers is 5%. Numbers written in bold are the least MSEs obtained for each parameter with magnitude of outliers 3σ and 5σ.

		$\delta = 3\sigma_\epsilon$					$\delta = 5\sigma_\epsilon$				
n	ϕ	MLE	OLS	DE	SA	DESA	MLE	OLS	DE	SA	DESA
25	.1	0.1203	0.0903	0.0352	0.0362	**0.0195**	0.2687	0.0901	0.0331	0.0330	**0.0188**
	.2	0.0954	0.0814	0.0389	0.0389	**0.0236**	0.2098	0.0790	0.0399	0.0396	**0.0238**
	.3	0.0852	0.0695	0.0465	0.0477	**0.0342**	0.1636	0.0984	0.0570	0.0576	**0.0404**
	.4	0.0866	0.0621	0.0580	0.0582	**0.0490**	0.1352	0.1077	0.0801	0.0806	**0.0655**
	.5	0.1004	0.0694	0.0723	0.0715	**0.0655**	0.1308	0.1289	0.1104	0.1109	**0.0973**
	.6	0.1237	0.0820	0.0846	0.0859	**0.0802**	0.1506	**0.1670**	0.1445	0.1450	0.1315
	.7	0.1529	0.1143	0.0936	0.0952	**0.0908**	0.2034	0.1965	0.1727	0.1745	**0.1639**
	.8	0.1767	0.1637	0.0940	0.0943	**0.0927**	0.2747	0.2499	0.1875	0.1883	**0.1810**
	.9	0.1880	0.2543	0.0762	0.0757	**0.0756**	0.3549	0.3623	0.1685	0.1680	**0.1615**
50	.1	0.0807	0.0441	**0.0192**	**0.0192**	0.0539	0.2114	0.0478	0.0182	0.0184	**0.0120**
	.2	0.0517	0.0315	0.0217	**0.0214**	0.0494	0.1484	0.0460	0.0242	0.0240	**0.0182**
	.3	0.0388	0.0230	0.0288	**0.0287**	0.0477	0.1022	0.0499	0.0401	0.0400	**0.0349**
	.4	0.0418	**0.0274**	0.0383	0.0380	0.0477	0.0785	0.0634	0.0633	0.0629	**0.0592**
	.5	0.0575	**0.0451**	0.0497	0.0497	0.0484	**0.0807**	0.0862	0.0919	0.0918	0.0883
	.6	0.0754	0.0750	0.0607	0.0608	**0.0452**	**0.1071**	0.1268	0.1230	0.1226	0.1196
	.7	0.0905	0.1128	0.0659	0.0659	**0.0355**	0.1511	0.1814	0.1464	0.1461	**0.1439**
	.8	0.0931	0.1659	0.0628	0.0632	**0.0287**	0.1935	0.2561	0.1544	0.1546	**0.1526**
	.9	0.0813	0.2461	0.0451	0.0451	**0.0282**	0.1984	0.3611	0.1244	**0.1237**	0.1244
100	.1	0.0558	0.0183	**0.0100**	**0.0100**	0.0278	0.1723	0.0228	0.0097	0.0097	**0.0073**
	.2	0.0266	**0.0115**	0.0124	0.0123	0.0254	0.1072	0.0231	0.0154	0.0152	**0.0133**
	.3	0.0183	**0.0136**	0.0187	0.0187	0.0302	0.0630	0.0298	0.0304	0.0303	**0.0291**
	.4	0.0295	0.0291	0.0278	**0.0277**	0.0343	0.0487	**0.0519**	0.0531	0.0529	0.0526
	.5	0.0475	0.0540	0.0381	0.0381	**0.0376**	**0.0675**	0.0880	0.0808	0.0806	0.0810
	.6	0.0620	0.0840	0.0474	0.0474	**0.0398**	0.1104	0.1394	0.1102	0.1096	**0.1093**
	.7	0.0671	0.1204	0.0508	0.0509	**0.0382**	0.1543	0.2038	0.1310	0.1309	**0.1304**
	.8	0.0610	0.1666	0.0459	0.0460	**0.0289**	0.1694	0.2870	0.1336	0.1335	**0.1323**
	.9	0.0440	0.2347	0.0292	**0.0291**	0.0123	0.1364	0.3884	0.0977	**0.0976**	**0.0976**
250	.1	0.0373	0.0049	0.0046	0.0041	**0.0037**	0.1438	0.0088	0.0040	0.0040	**0.0036**
	.2	0.0114	0.0056	0.0111	0.0064	**0.0062**	0.0753	0.0139	0.0093	0.0093	**0.0091**
	.3	0.0167	0.0177	0.0217	**0.0123**	0.0124	0.0409	0.0298	0.0238	0.0236	**0.0241**
	.4	0.0311	0.0362	0.0362	0.0212	**0.0210**	0.0512	0.0632	0.0461	**0.0460**	0.0463
	.5	0.0430	0.0576	0.0518	0.0309	**0.0307**	0.0927	0.1102	0.0733	0.0732	**0.0733**
	.6	0.0504	0.0821	0.0652	0.0391	**0.0384**	0.1327	0.1690	0.1013	0.1012	**0.1007**
	.7	0.1515	0.1110	0.0704	0.0420	**0.0411**	0.1540	0.2366	0.1206	0.1209	**0.1205**
	.8	0.0436	0.1474	0.0621	0.0361	**0.0356**	0.1475	0.3182	0.1201	0.1200	**0.1194**
	.9	0.0249	0.1974	0.0351	**0.0196**	**0.0196**	0.0987	0.4137	**0.0792**	**0.0792**	0.0798

Table 2: The estimated mean square errors (MSEs) of $\hat{\phi}_{MLE}$, $\hat{\phi}_{OLS}$, $\hat{\phi}_{DE}$, $\hat{\phi}_{SA}$, and $\hat{\phi}_{DESA}$ when percentage of additive outliers is 10%. Numbers written in bold are the least MSEs obtained for each parameter with magnitude of outliers 3σ and 5σ.

Figure 2 shows the graphs of MSEs when percentage of outliers if 5%, magnitude of outliers effect is 3σ and lengths of series are 25, 50, 100 and 250. Note that the resulting MSEs of $\hat{\phi}_{MLE}$, represented by line1, are generally higher when the lengths of series are 25 and 50; lower when length of series is 100; and even lower when length is 250. These observations imply that MSEs obtained by $\hat{\phi}_{MLE}$ consistently get lower as the lengths of the series get longer. Similarly, the OLS estimator, $\hat{\phi}_{OLS}$, obtained lower MSEs (line2) when n is large, particularly for ϕ values lower than 0.5; but, not when ϕ are greater than 0.5. Using OLS, the MSEs go larger as the ϕ values get closer to 1. However, MSEs for $\hat{\phi}_{OLS}$ are lowest at 0.3, 0.4 and 0.5 for shorter series of lengths 25 and 50. For almost all situations, the three algorithms, DE, SA, and DESA have lower MSEs (line3, line 4, and line5, respectively), with DESA having the lowest, especially when n=250.

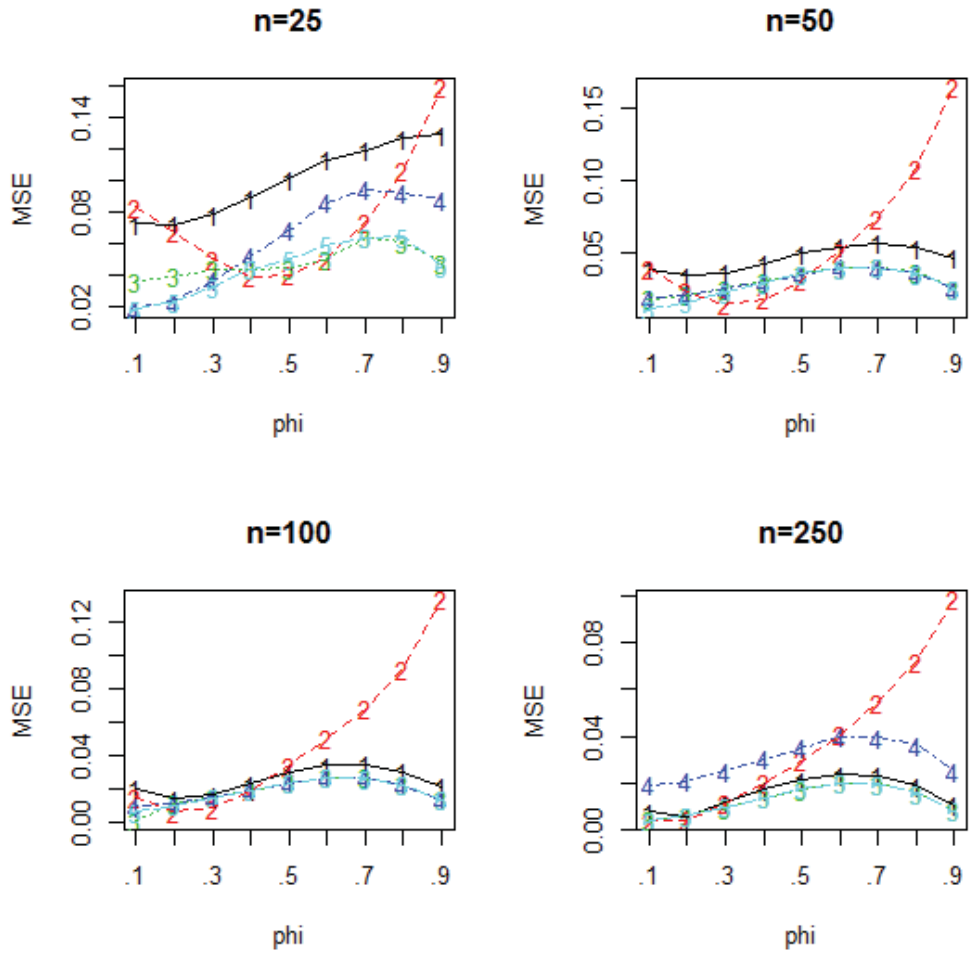

Figure 2: The estimated mean square errors (MSEs) of $\hat{\phi}_{MLE}$, $\hat{\phi}_{OLS}$, $\hat{\phi}_{DE}$, $\hat{\phi}_{SA}$, and $\hat{\phi}_{DESA}$ when percentage of additive outliers is 5% and magnitude of AOs effect is $3\sigma_e$. Lines numbered $1, 2, 3, 4$, and 5 correspond to $\hat{\phi}_{MLE}$, $\hat{\phi}_{OLS}$, $\hat{\phi}_{DE}$, $\hat{\phi}_{SA}$, and $\hat{\phi}_{DESA}$, respectively.

Figure 3 is similar to Figure 2, but this time the magnitude of outliers effect is 5σ. Note that the resulting MSEs for the estimator, $\hat{\phi}_{MLE}$ (line1) are getting lower as the length of series is getting longer. However, the MSEs are still higher compared to those of DE, SA, and DESA. Also, $\hat{\phi}_{OLS}$ (line2), performs best when n=25 at wider range of phi values from 0.3 up to 0.8. When length of series is increase to 50, 100 and 250, OLS resulted to greater MSEs when ϕ values are greater than 0.5. Unfortunately, MSEs using OLS consistently get higher as ϕ values reach 1. The three algorithms, DE, SA, and DESA have lower and relatively closer MSEs, with SA having the lowest MSEs especially when when n=250.

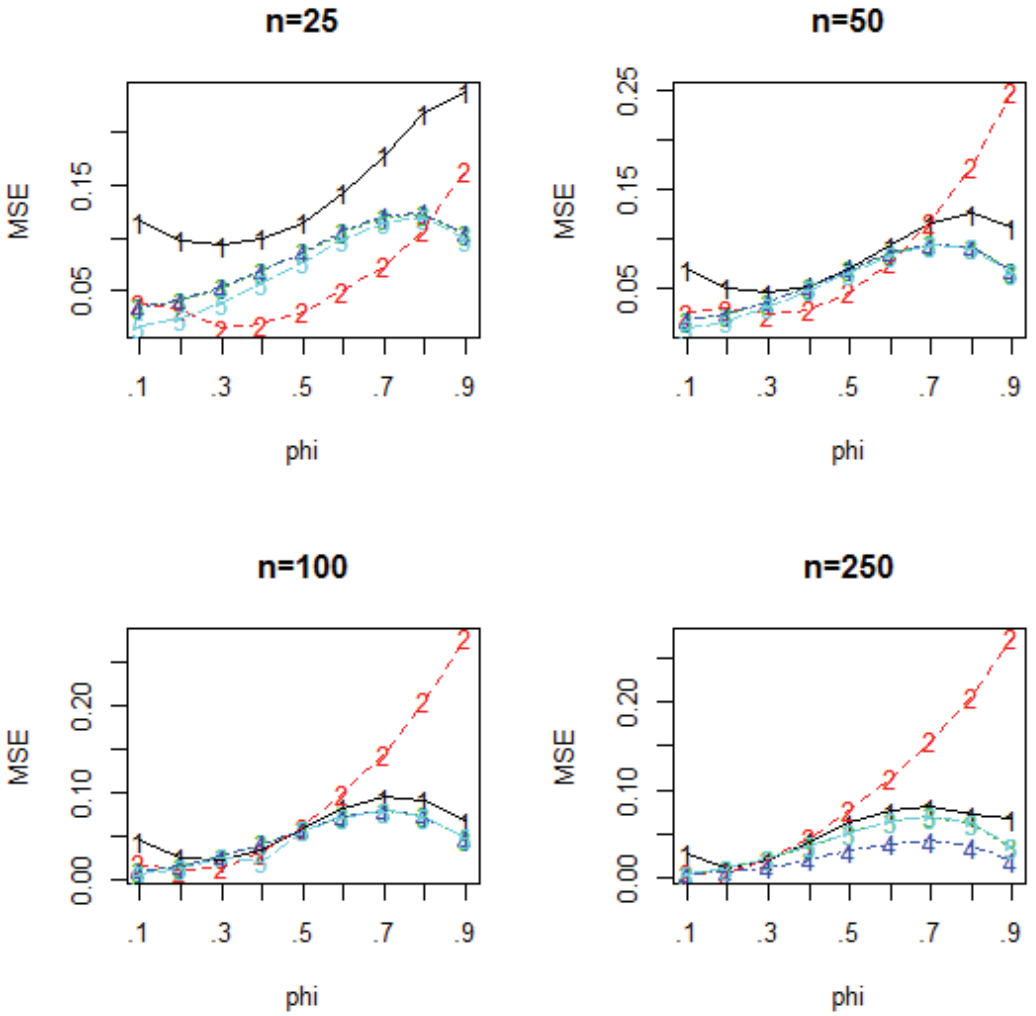

Figure 3: The estimated mean square errors (MSEs) of $\hat{\phi}_{MLE}$, $\hat{\phi}_{OLS}$, $\hat{\phi}_{DE}$, $\hat{\phi}_{SA}$, and $\hat{\phi}_{DESA}$ when percentage of additive outliers is 5% and magnitude of AOs effect is $5\sigma_e$. Lines numbered $1, 2, 3, 4$, and 5 correspond to $\hat{\phi}_{MLE}$, $\hat{\phi}_{OLS}$, $\hat{\phi}_{DE}$, $\hat{\phi}_{SA}$, and $\hat{\phi}_{DESA}$, respectively.

Figure 4 shows the MSEs obtained from the five algorithms, when percentage of outliers is increased to 10% and magnitude is 3σ. Note that when n=25, OLS has the highest MSEs; but these MSEs consistently get lower as the length of the series increases. However, the MSEs are still higher compared to those of DE, SA and DESA, even when n is 250. Also, $\hat{\phi}_{OLS}$ (line2), performs best when n=50 and when ϕ values are 0.3 up to 0.5. When length of series was increased to 50, 100 and 250, OLS resulted to greater MSEs when ϕ values were greater than 0.5 and consistently went higher as ϕ reached 1. In almost all situations, DESA has the lowest MSEs. However, differences in MSEs for the DE, SA and DESA and not significantly visible when n=250.

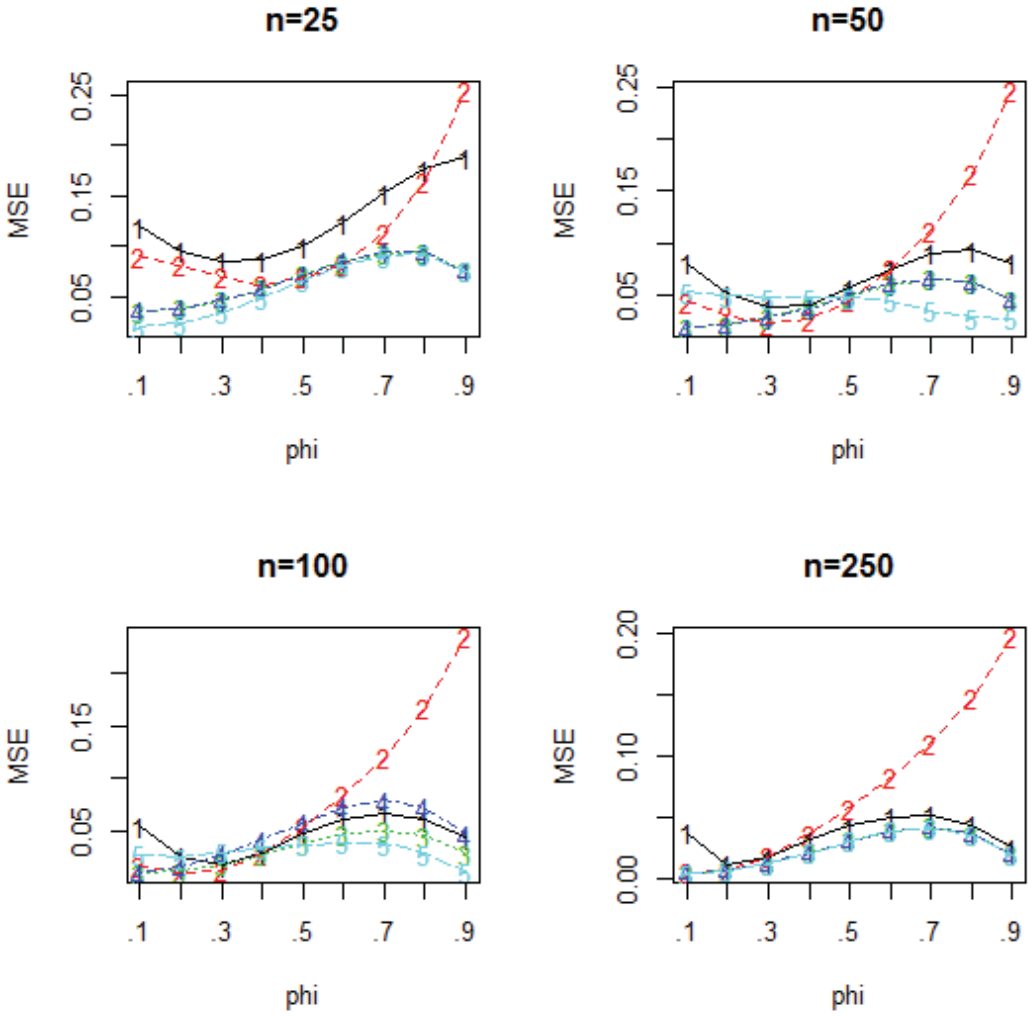

Figure 4: The estimated mean square errors (MSEs) of $\hat{\phi}_{MLE}$, $\hat{\phi}_{OLS}$, $\hat{\phi}_{DE}$, $\hat{\phi}_{SA}$, and $\hat{\phi}_{DESA}$ when percentage of additive outliers is 5% and magnitude of AOs effect is $5\sigma_e$. Lines numbered $1, 2, 3, 4$, and 5 correspond to $\hat{\phi}_{MLE}$, $\hat{\phi}_{OLS}$, $\hat{\phi}_{DE}$, $\hat{\phi}_{SA}$, and $\hat{\phi}_{DESA}$, respectively.

Figure 5 shows that when both percentage and magnitude of outliers effect were increased to 10% and 5σ, respectively, MSEs for $\hat{\phi}_{MLE}$ (line1) were generally higher when ϕ values are 0.1, 0.2 and 0.3; while, $\hat{\phi}_{OLS}$ (line2) obtained the highest MSEs at phi values greater than 0.5. For almost all values of ϕ, and even with all the different lengths of series, the three algorithms, DE, SA, and DESA, obtained MSEs (line3, line4, and line5, respectively) which are relatively lower compared to those of MLE and OLS. Though hardly visible, DESA have the lowest MSEs for almost all situations.

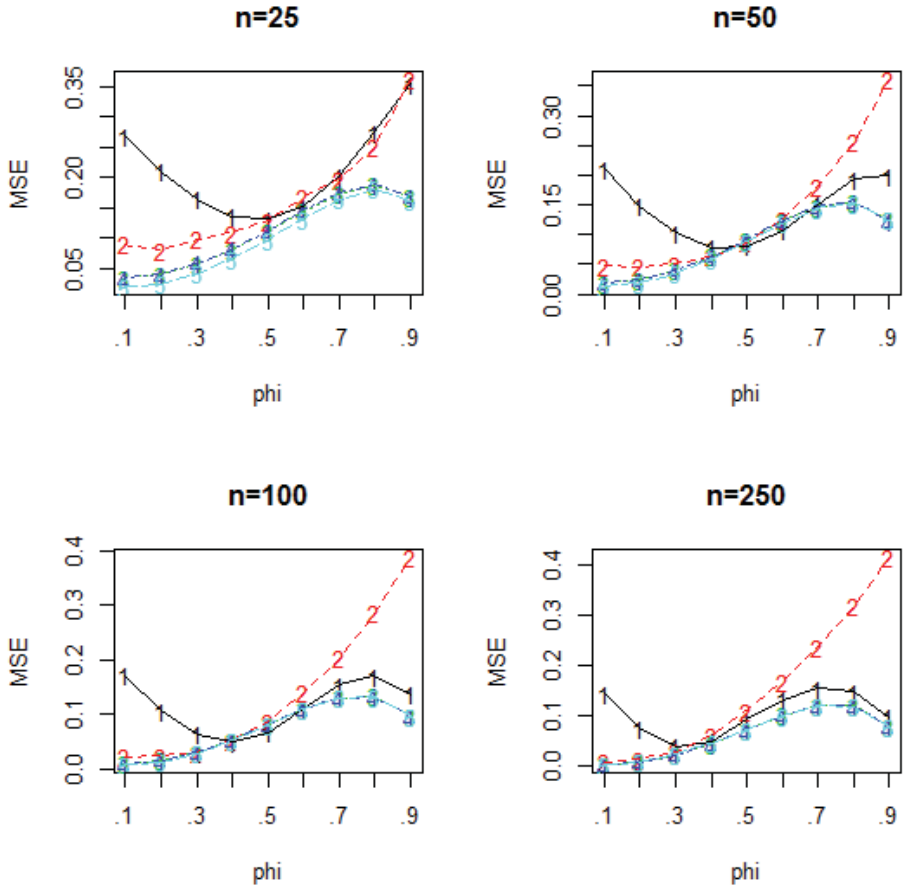

Figure 5: The estimated mean square errors (MSEs) of $\hat{\phi}_{MLE}$, $\hat{\phi}_{OLS}$, $\hat{\phi}_{DE}$, $\hat{\phi}_{SA}$, and $\hat{\phi}_{DESA}$ when percentage of additive outliers is 10% and magnitude of AOs effect is $5\sigma_e$. Lines numbered $1, 2, 3, 4$, and 5 correspond to $\hat{\phi}_{MLE}$, $\hat{\phi}_{OLS}$, $\hat{\phi}_{DE}$, $\hat{\phi}_{SA}$, and $\hat{\phi}_{DESA}$, respectively.

4 DESA ALGORITHM ON WIND SPEED DATA

In this section, we applied the proposed estimator to wind speed data taken from Security Control and Data Acquisition (SCADA) wind farm. The real data set is a 10-minute average wind speed recorded 3 consecutive days in June 2013 giving a total of 432 observations. The time series plot, the ACF and PACF, as shown in Figure 6, suggest that AR(1) model is suitable. Test of stationarity of the series is done using Dickey-Fuller's test. In order to assess the performance of our proposed DESA algorithm in fitting a first-order autoregressive model to the actual data, we also obtained parameter estimates using MLE, OLS, DE and SA. The parameter estimates and their corresponding square errors are shown in Table 3.

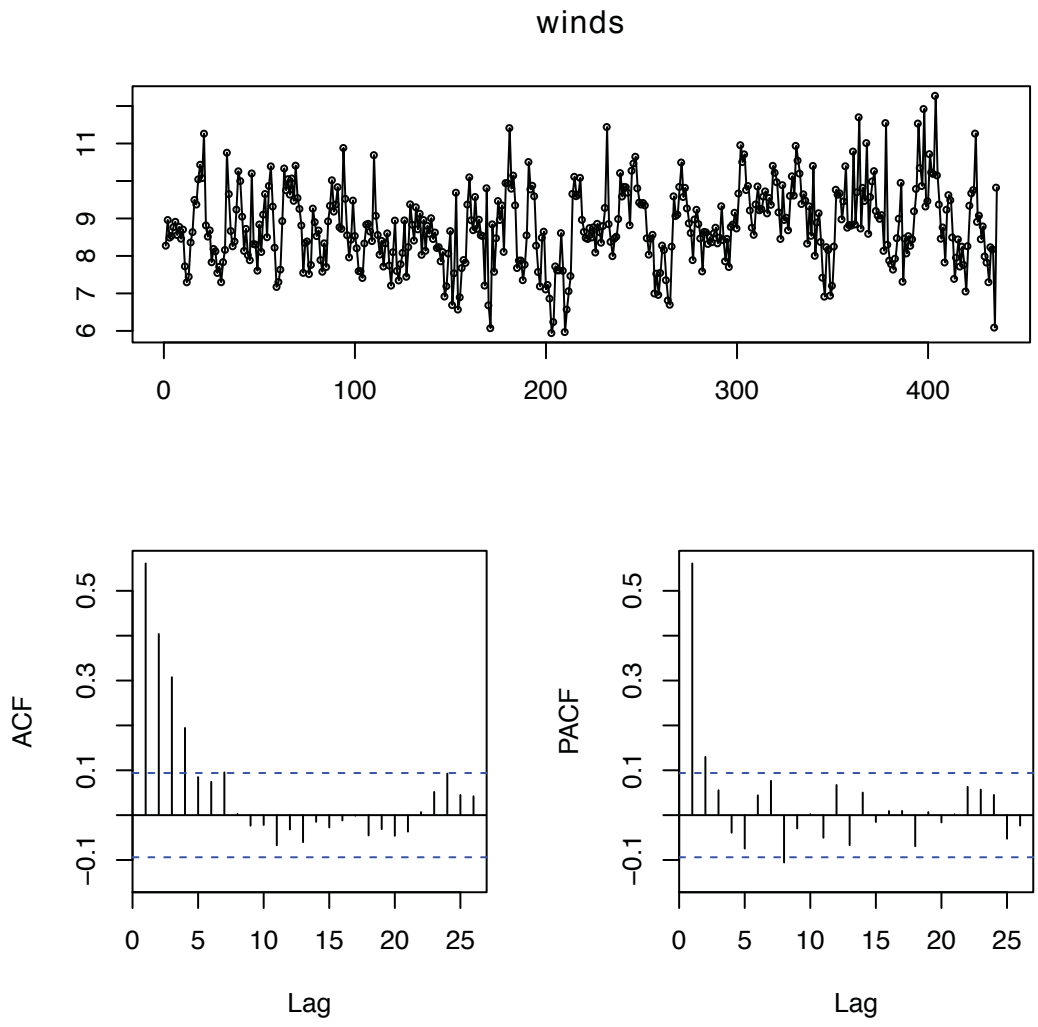

Figure 6: Three days 10-minute average wind speed.

As presented in Table 3, DESA has the lowest standard error (SE). Thus, we may model the three days 10-minute average wind speed as AR(1) process, with $\hat{\phi}_1 = 0.5718$. This provides an

improved model fit by about 50.37%, 50.52%, 1.7%, or 1.56% decrease in standard error, when using any of $\hat{\phi}_{MLE}$, $\hat{\phi}_{OLS}$, $\hat{\phi}_{DE}$, and $\hat{\phi}_{SA}$, respectively. The application to a real data on wind speed demonstrates that DESA provides good and even better alternative parameter estimates for the fitting problem. However, estimates obtained by DE and SA algorithms are comparable to DESA algorithms.

Estimator	Estimate, $\hat{\phi}_1$	SE
$\hat{\phi}_{MLE}$	0.5614	0.03961
$\hat{\phi}_{OLS}$	0.5628	0.03973
$\hat{\phi}_{DE}$	0.5705	0.02000
$\hat{\phi}_{SA}$	0.5707	0.01997
$\hat{\phi}_{DESA}$	0.5718	0.01966

Table 3: The parameter estimates, $\hat{\phi}_1$, and standard errors (SEs) obtained by MLE, OLS, DE, SA and DESA for the 10-minute average wind speed data.

5 CONCLUSIONS

The addition of a new strategy based on DE and the incorporation of the SA algorithm for a selection strategy makes DESA a robust optimization algorithm. The proposed hybrid algorithm obtained acceptable solutions particularly for AR(1) processes with unknown drift and additive outliers. DESA has shown reliability in finding global minimum of the reference problem sets. Simulation results have shown that DESA provides MSE lower than those of MLE and/or OLS for almost all situations. It also obtained comparable performance on fitting AR(1) models to the 10-minute average wind speed.

REFERENCES

[1] R. S. Tsay, Time series model specification in the presence of outliers. *Journal of American Statistical Association*, **81**, 132–141, 1986.

[2] C. Chen, L.M. Liu, Forecasting time series with outliers. *Journal of Forecasting*, **12**, 13–35, 1993.

[3] A.J. Fox, Outliers in time series. *Journal of the time series society*, **34**, 350–363, 1972.

[4] R. S. Tsay, Outliers, level shifts, and variance changes in time series. *Journal of Forecasting*, **7**, 1–20, 1988.

[5] O.H. Bustos, V. J. Yohai, Robust Estimates for ARIMA models. *Journal of the American Statistical Association*, **81**, 155–168, 1986.

[6] R.H. Glendinning, Determining the order of an ARMA model from outlier contaminated data. *Communications in Statistics*, **27**, 13–40, 1998.

[7] G. Masarotto, Robust Identification of Autoregressive Moving Average Models. *Applied Statistics*, **36**, 214–220, 1987.

[8] B. Abraham, A. Chuang, Expectation-maximization algorithms and estimation of time series models in the presence of outliers. *Journal of Time Series Analysis*, **14**, 221-234, 1993.

[9] M. Mitchell, *An Introduction to Genetic Algorithms*. The MIT Press, 1998.

[10] J.H. Holland, *Adaptation in Natural Systems*. University of Michigan Press, Ann Arbor, 1975.

[11] D. Corne, M. Dorigo, F. Glover, *New Ideas in Optimization*. McGraw Hill, 1999.

[12] T.C. Hsia, Simple Robust Schemes for Space Control of Robot Manipulators. *Int. J. of Robotics and Automation*, **9**, 913-167-174, 1994.

[13] O.C. Martin, S.W. Otto, Combining Simulated Annealing with Local Search Heuristics. *Annals of Operations Research*, **63**, 57-75, 1996.

[14] J. Kennedy, R.C. Eberhart, Y. Shi, On the usage of Differential Evolution for Function Optimization. *IEEE Trans. Syst. Man Cybern. A*, **29**, 63-76, 1999.

[15] R. Storn, K. Price, Differential Evolution- a simple and efficient heuristic for global optimisation. *Proceedings of IEEE International Conference on Evolutionary Computation*, **11**, 341-359, 1997.

[16] K.M. Passino, Biomimicry of bacterial foraging for distributed optimization and control. *IEEE control system magazine*, **22**, 52-67, 2002.

[17] F.L. Chu, Forecasting tourism demand with ARMA-based methods. *Tourism Management*, **30**, 740-751, 2009.

[18] K. Ursem, P. Vadstrup, Parameter identification of induction motors using differential evolution. *Proceedings of IEEE Congress on Evolutionary Computation 2*, 2003.

[19] I. Falco, Automatic classification of hand segmented image parts with differential evolution. *Rothlauf F., et al., (Eds.), Evo Workshops, (LNCS 3907)*, **30**, 403-414, 2006.

[20] M.G.H. Omram, A.P. Engelbrecht, A. Salman, Differential evolution methods for unsupervised image classification. *Proceedings of IEEE Congress on Evolutionary Computation 2*, **2**, 966-973, 2005.

[21] S. Paterlini, T. Krink, Differential evolution and particle swarm optimization in partitional clustering. *Computational Statistics Data Analysis*, **5**, 1220-1247, 2005.

[22] B.V. Babu, S.A. Munawar, Differential evolution strategies for optimal design of shell-and-tube heat exchangers. *Chemical Engineering Science*, **62**, 2739-3720, 2007.

[23] A.C. Nerachou, S.L. Omirou, Differential evolution for sequencing and scheduling optimization. *Journal of Heuristics*, **12**, 395-411, 2006.

[24] M. Rout, B. Majhi, R. Majhi, G. Panda, Novel stock market prediction using a hybrid model of adaptive linear combiner and differential evolution. *Proceeding of 2nd International Conference on Recent Trends in Information, Telecommunication and Computing* **2**, 187-191, 2011.

[25] X.-S. Yang, Firefly algorithms for multimodal optimization. *Stochastic Algoritms and Applications, SAGA 2009, Lecture Notes in Computer Sciences*, **5792**, 169-178, 2009.

[26] E.H. Aarts, J.H.M. Korst, *Simulated Anneaing and Boltzmann Machines*. John Wiley and Sons Inc., 1989.

[27] S. Kirkpatrick, C.D. Gelatt,Jr., M.P. Vecchi, Optimization by Simulated Annealing. *Science, New Series*, **220**, 671-680, 1983.

[28] R.C. Addawe, J.M. Addawe, E.P. Adorio, J.C. Magadia, DESA: A Hybrid Optimization Algorithm for High Dimensional Functions. *Proceedings of the IASTED International Conference on Control and Applications (CA 2006)*, Montreal, Quebec, Canada, May 24-26, 2006.

[29] T. Rogalsky, R.W. Derksen, S. Kocabiyik, Differential evolution in aerodynamic optimization *Proc. 46th Annu. Conf. of Can. Aero-naut. Space Inst.*, Montreal,QC,Canada, May 1999, pp. 29-36.

[30] I.L. Lopez Cruz, L.G. Van Willigenburg, G. Van Straten, Efficient Differential algorithm in multimodal optimal control problems. *Applied Soft Computing*, **3**, 97-112, 2003.

[31] L. Ingber, Very Fast Simulated Re-annealing. *Mathematical Computation Modelling*, **12**, 967-973, 1989.

[32] J.R. Parker, Simulated Annealing for Fitting Linear Combinations of Gaussians to Data. *Computing*, **65**, 291-312, 2000.

[33] M. Locatelli, Convergence Properties of Simulated Annealing for Continuous Global Optimization *Applied Probability Trust*, **33**, 1127-1140, 1996.

[34] D. Böning, *Computer-Assisted Analysis of Mixtures and Applications*. John Wiley and Sons Inc., 1989.

[35] A.K. Jain Statistical Pattern Recognition: A review. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, **22**, 4-36, 2000.

[36] J. Zhang, A. Sanderson, *Adaptive Differential Evolution*. Springer-Verlag, 2009.

[37] I. Chang, G.C. Tiao, C. Chen, Estimation of time series parameters in the presence of outliers. *Technometrics*, **30**, 193–204, 1988.

[38] K.V. Price, R.M. Storn, J.A. Lampinen, *Differential Evolution - A practical Approach to Global Optimization*. Springer-Verlag, 2006.

[39] K. Mullen, D. Ardia, D. Gil, D. Windover, J. Cline, DEoptim: An R Package for Global Optimization by Differential Evolution. *Journal of Statistical Software*, **40**, 1-26, 2011.

[40] Y. Xiang, S. Gubian, B. Suomela, J. Hoeng, Generalized Simulated Annealing for Global Optimization: the GenSA Package for R. *The R Journal* **5**, 13-29, 2013.