Nonlinear Least Square Regression by Adaptive Domain Method With Multiple Genetic Algorithms

Satoshi Tomioka, Shusuke Nisiyama, and Takeaki Enoto, *Member, IEEE*

***Abstract*—In conventional least square (LS) regressions for nonlinear problems, it is not easy to obtain analytical derivatives with respect to target parameters that comprise a set of normal equations. Even if the derivatives can be obtained analytically or numerically, one must take care to choose the correct initial values for the iterative procedure of solving an equation, because some undesired, locally optimized solutions may also satisfy the normal equation.**

In the application of genetic algorithms (GAs) for nonlinear LS, it is not necessary to use normal equations, and a GA is also capable of avoiding localized optima. However, convergence of population and reliability of solutions depends on the initial domain of parameters, similarly to the choice of initial values in the abovementioned method using the normal equation. To overcome this disadvantage of applying GAs for nonlinear LS, we propose to use an adaptive domain method (ADM) in which the parameter domain can change dynamically by using several real-coded GAs with short lifetimes.

Through an example problem, we demonstrate improvements in terms of both the convergence and the reliability by ADM. A further merit in the proposed method is that it does not require any specialized knowledge about GAs or their tuning. Therefore, the nonlinear LS by ADM with GAs are accessible to general scientists for various applications in many fields.

***Index Terms*—Adaptive domain, convergence, real-coded genetic algorithm (GA), multimodal problem, nonlinear least square (LS) regression, reliability.**

I. Introduction

LEAST SQUARE (LS) REGRESSION techniques [1] are useful to determine best fit parameters for simulation models in many engineering and scientific fields. The simulation model is based on a function that includes some variable (tunable) parameters, which are determined by finding the minimal sums of squares of residuals between the function and experimental data. In conventional techniques, the parameters determined from LS residuals are derived from a set of equations called *normal equations* in which the derivative of the sum of squares' residual with respect to each parameter equals zero. When the function to be estimated is nonlinear, there are two main disadvantages in this technique. First, it is not easy to obtain the derivatives analytically. Second, in addition to the correct LS solution, certain locally optimized solutions may also satisfy the normal equation; therefore, the choice of initial values in a parameter space greatly affects the reliability of the solution [1], [2].

A genetic algorithm (GA) is one of evolutionary computations that were first applied by Rechenberg [3] and Holland [4]. It imitates the process of biological evolution in nature, and it is classified as one type of random search techniques. Various candidate solutions are tracked during the search procedure in the system, and the population evolves until a candidate of solution fitter than a predefined criteria emerges. In most GAs [5], a candidate solution, called an individual, is represented by a binary string, i.e., a series of 0 or 1 elements. Each binary string is converted into a phenotype that expresses the nature of an individual, which corresponds to the parameters to be estimated in the LS problem. The GA evaluates the fitness of each phenotype with respect to cost functions such as the residual in an LS problem. A general GA involves two major genetic operators; a crossover operator to increase the quality of individuals for the next generation, and a mutation operator to maintain diversity in the population. During the operation of a GA, individual candidate solutions are tracked in the system as they evolve in parallel. Therefore, GA techniques provide a robust method to prevent against final results that include only locally optimized solutions.

The following are the main merits for applying a GA to an LS problem. It does not require any effort to calculate the derivatives. Furthermore, no special techniques are required to solve the nonlinear equation. Also, it is not required to make a selection of initial points in the space of parameters to be estimated. Only coding to represent the LS residuals is necessary.

On the other hand, there are some drawbacks concerning the use of GA for general applications. The first disadvantage is with respect to computational time. Since a GA is based on a random search technique, computational time required by a GA is higher than that of a conventional LS based on algebraic solver. Furthermore, it is known that GA methods are effective in generating convergence of the population in early generations but the search may slow down or stall in later generations. Applications of GAs sometimes result in premature convergence, in which a population concentrates around an undesired local optimum. In this situation, the population, after falling into the local optimum, becomes more and more similar with the increase of generations, and consequently, a global search to escape from local optimum does not work effectively [6]. To avoid this premature convergence some local tunings are applied, such as controlling the pressure of a selection, or the ratio between the number of individuals created by mutation strategy and that created by crossover strategy. Such tunings that depend on the

Manuscript received November 10, 2004; revised June 27, 2005 and November 20, 2005. This research was supported in part by the Ministry of Education, Science, Sports and Culture under Grant-in-Aid for Scientific Research (C), 17560049, 2005.

The authors are with the Graduate School of Engineering, Hokkaido University, Sapporo 060-8626, Japan (e-mail: tom@qe.eng.hokudai.ac.jp; shu@qe.eng.hokudai.ac.jp; tenoto@qe.eng.hokudai.ac.jp).

Digital Object Identifier 10.1109/TEVC.2006.876363

problem under consideration help to maintain the reliability of solutions; however, they also tend to induce delayed convergence due to the tradeoff relationship between reliability and computational performance [6], [7]. To solve this slowing down, there are some well-known techniques, which include switching to methods other than the GA in later generations [7]–[9] and restricting GA parameters such as a mutation amplitude in accordance with increase in generations [6]. However, since early switching intended to achieve fast convergence may again induce results that favor a localized optimum, the correct choice of the timing to switch between different methods is difficult for users who are not familiar with the GA. Furthermore, a GA may result in local optima if incorrect initial settings are provided, such as an excessively narrow definition of solution's domain for the sake of faster convergence. This is similar to the problem of the initial point selection in conventional LS techniques. The second disadvantage is that the parameters to be determined in LS are real numbers and moreover their domains generally are not known. However, the binary coding normally used can represent only discrete numbers. To represent real number parameters efficiently, modified binary coding called Gray coding was applied in early studies [13], [14]. Even in the use of the Gray coding, since both the resolution of a gene and its length are limited, the range of the phenotype of gene is finite. Since both the resolution and the domains are fixed in the coding, less accurate solutions were obtained [6]. Moreover, in problems for which domains of parameters are not known, the binary coding and the Gray coding are not applicable. In many real-number-based techniques proposed during the past decade, it has been demonstrated that by representing physical quantities as genes, i.e., as components of an individual, it is possible to obtain faster convergence and better resolution than by use of binary or Gray coding. A program employing this kind of method is called an "Evolution Program" by Michalewicz [6] or a real-coded GA. In this study, we adopt the real-coded GA.

Additional disadvantages of GAs include the problem concerning domains of parameters to be estimated and the difficulties to control the domain without loss of convergence. In order to solve the former disadvantage, some ideas to change the domain dynamically during the process of a GA have been proposed [10], [11]. By such methods, the details of which are discussed in Section III, it is possible to achieve faster convergence and better accuracy, but these are not applicable when the global solution is not within the initial domain. To overcome this drawback, in this study, we propose a new adaptive domain method (ADM) using multiple independent GAs with short lifetimes. We show here a brief summary of ADM. The ADM invokes several GAs with the information about current ranges of domains of the initial population in each GA. Each GA returns the best individual and its associated cost function within several generations. The ADM decides the new domains by a statistical analysis of all of the results from GAs, and the resulting domain is used when ADM invokes next series of GAs. In the system, the domain of the solutions varies dynamically. The ADM is also advantageous in that there are no sensitive parameters required to control a GA in order to ensure convergence.

We enumerate the requirements to the nonlinear LS by GAs with ADM in order of importance: it should be able to find the global optimum even if the initially defined domain does not include the global optimum; it requires no selection of sensitive GA tuning parameters such as the timing to switch the algorithms or the mutation amplitude, which may be difficult for novice users of GA techniques; and its computational time should not be much higher than that of other methods using GAs. Some readers may wonder about a possible relation between abovementioned requirements and "No Free Lunch (NFL) theorems [12]." These theorems conclude that there is no single algorithm with the best performance for all optimization problems, where performance means the number of times of evaluation of search points. These theorems require the assumption that once a point is evaluated, it is not evaluated again. However, as most GAs, including the GA applied in this paper, do not record the history of searching, it is practically impossible to avoid the revisiting of points. Accordingly, the imperial performance is worse than the predicted ideal performance that is considered the same for any algorithm according to NFL theorems. The ADM changes the search domain according to the results from several GAs, sometimes broadening and other times narrowing it, or sometimes making only small adjustments. In the broadened case, the probability to revisit the same area is reduced. When the domain becomes narrower, the probability is increased but the computational cost to obtain the solution in the domain becomes smaller than the case of larger search space. And in the case where ADM suggests only a minor domain change, GA searches again to obtain more information about the reliability. Thus, the total number of instances of revisiting may be smaller than for general GA problems, although its performance may be worse then that of the ideal performance.

The outline of this paper is as follows. In Section II, the difficulties of using LS for nonlinear problems and the potential application of GAs to problems of regression are discussed. Next, application of ADM using multiple GAs is proposed in Section III, and subsequently, the real-coded GAs and their genetic strategies are discussed in Section IV. The applicability of ADM is demonstrated for simulation data and results are verified by actual experimental data in Section V, and finally, conclusions are drawn regarding ADM in Section VI.

II. Nonlinear LS Regression

In order to understand the nature of a system, it is generally important to represent the system as a model expressed by analytical functions. To determine the best modeling functions, it is common to employ statistical processing of experimental data by such methods as LS regression.

Consider a problem to fit observations (x_i, y_i), where $i \in \{1, \cdots, N\}$, into an estimating function that represents the model with several parameters $a_m, m \in \{1, \cdots, M\}$

$$y(x) = y(x; a_1, \cdots, a_M) \tag{1}$$

where parameters a_m are not given and the form of the function y is given. The purpose of fitting is to find the parameters that comprise a maximum-likelihood function. Every observation y_i is not coincident with $y(x_i)$ due to errors. When the distribution

of errors conforms to the normal distribution (Gaussian distribution), the maximum-likelihood estimator is called an LS solution. The LS solution satisfies the condition that a sum of the squares of residuals between the observations y_i, and the estimations $y(x_i; a_1, \ldots, a_M)$ is minimized [1]

$$\text{minimize in space of } \mathbf{a}: \quad \chi^2(\mathbf{a}) = \sum_{i=1}^{N} \left(\frac{y_i - y(x_i; \mathbf{a})}{\sigma_i} \right)^2 \tag{2}$$

where $\mathbf{a}$ shows an M-vector which consists of the parameters a_m, and σ_i represents the standard deviation of the data i. At this minimum point, every partial derivative of the χ^2 with respect to a_m vanishes

$$\frac{\partial \chi^2}{\partial a_m} = 0. \tag{3}$$

The set of these equations is called the *normal equation*.

If the estimated function is linear with respect to the parameters, such as $y(x; \mathbf{a}) = \sum X_m(x) a_m$, then the derivatives can be easily derived by hand, and the solution can be obtained by solvers of simultaneous equations such as Gaussian elimination or LU decomposition techniques.

For nonlinear estimated functions, however, the derivatives cannot be calculated easily. Even if one can obtain the derivatives numerically, there is another problem that the normal equation shown in (3) may have several solutions. Of these solutions, the one that satisfies an LS condition is unique, and the others correspond to local minima or local maxima of the residual $\chi^2(\mathbf{a})$ in (2). If we could obtain all of the solutions, then we would be able to decide which one is the LS solution by the consideration of (2). However, most solvers that are based on iterative procedures return only one solution, which depends on the initial point in the iteration. To specify the correct initial point requires empirical insight.

In contrast, when applying a GA to an LS regression, there is no such need to specify the initial value of a parameter. The LS regressions by GAs search for the parameters using (2) directly, and (3) is not applied. Thus, it is not necessary to make any effort to compute the derivatives.

Furthermore, a GA has an advantage to extend from the LS method to *the least median square method* that can exclude some data points with huge measurement error, by only a small modification, namely, to replace the summation operator in (2) with the median operator [14], [15].

As described above, the application of GAs to LS regressions has some important merits. However, there remains another problem. In a similar way to the selection of initial point for iterations based on the normal equation, one must specify the domains of solutions in GA calculations, in order to ensure that estimated results are included. If the selected domain is too wide, the computations could be too costly or time-consuming. On the other hand, a choice of narrow domain is troublesome, since once one specifies an incorrect domain which does not contain an expected solution due to overly optimistic expectations of faster convergence, a global optimal solution may never be obtained. A methodology is proposed to overcome this disadvantage in the next section.

III. Adaptive Domain Method (ADM)

Our aim is the development of a GA-based nonlinear LS regression method in which the following are required, enumerated in order of importance.

- Reliability: A global optimum solution can be obtained with any domain setting of the parameters to be estimated in LS.
- Ease of operation: The estimation system does not require any specialized knowledge about GA tuning, i.e., it is practical and applicable for users who are not familiar with GAs.
- Small computational cost: The system converges efficiently and the computational time is not much higher than that of other techniques using GAs.

Generally, a GA searches for the solution within a predefined space of parameters to be solved. As described in the end of the previous section, it is difficult to correctly define the domain to achieve both reliability and good convergence. In order to solve this problem, we propose a new approach, ADM, to change the domain of parameters dynamically during the estimation process.

The parameter space in binary coding is limited both by the resolution of a gene and by its length or domain, as described in the Section I. All of the candidate solutions are located at discrete points within the parameter domain. In contrast, for the real-coded GA applied in this study, the resolution of the phenotype is as small as the resolution of the computer, and thus it is considered as a continuous real number. Furthermore, some of the genetic operators we applied, the details of which are described in Section IV, such as *the crossover by the extrapolation* and *the fluctuation* can generate individuals located outside of the predefined domain. Therefore, even if the domain defined initially or redefined in an estimation process does not include the desired solution, the GA may successfully find the desired solution. However, the other operators such as *the random mutation* and *the crossover by the interpolation* are only available for searching within the domain, so the chance of correcting for solutions resulting from an inappropriate choice of domain will be small. Therefore, another mechanism to search for optimal solutions outside the domain is required.

Some novel ideas about adaptive domain or range have been proposed. Arakawa *et al.* [10] introduced an adaptive real range technique for binary coded GAs. The outline of their study is as follows. Genotypes at an evenly spaced interval in a finite range are mapped onto phenotypes that are spaced at an uneven interval, using a Gaussian function. For example, consider the integral of a Gaussian function $y(x)$, with an average μ, and a standard deviation σ as $y(x) = \int \exp[-(x-\mu)^2/(2\sigma^2)]\, dx$, where x is a real number that is a phenotype. (In the original paper, the mapping function is defined as a Gaussian function; however, to demonstrate easily, we choose its integral here.) The result $y(x)$ is considered as a genotype, and it is chosen as a binary coding with some scaling y_i, i.e., even interval. The phenotype corresponding to some genotype y_i, is obtained by

an inverse mapping $x_i = y^{-1}(y_i)$. The intervals of x_i are uneven. In the neighborhood of middle range genotypes, the density of phenotypes becomes high, and conversely around both extremes the density becomes low. The average and the standard deviation are computed from the population of the preceding generation, which is generated by GA procedure where the cost function of individual y_i is evaluated via x_i. If the population is concentrated around some point, the standard deviation becomes small, and the search range of phenotype is narrower in the next generation. The change of both the average and the standard deviation leads the mapping function y and the domain is changed. For the real-coded GA, Oyama *et al.* [11] adopted a similar approach. These techniques can dynamically change the parameter domain, and thus can overcome the disadvantage concerning resolution that is involved in binary coding. However, these improved methods have other disadvantages. In spite of the difference in phenotype mapping, x_i, between generations, the genotype, y_i, of the GA determines the population of the next generation. This means that the genotype for an individual is same but the phenotype of that is different between the successive generations; therefore, one cannot change the domain much drastically. Furthermore, once a population is concentrated in the neighborhood of a local optimum, it is difficult to escape from that area. Accordingly, in the abovementioned studies, limitations of the standard deviation of parameters are set initially. The correct setting of the parameters is difficult for users who do not have detailed knowledge about their adaptive range mechanism.

As simple techniques to overcome this handicap, other modifications of genetic operators have become available for effective searches. In general, the population converges rapidly into a narrow space in early generations; however, the convergence speed slows down in later generations. The *nonuniform mutation* [6] works effectively even in later generations. However, this also has the difficult requirement to set the value of a factor to control the mutational amplitude, as discussed in the Section IV. Yi *et al.* [16] proposed a mutation mechanism that applies a local search algorithm based on gradients of fitness. Since these modifications also have the drawback of local optima, it is necessary to tune the factor to control the GA.

There are other ideas to take advantage of the natural tendency of GAs to converge in early generations. Baskar *et al.* [8] proposed a procedure composed of two phases. Specifically, its second phase is implemented by random searches in order to narrow the searching space after attaining a rough convergence of the population by a GA in the first phase. Kwon *et al.* [17] and Djurišić *et al.* [18] studied alternating repetitions of GAs with short lifetimes as a technique for domain narrowing. In Kwon's method, the center of the new domain agrees with the parameter values of the best solution in the preceding short lifetime GA, and the domain width is redefined at a reduced size by multiplying a given factor less than unity. In Djurišić's method, the bounds of the new domain are computed by a relaxation technique, in which the current bounds and an average of the final population in the short lifetime GA are considered. Here, the bounds of new domain are defined as an interpolation between the current bound and the average. Since these methods were developed only for the sake of narrowing the domain; for modeling based on such methods, there is still no easy way to avoid the possibility of local optimum solutions.

In this paper, we propose the application of multiple GAs to enable a change of domain in order to overcome these disadvantages, namely, the lack of an effective method to escape from local optimum solutions, a limitation of speed to change of domain, and required use of factors that may be unfamiliar for general users.

A. Outline of ADM Using Multiple GAs With Short Lifetimes

Consider the case where several local optimum solutions are included in a search domain. When executing a GA for this problem, the GA will find a result as either a local optimum or a desired solution. The reliability, namely, the probability that the result is the desired solution, may be estimated by the fitness of the result. However, it is difficult to verify the confidence of such a result. We attempt to use another measure of the reliability. Trying additional runs with different random series, one can determine the reliability of these solutions. If the results agree, the probability that the results are the desired solution is greater. Otherwise, it is concluded that they include some local optima or they have not yet fully converged.

In real-coded GA, the phenotype of an individual consists of several components corresponding to the parameters to be estimated in the problem. In other words, the phenotype shows a point in the space defined by the axes of the parameters. First, we consider a parameter on one axis. Some GAs with different random series will return different solutions each of which has the largest fitness in the respective final population. If the surviving parameter values are concentrated, with respect to the axis of the solutions, in the neighborhood of a particular point on the axis, then these parameter values have greater reliability, i.e., the probability that the surviving parameter values are located around the desired solution's parameter value is relatively high. In this case, it is possible to narrow the domain with respect to this parameter to accelerate the convergence. Otherwise, the domain cannot be reduced, and conversely, it may be advisable to expand the domain in the worst case. Considering this nature, we propose a new method to vary the domain of the parameters dynamically, by applying several GAs with short lifetimes as ADM.

The relation between ADM and multiple GAs is shown in Fig. 1. Each GA computes not only the best estimation but also its fitness. An estimation with larger fitness is interpreted to be more reliable, so the fitness can be used as a relative weighting of the estimation. All GAs run independently because of different random series, and each will return a different estimation; some will be around the desired solution and others will be around local optimum solutions. Since the weighting of the former is larger than the latter, a redefinition of the domain is expected to be biased to the side of the desired solution. After the average and the standard deviation are computed from the results of all GAs, the new domain is updated based upon the average and the standard deviation. These procedures are repeated until the fitness of the best individual satisfies all convergence conditions.

When a desired solution is not included in results, the distribution of surviving results will be expected not to concentrate

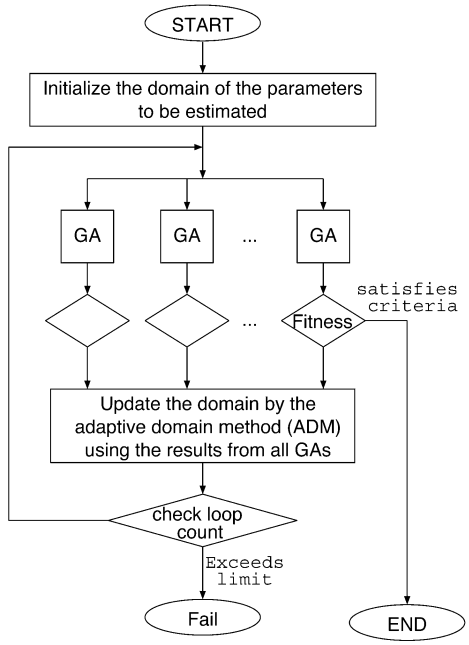


Fig. 1. Method of the adaptive domain using multiple GAs with short lifetimes.

in one particular area. In this case, the ADM will not narrow the parameter domain, and it may conversely widen the domain. If the distribution is concentrated into an area around a particular local optimum, the ADM may not work properly. However, this is believed to be a rare case for the following reasons: all the GAs should not return similar solutions because of use of random seeds; the estimations should not fall into local optimum perfectly because of short lifetimes; each GA has some facilities that can search outside the domain, namely, *crossover by extrapolation* and *fluctuation*, which will be explained in detail in Section IV. When one chooses a sufficiently wide domain to contain some local optimum solutions, this case will be completely prevented. Of course it is best, however, to select the domain that is certain to include the desired solution. An example of this case is shown in the section concerning numerical tests (Section V).

We discussed the merits in the use of multiple GAs. However, the use of multiple GAs induces an increase of computational cost. Fortunately, GAs with ADM do not demand high computational performance to attain convergence, because the convergence of solutions is mainly achieved by ADM rather than by GAs. Therefore, we are not excessively concerned as to whether the result of a GA is the desired solution or not. The requirement imposed upon the GAs from the ADM is only the ability to find *near-optimal individuals*, i.e., the individuals in the vicinity of either a local optimum or a global optimum, which need not be limited to only global optimum solutions. Moreover, a GA generally exhibits rapid convergence of a population in early generations. This means that the small area around either the desired solution or the local minimum can be found in early generations. Thus, since each GA need not converge by itself, the lifetime of the GA (i.e., the total number of generations to compute) can be shorter. In contrast, if the total number of generations is large, the probability that the GA returns only a particular local optimum increases. In this case, the solutions from every GA may be concentrated around the local optimum, and finally, ADM may not work effectively. The lifetime of each GA is a predefined quantity which affects both the reliability of solutions and the computational cost, and its optimum tuning may depend on the nature of problems under consideration. However, we think the setting of this factor is not so difficult for unfamiliar users about GAs.

B. Implementation of ADM

The parameter domain redefinition algorithm proposed in this study is as follows. The best individual from the qth GA, $\hat{\mathbf{a}}^q$, which consists of the parameters $\hat{a}_m^q$, $m \in \{1, \ldots, M\}$ has an evaluation value $E(\hat{\mathbf{a}}^q)$ as an error defined as the square root of the least mean square of the residual as follows:

$$E(\hat{\mathbf{a}}^q) = \left(\frac{1}{N} \sum_{i=1}^{N} \left(\frac{y_i - y(x_i; \hat{\mathbf{a}}^q)}{\sigma_i} \right)^2 \right)^{\frac{1}{2}} \tag{4}$$

where $(x_i, y_i), i \in \{1, \ldots, N\}$ are the measurement data, σ_i is the standard deviation of measurements, which are obtained by several repetitions of experiments or by a theoretical error distribution of the experimental setup, and $y(\,)$ represents the function to be fitted.

Suppose that an error with respect to mth parameter is proportional to the evaluation value $E(\hat{\mathbf{a}}^q)$

$$\Delta_m^q \propto E(\hat{\mathbf{a}}^q). \tag{5}$$

Also, assuming that the samples $\hat{a}_m^q$ are distributed around the average of the parameter with normal distribution, one can write the likelihood function as follows:

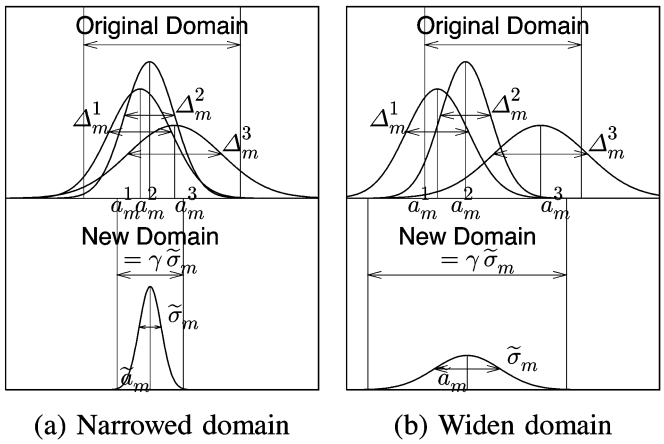
$$L(\tilde{a}_m \hat{a}_m^q, \Delta_m^q) = \frac{1}{\sqrt{2\pi}\Delta_m^q} \exp\left(-\frac{(\tilde{a}_m - \hat{a}_m^q)^2}{2\left(\Delta_m^q\right)^2} \right) \tag{6}$$

where $\tilde{a}_m$ is the most likely solution. The overall likelihood function from all of the best results can be presented as a joint distribution that is given by the product of all likelihood functions

$$\begin{aligned} L_m &= \prod_{q=1}^{N_q} L(\tilde{a}_m \hat{a}_m^q, \Delta_m^q) \\ &= \left(\prod_{q=1}^{N_q} \frac{1}{\sqrt{2\pi}\Delta_m^q} \right) \exp\left[-\frac{1}{2} \sum_{q=1}^{N_q} \frac{(\tilde{a}_m - \hat{a}_m^q)^2}{\left(\Delta_m^q\right)^2} \right] \end{aligned} \tag{7}$$

where N_q is number of GAs. The solution that maximizes the likelihood function can be obtained analytically as a weighted average

$$\tilde{a}_m = \sum_{q}^{N_q} w_q \hat{a}_m^q \tag{8}$$



(a) Narrowed domain (b) Widen domain

Fig. 2. Determination of the new domain. Each plot in the upper figures shows the normal distribution, expressed in terms of both the center a_m^q and the width Δ_m^q, which are obtained from the best estimation from each GA with the original domain. The lower plots show the normal distributions of the center points $\tilde{a}_m$ and the width $\tilde{\sigma}_m$. The new domain is determined by this distribution. The difference between the figures on both sides is the degree of dispersion of the results from GAs.

where w_q represents the weighting factor for averaging

$$w_q = \frac{1/(\Delta_m^q)^2}{\sum_q^{N_q} 1/(\Delta_m^q)^2}\,. \tag{9}$$

Similarly, the variance with respect to the mth parameter $\tilde{\sigma}_m$, can be obtained by the weighting factor

$$(\tilde{\sigma}_m)^2 = \sum_q^{N_q} w_q(\hat{a}_m^q - \tilde{a}_m)^2. \tag{10}$$

The new domain is determined from the average and the variance

$$\begin{pmatrix} \tilde{a}_m^{\max} \\ \tilde{a}_m^{\min} \end{pmatrix} = \tilde{a}_m \pm \gamma_m \tilde{\sigma}_m \tag{11}$$

where γ_m, a factor used to indicate the confidence interval of the parameter, is defined as the ratio between the width and the standard deviation. When the samples of each parameter obey a normal distribution, the probability that the value is contained in the domain is 0.95 in the case of $\gamma_m \simeq 1.96$, and also it reaches 0.99 in the case of $\gamma_m \simeq 2.58$. Subsequently, in this paper, we refer to this factor as the *width factor*. The aforementioned method to obtain the new domain from the results of several GAs is depicted in Fig. 2. If the results of the GAs are concentrated, the domain width $\gamma_m\tilde{\sigma}_m$ becomes narrower due to the small variance $\tilde{\sigma}_m$. Also, the center $\tilde{a}_m$ is varied.

This approach is dependent on the assumption that the error of a parameter is proportional to the evaluation value in (5), which may induce an error in terms of the computation of the new domain. To reduce this effect, we apply the relaxation method

$$\begin{pmatrix} a'^{\max}_m \\ a'^{\min}_m \end{pmatrix} = \beta_m \begin{pmatrix} a_m^{\max} \\ a_m^{\min} \end{pmatrix} + (1-\beta_m) \begin{pmatrix} \tilde{a}_m^{\max} \\ \tilde{a}_m^{\min} \end{pmatrix} \tag{12}$$

where β_m is a relaxation factor, if $\beta_m = 0$, then relaxation is not in effect, also if $\beta_m = 1$ the domain is not updated.

In the last step of the ADM procedure, the new domain is restricted, if there are the limitations of range due to a physical quantity's own limitations (e.g., the physical quantity must be positive). We refer to this type of limit as a "hard limit" in a later description.

In this section, we explained the details of implementing ADM for LS problems. However, by simply choosing (4) to satisfy (5), ADM may also be applied for other optimization problems, providing that the parameter to be optimized is a real number.

IV. GAs for Real-Valued Variables

In this section, we will explain in some detail about the GA that is invoked by the ADM. The ADM gives several GAs an initial domain of parameters that are to be estimated. The GA output results include both the best individual in the final population and its fitness. The requirement for the GA with ADM is to obtain as quickly as possible *the near-optimal individuals*, i.e., the individuals in the neighbor of either a global optimum or local optima. Even if the solution is a local optimum, it should not create a problem because the GA's robustness against local optima is kept in check by the ADM. In other words, since the ADM searches for the global optimum, the GA itself is not required to find the global optimum.

In contemporary research, beginning with early studies concerning GA, as summarized by Goldberg [5], the gene is typically coded as a binary string, and the individual is expressed as a long one-dimensional binary string by concatenation of all the strings corresponding to genes. However, in the cases where a property is in the form of a real number, the binary coding has disadvantages: it will be impossible to express exact solution because the binary code's resolution is limited, as determined by the string length. In addition, in LS type problems, the coding requires conversion from binary strings (genotypes) to real numbers for representation of some parameters (phenotypes). In spite of these disadvantages, binary coding has been applied in a wide variety of applications and problems, due to the simple and useful genetic operators available for binary coding.

By using a set of genetic operators, it is possible to perform two different functions simultaneously. The first function is to accelerate the convergence toward a population that includes a large number of superior individuals. In some cases, the crossover operator, which exchanges substrings of the two parents with each other in the binary coding, is adopted for this purpose. As an example, we consider a problem of a two parameter space. When one of the parents in the crossover strategy has a superior gene with respect to one parameter and the other parent has a superior gene with respect to another parameter, the probability that the resulting child has two superior genes that are given by both the parents is increased. But this strategy introduces a certain probability for the solution to fall into local optima. The second function is to maintain the diversity of a population, which means that the operator has the ability to avoid the concentration of individuals into a narrow space around local optima. In some cases, the mutation operator plays

a role to provide a method of escaping from local optima. This operation is executed by simply inverting one or more digits in a binary string. Note that these strategies are switched in other cases, and they depend on the implementations of strategies and they also depend on problems under the consideration. If there is another known coding method that has both a tendency toward convergence and a tendency to support diversity of solutions, then such a method might be applied as an of the binary coding. Michalewicz [6] shows several floating-point implementations, which are called evolution programs. He showed that the use of real-coded genes in which genotypes are represented by continuous, real numbers, whose resolution is equal to that of the computer, instead of binary strings is more effective than the binary coding, in terms of both the resolution and computational cost. In recent literature, Herrera *et al.* provide a review with respect to the handling of real-coded parameters [19], and Blanco summarizes a brief list [20].

In the previous section, we provided details about the application of ADM using multiple GAs with short lifetimes. Since the GAs work with different random series and the total number of generations is small, the results from the GAs are generally different. This implies that the diversity of solutions is already considered in nature. On the other hand, the convergence of the population in a GA corresponds to the convergence of the domain in ADM. Therefore, the requirement for each GA in ADM is to obtain *the near-optimal individuals* as quickly as possible, as shown at the beginning of this section. It would be ideal to obtain the global optimum with fast convergence by means of a single GA, but this is difficult because the convergence depends on the problems under consideration, and several GA parameters may require tuning.

The following are genetic strategies we adopted for implementation with real-valued variables. To accelerate the ADM, some of the strategies (i.e., "*crossover by extrapolation*" and "*fluctuation*") have the ability to search outside the initially defined domain, which is fixed for each GA when it is called from the ADM. Others are intended to find *the near-optimal individuals* effectively. Moreover, some parameters and strategies that can serve to control the GAs are discussed next. All of our detailed results are not shown here, but from our numerical experiments we found that the settings of parameters to control the GAs have only a small effect on computational cost and they do not affect to the reliability. This indicates that the convergence and reliability mainly depend on ADM, and one need not pay any attention to choosing the correct parameters to control a GA.

A. Genetic Strategies for Real-Valued Variables

The coding we adopted is the real-coded gene model, in which a gene represents a phenotype parameter without any conversion. The individual is represented as a vector $\mathbf{a}$, the elements of which are floating-type genes $a_m, m \in \{1, \ldots, M\}$.

1) Crossover: The crossover operator adopted in this study is based on "arithmetic crossover" as introduced by Michalewicz [6]. When two parent individuals are denoted as $\mathbf{a}^k = (a_1^k, \ldots, a_M^k)$, $k \in \{1, 2\}$, two offspring $\mathbf{a'}^k = (a'^k_1, \ldots, a'^k_M)$ are reproduced as interpolations of both parents' genes

$$\begin{aligned} a'^1_m &= \lambda a^1_m + (1-\lambda) a^2_m \\ a'^2_m &= (1-\lambda) a^1_m + \lambda a^2_m \end{aligned} \tag{13}$$

where λ is a constant in the system. At first glance, choosing $\lambda = +0.5$ may look like a good idea; however, the vectors of a pair of offspring are pointing to the same point in the parameter space. Wright [13] proposed an additional extrapolative reproduction, which is expressed by $\lambda = -0.5$. We adopt the Wright's crossover with some modification

$$a'^{n_1}_m = \frac{1}{2} a^1_m + \frac{1}{2} a^2_m, \quad a'^{n_2}_m = \frac{3}{2} a^1_m - \frac{1}{2} a^2_m \tag{14}$$

where the indexes of offspring n_1 and n_2 are chosen for each gene using a random number $r \in [0, 1]$ and a given factor $p_c \in [0, 1]$, as follows:

$$(n_1, n_2) = \begin{cases} (1, 2), & r < p_c \\ (2, 1), & r \geq p_c \end{cases}. \tag{15}$$

In the case of $p_c = 1$, always $n_1 = 1$, $n_2 = 2$; therefore, every gene in the individual a'^1 is generated by interpolations and a'^2 is generated by extrapolations. The case of $p_c = 0$ is same as the case of $p_c = 1$ except that the parents are exchanged. In the case of $p_c = 0.5$, the probability of interpolation and that of extrapolation are same in each new individual. Once two parents are selected from the pool of parents, the crossover strategy is applied for every gene in each child, namely, $m = 1, \ldots, M$. The random number for one locus of gene is different from for another locus, so the reproduction method, of interpolation or extrapolation, is different between each gene in an offspring.

The crossover strategy by interpolation increases the likelihood of convergence; in contrast, that by extrapolation acts to maintain the diversity of a population. Note that *the crossover by extrapolation* has the ability to reproduce outside the domain in some cases. The GA does not prevent the generated individual from going outside of the domain.

2) Mutation and Fluctuation: We adopt two mutations. The first is a *random mutation* [6]. One of the genes in an individual is replaced by a random number within the parameter domain, $[a_m^{\min}, a_m^{\max}]$. The gene to be mutated is selected at random, namely m is one of members in $1, \ldots, M$. This mutation operator acts to maintain the diversity.

The second is a simplified version of Mühelenbein's mutation [21] to achieve a good convergence by neighborhood search. The gene of the offspring a'_m is computed according to

$$a'_m = a_m + R \cdot (a_m^{\max} - a_m^{\min}) \cdot r \tag{16}$$

where r is a uniform random number $r \in [-0.5, 0.5]$, and R is the ratio of the mutational amplitude to the parameter domain, which is constant in the system as $R \in [0, 1]$. In this paper, we denote this mutation as "*fluctuation*." Note that the fluctuation

has the ability to enable reproduction of offspring outside the domain, as with *the crossover by the extrapolation*.

There is a better known mutation operator called the *nonuniform mutation* [6], but we do not adopt it. In this nonuniform operator, the strength of mutation becomes smaller with advancing of generations

$$a'_m = \begin{cases} a_m + \Delta(t, a_m^{\max} - a_m), & \text{if } \tau = 0 \\ a_m + \Delta(t, a_m - a_m^{\min}), & \text{if } \tau = 1 \end{cases}$$
$$\Delta(t, y) = y\left(1 - r^{\left(1 - \frac{t}{G_{\max}}\right)^B}\right) \tag{17}$$

where r and τ are random numbers, $r \in [0,1]$, $\tau \in \{0,1\}$, and also t, $G_{\max}$, and B denote the generation number, the maximum generation, and given index to control the mutational amplitude, respectively. Since B and $G_{\max}$ are constant, the mutational amplitude has been determined before a GA runs. Therefore, it is impossible to consider the convergence of a population dynamically. Even worse, predetermination of the constant B is difficult for users who do not know how to select GA parameters. Since this constant is sensitive and influences the stability of the estimation, once this constant is chosen incorrectly, the mutation will not work and the diversity of the population will be lost. The purpose of the *nonuniform mutation* operator is to narrow the parameter space of mutational amplitude in accordance with advancing generations. Since the ADM dynamically changes the domain size, $(a_m^{\max} - a_m^{\min})$, the *fluctuation* achieves this purpose without this difficult and sensitive parameter. Since the fluctuation needs as input only the range of mutation in terms of the ratio to the domain size, it is easier to define than that of the nonuniform operator and it is less sensitive to influence the stability of a search. Thus, in this study, we elected not to adopt the *nonuniform mutation* operator.

3) Selection: No special considerations for the real-coded GAs are necessary concerning implementation of the selection strategy; this case is similar to that of the selection for binary coding. We adopt a proportional selection [5] with a slight modification [22]. Consider the minimum search problem as an example. In this case, an individual with smaller evaluation function should have greater fitness and also larger probability to be a parent. The probabilities are calculated in each generation by using biased evaluation functions. First, all the individuals in the population of size K are sorted with respect to evaluation function $E(\mathbf{a}^k)$ in ascending order. Next, the discrimination level is determined as the evaluation function of the $(K_p + 1)$th individual, where K_p is the pool size of parents to produce offspring. Finally, the probabilities are computed by the following relation:

$$p_k = \begin{cases} \dfrac{E(\mathbf{a}^{K_p+1}) - E(\mathbf{a}^k)}{\sum_{k=1}^{K_p}\left(E(\mathbf{a}^{K_p+1}) - E(\mathbf{a}^k)\right)} & (k \le K_p) \\ 0 & (k > K_p). \end{cases} \tag{18}$$

In this selection strategy, the bias $E(\mathbf{a}^{K_p+1})$ can be calculated automatically from the distribution of the evaluation functions in each generation by simply specifying K_p as a fixed size.

This strategy accelerates the convergence of population, but might be a risk to induce premature convergence to local optima because its influence is more powerful when the fitness variance is large in early generations. However, this is not a problem because the purpose of the GA with ADM is only to search for *the near-optimal individuals* that include local optima.

4) Elitism: The elitism strategy proposed by De Jong [23] prevents loss of a superior individual in convergence processes. It can be simply implemented by allowing the individual with the best fitness in the last generation to survive into the new generation without any modifications. The purpose of this strategy is same to the purpose of the selection strategy.

5) Migration on Island Model: The migration on island model [24]–[26] is a different kind of strategy compared with the above operators, and this strategy is considered optional. The ADM uses multiple GAs. When multiple GAs are running in a system, it allows exchanges of the best individuals in among different GAs. In our implementation, the opportunity for such exchanges happens only once, when GAs are invoked from ADM. In other words, the best individual from other GAs is contained in the initial population. Since the best individual, once found, will not be lost as long as the elitism operator is also available, this strategy promotes convergence [25], [26]. However, it may also induce the population to fall into a local optimum. The relation between the system with ADM and the GAs is similar to the relation between a GA and the population. If one considers the ADM as a conventional GA and considers the GAs as conventional population, this migration strategy acts as an elitism operator for the ADM. Thus, it may obstruct the diversity that is one of the main purposes of using ADM. An example of this case will be shown in the Section V.

Procedure of Single Genetic Algorithm
begin
initialize individuals $\mathbf{a}^k \quad (k = 1, \cdots, K)$
set the immigrant individual into the population, if the migration is enabled.
repeat (*generation :* $1, \cdots, L_{\mathrm{GA}}$)
compute evaluation functions of individuals, $E(\mathbf{a}^k)$
if ($\min[E(\mathbf{a}^k)] <$ *the predefined criteria*) **break;**
compute p_k *according to* $E(\mathbf{a}^k)$*, where* p_k *shows the probability of each individual to be a parent that generates offspring*
for each *offspring*
pick up parent(s) from parents pool according to p_k
generate offspring by the genetic operators (crossover, mutation, fluctuation, elitism)
push back the parent(s) picked up to the parents pool
settle the offspring to the offspring s pool
end for
update the population as a parents pool by copying from offspring s pool
end repeat
return *the best individual and its evaluation function*
end

Fig. 3. Procedure of the **Single GA**.

B. Procedure

The outline of the GA we adopted is shown in Fig. 3. The main purpose of this GA is to find a candidate solution with the smallest residuals in the particular parameter domain which is suggested by the ADM.

In the initialization, individuals are mapped at random into the parameter space. If the optional migration strategy is made available, one individual selected at random is replaced by the immigrant.

In the program loop with respect to each generation, there are four components: the evaluation of individuals, checking the convergence, the selection with associated calculation of the probability for reproduction, and the reproduction of the population for the next generation. This loop is repeated L_{GA} times that is predefined to control the GA. In the GA with ADM, this limit L_{GA} is set to a small number, i.e., generations are assigned a short lifetime.

The evaluation function for every individual is computed by (4) for the LS problem. Next, the condition to terminate is checked. If the evaluation function of the best individual in the population satisfies a predefined system criterion, then the GA returns the best individual as the final result, and the ADM also completes successfully. Otherwise, the GA goes to the next step to search better individuals. The probability to be selected for every individual p_k, is computed by (18). Individuals with a positive probability are settled into a pool of parents. Finally, offspring are reproduced by genetic operators. The number of the offspring produced by each operator is predefined; K_e for elitism the number is always 1, but it is K_c for crossover, K_m for random mutation, and K_f for fluctuation. The sum of the number of offspring is always equal to the population size of parents

$$K_e + K_c + K_m + K_f = K. \tag{19}$$

An offspring reproduced by the elitism operator is selected as the individual that has the largest probability among the pool of parents, and the offspring is settled to a pool of offspring. The parent is pushed back to the pool of parents. In both the mutation and the fluctuation, a parent is selected from among the pool of parents according to the probabilities, and the offspring is settled to the pool of offspring. In the crossover, two parents are selected, and two offspring are generated. In each operation after the parents are referenced to reproduce offspring they are returned to the pool of parents without any modifications. This means that particular parents with higher probabilities are selected multiple times to generate offspring. In a single generation, the operator to reproduce an offspring is applied only once; therefore, there are no offspring for which an operator is applied several times. In the last step of this program loop, the population for the next generation is updated by the pool of offspring.

V. Numerical Test

The purpose of this section is to demonstrate the effectiveness of the ADM. The effects in terms of specific parameters to control GAs will not be discussed here, and the systematic analysis of such effects is one topic for future investigation. However, we have obtained the following empirical knowledge from our other numerical experiments: the GA's parameters affect the computational cost, but their influence on the reliability of an estimation is small.

A. Input Data to the System of the LS Method

We begin with a problem represented by a model of an optical interference pattern with spatial distribution of light intensity that results from two coherent Gaussian beams. In an optical interference experiment, first, a light wave, which is plane wave with Gaussian shape distribution in the transverse plane, is divided into two waves. Next, these waves are guided along separate paths, and finally, they are superimposed at a certain angle with respect to each other, onto a two-dimensional (2-D) screen. A sensor device such as a camera may be used to detect the intensity distribution, which represents the square of the amplitude of a superimposed wave, including some background light. The test function to be estimated is determined by the intensity distribution measured by the sensor, which can be expressed as follows in a one-dimensional (1-D) problem:

$$\begin{aligned} y(x) &= A_1(x)^2 + A_2(x)^2 \\ &\quad + 2A_1(x)A_2(x)\cos(\Delta k x - \Delta\theta) + I_{\mathrm{bg}} \\ A_1(x) &= A_0 e^{-\frac{(x-x_0)^2}{w_0^2}}, \quad A_2(x) = \alpha A_1(x) \end{aligned} \tag{20}$$

where A_0 is the maximum intensity of one of beams, α denotes the intensity ratio between the two beams, x_0 and w_0 denote the common center and the common width of Gaussian beams, respectively, Δk and $\Delta\theta$ are, respectively, the differences of wave numbers along the x axis and of initial phase between two beams, and I_{bg} denotes the background intensity. The vector to be estimated is represented as a list of seven parameters $(A_0, \Delta k, \Delta\theta, \alpha, w_0, x_0, I_{\mathrm{bg}})$. It is possible to obtain the derivatives of the test function with respect to the parameters that appeared in *normal equation* by the general LS method; however, it requires much effort.

To demonstrate the proposed method, we prepared two data sets of simulated measurements and one experimental data set as follows:

Sim-NoError:
data without error from assumption
of a true parameter vector,
Sim-with-Error:
data with error with normal distribution
to the data from the true vector,
Exp : real experimental data.

The input data of "**Sim-NoError**" and "**Sim-with-Error**" are shown in Fig. 4, which also shows estimated results. Since the least mean square of a residual is estimated by (4), our system requires an estimation of the standard deviation for every data point, which is based on the measurement model or on several experiments. In the first two data sets, the standard deviation is given as a value that is 1/2 of the square root of each measurement, and in the last one, it is estimated from several experiments. In the case without error (**Sim-NoError**), the ideal value of each residual shrinks to zero when the vector of the system solution agrees with the assumed vector. In the cases with error (**Sim-with-Error**, **Exp**), the ideal least mean squares, i.e., the

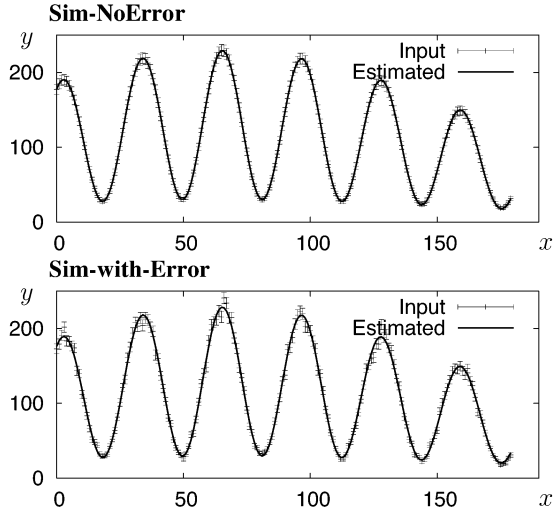


Fig. 4. Input data and examples of estimated results. Each input data point is plotted with an error bar that shows the standard deviation of the data itself. These data are applied as input data for the estimation system. Also, the plotted function estimated from the vector of seven parameters, which is the solution of the system, is depicted as a solid line. The estimations in this figure are computed by a **Single GA**.

TABLE I
CONDITIONS OF THE ADMS

Method	Number of GAs: N_q	Generations in each GA: L_{GA}	Times of ADM scheme	Migration
Single GA	1	3000	1	—
ADM				
4 GAs	4	30	25	disable
4 GAs Mig.	4	30	25	enable
2 GAs Long	2	30	100	disable
4 GAs Long	4	30	50	disable
8 GAs Long	8	30	25	disable

least mean square defined by (4) when every estimated parameter is equal to the assumed parameter, approaches to unity because the expected standard deviation between the estimated solution and measurement for each data point agrees with the standard deviation of the measurement error. Even if one were to use an ideal estimation system, it would still return a different estimation from the assumed parameters because the number of sample data points is finite. Therefore, the actual least mean square is not exactly one. In order to ensure that the actual least mean square agrees with one, the number of sample data should be sufficiently large compared with the number of parameters. The number of sample points of each input data is 180.

To demonstrate the applicability of the ADMs, we compute under different conditions, as shown in Table I. Each approach for the input, of which true parameters are known (**Sim-NoError** or **Sim-with-Error**), is evaluated with two types of initial domain, as shown in Table II. In an "**internal**" one, the initial domain of parameters contains the true solution; but an "**external**" one does not include the true solution in its domain. The *hard limits* shown in the Table II are set in two cases. The first is a consequence of physical restrictions due to the nature of the parameters. The purpose of the second case is to enable a reduction of the search space without lack of generality, which means

TABLE II
ASSUMED TRUE PARAMETERS AND THEIR INITIAL DOMAINS

parameter	true value	initial domain **Internal** lower, upper	initial domain **External** lower, upper
A_0	10	[0 , 20 ⟩	[0 , 5 ⟩
Δk	0.2	[0.10 , 0.30 ⟩	⟨ 0.25 , 0.30 ⟩
$\Delta\theta$	0.5	[−3.2 , 3.2]	[−3.2 , 3.2]
α	0.5	[0 , 1]	[0 , 1]
w_0	200	[0 , 400 ⟩	[0 , 30 ⟩
x_0	65	⟨ 0 , 600 ⟩	⟨ 300 , 600 ⟩
I_{bg}	5	[0 , 10]	[0 , 3 ⟩

The marks [and] show hard limits that act a role of limits during the ADM. Also the marks ⟨ and ⟩ show the initial limits that is only available in the first loop of the ADM.

unique representation of the test function by the parameters. The intensity A_0, and the intensity ratio α, should be positive because the intensities, A_1, A_2, are positive. Moreover, α can be restricted to less than or equal to one because of unique representation. The initial phase $\Delta\theta$ can be also restricted between $-\pi$ and $+\pi$. Note that the "**external**" domain is narrower than the "**internal**" one, and the former is included within the latter.

In addition, the control parameters of the ADM, the *width factor* in (11) and the *relaxation factor* in (12) are the same for all results; $\gamma = 3.0$, $\beta = 0.5$. Also, the parameters to control the GA are as follows: the factor that chooses the crossover type p_c in (15), is 0.8; the range of the fluctuation R in (16) is 0.3; the numbers of offspring produced by each genetic strategy are the elitism $K_e = 1$, the crossover $K_c = 20$, the random mutation $K_m = 2$, and the fluctuation $K_f = 8$; the population size is equal to these sum $K = 31$, and the pool size of parents $K_p = 26$. The details will not be shown here, but we have found from other results that convergence is achieved more quickly in cases in which the ratio of the crossover and the fluctuation is higher.

B. Simulation in a Case for Which the Assumed Parameters are Within the Predefined Domain (Internal Type)

1) Result for the Input "Sim-NoError": Regarding the first attempt, an example of the result for the case "**Sim-NoError**" by the simple real-coded GA is shown in the top of Fig. 4. It appears to be a good estimation; however, we cannot guarantee that other estimations with different random seeds will always return similar results.

To evaluate the reliability of the result, we ran the estimator 100 times with different random seeds. The frequency histogram of these trials by **Single GA**, and also the convergence profile of the least mean square are shown in Fig. 5. The convergence represents a saturation to almost 0.1 around 500th generation. Also, the frequency histogram shows most of the solutions are distributed around 0.1. However, in the case of nonerror model, the ideal LS should be equal to 0. There are two possible reasons for gap of these results; the populations are concentrated into a local optimum and the GA cannot escape from that area, or the GA is not working effectively because the parameter domains are too wide.

Next, we will show results from the ADM simulations. The number of generations in each GA with ADM, which has been shown in Table II, is determined in reference to the result of the

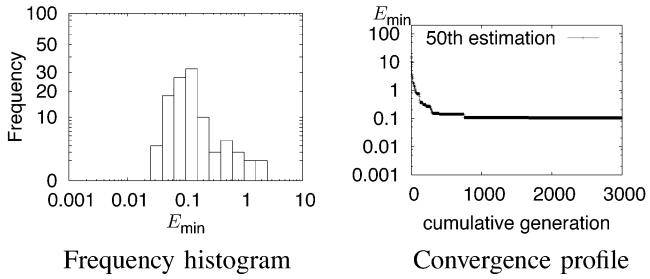


Fig. 5. Frequency histogram and typical convergence profile for "**Sim-No-Error**" by a "**Single GA**." The frequency histogram shows the final least mean square for 100 trials. The frequency in longitudinal axis is shown with nonlinear scale. The convergence profile is the result of the estimation for which the rank of the final least mean square is 50th in 100 trials. The longitudinal in the convergence profile shows the least mean square, whose final result is corresponding to the quadrature axis of the frequency histogram.

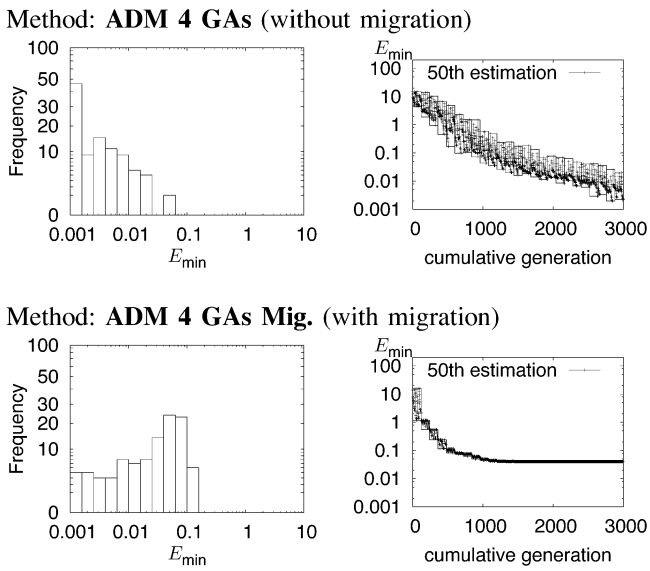


Fig. 6. Difference between cases with and without the migration strategies. Frequency histograms of final least mean square for 100 trials in the left-hand side, and typical examples of convergence profile are shown in the right-hand side. These data are computed by the ADM under "**internal type condition**" for "**Sim-NoError**." Frequencies for which the final least mean square is smaller than the lowest level are counted into the lowest section. The rank of the convergence profiles with respect to the final least mean square is 50th in 100 trials. Each box shows the group of an adaptive range approach that contains four GAs. The label in the abscissa, i.e., "cumulative generation," means that each generation of a single GA with short lifetime (Max. 30 generations) for the ADM is accumulated.

convergence profile by "**Single GA**." The role of the GA is to find *the near-optimal individual*; therefore, the number of generations is considered sufficient when the early convergence is almost complete. Fig. 6 shows a comparison of the effects of application of the migration strategy. In this figure, the comparison between "**ADM 4 GAs**" and "**ADM 4 GAs Mig.**" demonstrates that the migration strategy is effective only in early stage, and on the contrary it makes convergence worse at later stage. This result seems unexpected, since the migration strategy is generally considered effective to improve the convergence. The reason for this anomaly is determined to be premature convergence of the domain to a local optimum, which is due to the migration. The GAs that employ the migration strategy may estimate similar

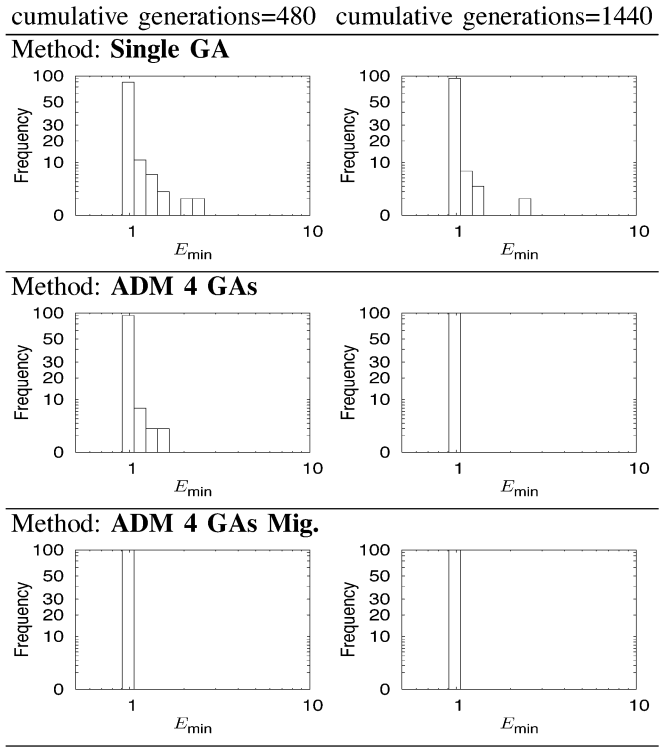


Fig. 7. Frequency histograms for "**Sim-with-Error**" under "**internal type condition**," when the cumulative generations are 480 and 1440.

solutions because the best solution at the last calling of GAs is included in initial population. Accordingly, the dispersion of the results from GAs becomes small, and the ADM will lock onto a very narrow area. Once the parameter domain falls into a very narrow and incorrect domain that can include only one local optimum, the ADM does not work effectively, because a single GA has a low probability of escaping from this incorrect domain. Thus, the migration strategy reduces the robustness of the ADM.

2) Result for the Input "Sim-With-Error": From the results of "**Sim-NoError**" in Fig. 5, we can find that there are several local optima for which the amplitude of the evaluation function is less than 0.1. In contrast, the ideal least mean square of "**Sim-with-Error**" is almost one. When the evaluation function of "**Sim-with-Error**" is considered as sum of the evaluation function for "**Sim-NoError**"and this ideal least mean square, then, in the case of "**Sim-with-Error**," the evaluation function of local optima in the vicinity of the global optimum is distributed around 1 with variance 0.1. From the viewpoint of the selection pressure in (18), the diversity is not reduced in this case. Thus, we hypothesize that the problem of "**Sim-with-Error**" is easier than that of "**Sim-NoError**." Actually, no differences were found in the final results regarding the cases, regardless of whether the migration strategy is enabled or not. Accordingly, we show the results before convergence. The frequency histograms at the 480th and 1440th cumulative generations, instead of the final results, are shown in Fig. 7. At the 1440th generation all results, except the one by "**Single GA**," converge around $E_{\min} = 1$. At the 480th generation the performance by "**ADM 4 GAs Mig**." is better than that by "**ADM 4**

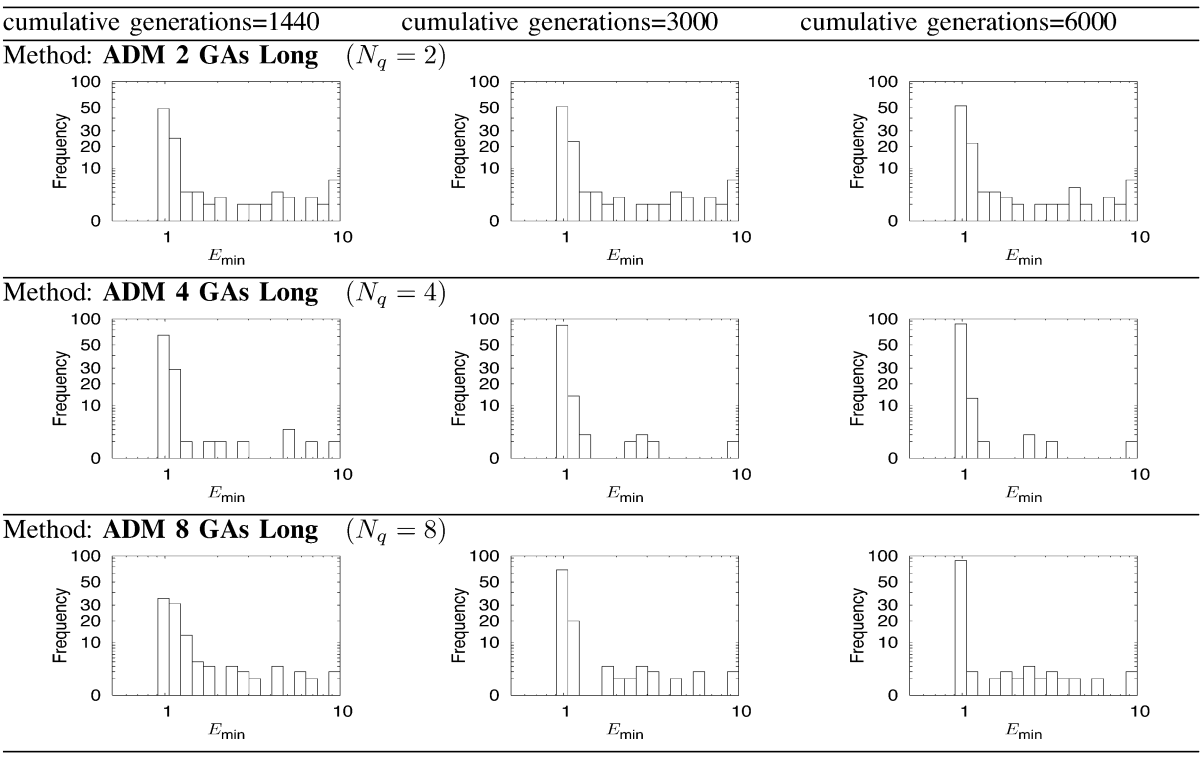


Fig. 8. Differences of frequency histograms with respect to number of GAs and cumulative generations. Every data is computed under "**external type condition**" for "**Sim-with-Error**." When the final least mean square exceeds highest level in the quadrature axis, the event is counted into the highest section.

GAs," which is different from the case for "**Sim-NoError**" in Fig. 6. The reason is that the population converges to the global optimum during the early stages in which the migration is effective, because the case of "**Sim-with-Error**" is an easier problem than that of "**Sim-NoError**." When the problems are easy cases where it is expected that the estimation will converge to the correct domain at an early stage, the migration is effective even for the smaller computational expense without lacking in reliability. Otherwise, the migration strategy should not be applied.

C. Simulation in a Case for Which the Assumed Parameters are Located Outside the Predefined Domain (External Type)

Next, we demonstrate the performance of an extrapolative search, in which some true values of the parameters to be estimated are not within the domain defined initially. Although we will not discuss the details in this paper because of lack of space, this situation is typically classified as a difficult estimation, and early convergence to the global optimum cannot be expected. Therefore, we may conclude it is not a good idea to apply the migration strategy, and we show only the results without the use of migration in this subsection.

The disadvantage of ADM appears in the case where only a local optimum is existing in the predefined or redefined domain. Even in this case, we can raise the reliability of solutions by increasing the number of GAs in ADM. However, this induces an increase of the computational expense. Next, we discuss the issue of deciding the number of GAs to execute in ADM.

Fig. 8 shows the frequency histograms in the cases where the number of GAs N_q, is 2, 4, and 8. In every case, some trials can successfully find the global optimum, while other trials fail. In the case of $N_q = 2$, the histograms for which cumulative generations are 1440, 3000, and 6000 are almost the same. Therefore, it is understood that they have already converged, and further convergence with increasing generations cannot be expected. Also, for $N_q = 4$, the histograms converge when the cumulative generations reach 3000. For $N_q = 8$, more delayed convergence can also be expected. Consequently, the convergence is achieved faster with a smaller number of GAs. If we assume that the acceptable solutions belong to the lowest concentrated section ($E_{\min} \lesssim 1.06$), then the probabilities to obtain acceptable solutions are shown in Table III. The maximum of cumulative generations $G_{\max}$ in this table means summation of number of generations in each GA; therefore, the computational cost is proportional to $G_{\max}$. In addition, since each GA with lifetime L_{GA} is invoked multiple times from the ADM, number of times of changing the domain by ADM is equal to $G_{\max}/(N_q L_{\mathrm{GA}})$; namely, it is proportional to $G_{\max}/N_q$. The comparison among the converged cases (underlined cases) shows that the reliability increases with the number of GAs, N_q. This is attributed to the

TABLE III
THE PROBABILITIES TO OBTAIN THE ACCEPTABLE SOLUTIONS

Number of GAs (N_q)	Max. cumulative generations : $G_{\max}$		
	1440	3000	6000
2	<u>0.48</u> (720)	<u>0.51</u> (1500)	<u>0.52</u> (3000)
4	0.51 (360)	<u>0.79</u> (750)	<u>0.82</u> (1500)
8	0.39 (180)	0.65 (375)	<u>0.83</u> (750)

The acceptable condition is $E_{\min} \lesssim 1.06$. The underline shows the results have converged. The parenthesis shows the value $G_{\max}/N_q$, Input measurement data is **Sim-With-Error** , and migration strategy is disabled.

TABLE IV
JOINT PROBABILITY TO OBTAIN THE ACCEPTABLE SOLUTION BY n-TIMES OF TRIALS

Number of GAs N_q	Max. cumulative generations : $G_{\max}$ Number of trials : n		
	$G_{\max}$=1440 $n=4$	$G_{\max}$=3000 $n=2$	$G_{\max}$=6000 $n=1$
2	0.92	0.76	0.52
4	0.94	0.95	0.82
8	0.86	0.87	0.83

The acceptable condition is $E_{\min} \lesssim 1.06$. Input measurement data is **Sim-With-Error** , and migration strategy is disabled.

fact that a case of smaller N_q has a larger probability that the ADM falls into the incorrect domain. Tuning of factors to control the ADM (e.g., *width factor* γ_m or *relaxation factor* β_m) does afford better results without doubt, but this is not considered a practical solution for general users who may be unfamiliar with this method.

The above results indicate that it is necessary to increase N_q and also $G_{\max}$ in order to obtain a more reliable solution; however, it requires more computational time. The design of mechanism by which the ADM invokes several GAs is based on the idea that reliability of solutions may be improved by utilizing multiple computations with different seeds. Similarly, several trials of running the estimation system should contribute to improving the reliability of the solution. Furthermore, such a system design may also reduce the computational expense. When the probability to obtain an acceptable solution in a trial is denoted as p_1, then the joint probability in n number of trials, p_n, is given by the following relation:

$$p_n = 1 - (1 - p_1)^n. \tag{21}$$

To compare under the same computational cost requirements, we show the probabilities for $n = 4$, 2, and 1, where $G_{\max} = 6000$ is corresponding to $n = 1$ in Table IV. All of the computational costs in Table IV are almost the same because the computational cost is proportional to the product of the cumulative generations $G_{\max}$, and the number of trials n. Some results in the cases $N_q = 4$ or 2 demonstrate better reliability than that of $N_q = 8$. Thus, in order to raise total reliability with the same computational cost, it is better to attempt several trials without good convergence than to seek a good convergence with one trial.

In the above discussion, we considered cases where only one CPU is applied for the estimations; however, in parallel computing with several CPUs the circumstances are different. If the number of GAs equals to the number of CPUs, the computational time is proportional to $G_{\max}/N_q$ instead of $G_{\max}$. The values of $G_{\max}/N_q$ are also shown in Table III within the parenthesis. Comparing the probability in terms of the similar $G_{\max}/N_q$, we can predict that increasing the number of GAs, N_q will yield better performance.

Finally, typical convergence profiles of parameters are shown in Fig. 9. The domains of parameters are modified dynamically with advance of the cumulative generations. The results demonstrate that the ADM is effective, even if the true solutions are out of the initial ranges.

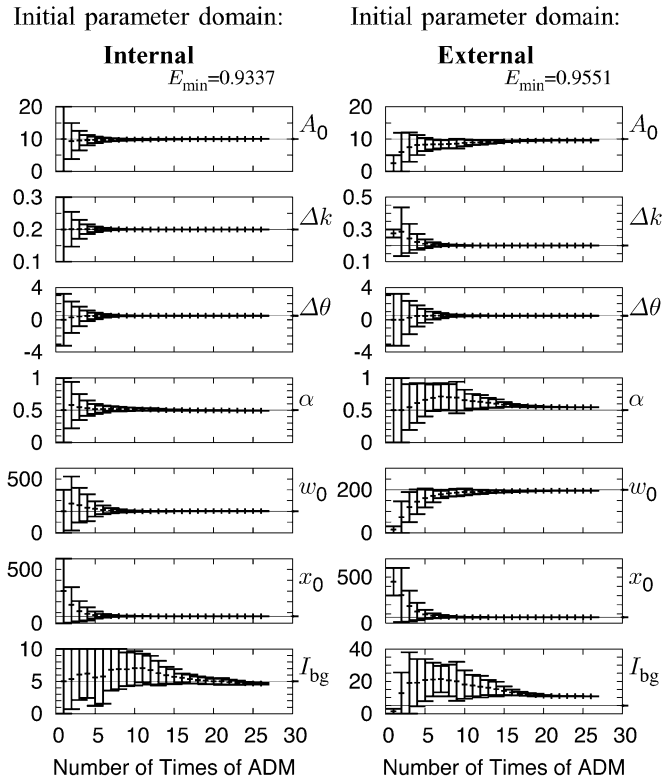


Fig. 9. Convergence of parameter domains for "**internal**" and "**external**" data. Input measurement data is "**Sim-With-Error**," and ADM is "**ADM 4 GAs**." The results are taken from the 50th rank in 100 trials. The bars at each iteration indicate the domain of parameters. Iteration means the number of times to apply the adaptive domain algorithm. The horizontal solid line in each figure shows the parameter assumed initially.

D. Fitting of Real Experimental Data

In the last part of this section, we demonstrate an estimation for a set of real experimental data. Fig. 10 shows input data acquired by an experiment and its estimated result. The experimental data was obtained from a laser interferometer as a 2-D image. The experimental devices were adjusted so that the fringes in the image were parallel to the vertical axis. The 1-D input data for the estimator is taken from the bounded area depicted at the top-left of Fig. 10. Both the average value and the standard deviation along the vertical axis are computed with respect to every horizontal point.

Table V shows the initial domain for the estimation and its estimated result. This result demonstrates that the technique is successful, even for the case of external domain searching with respect to Δk. The parameter $I_{\rm bg}$ has large variance. We hypothesize that this shows that the sensitivity of $I_{\rm bg}$ is small with respect to the LS (the evaluation function), i.e., $I_{\rm bg}$ is not important for LS estimation. The convergence profile in Fig. 9 supports our hypothesis; the convergence of $I_{\rm bg}$ is achieved after the other parameters converge. In other words, one can determine which parameters in an LS estimation are most sensitive to the least mean square by analyzing the convergence of parameters.

In Fig. 10, the frequency histogram shows that $E_{\min}$ is almost one. However, systematic errors are found within the middle of the figure. These are not due to any inconsistencies of the estimation method. As plausible explanations, the experimental

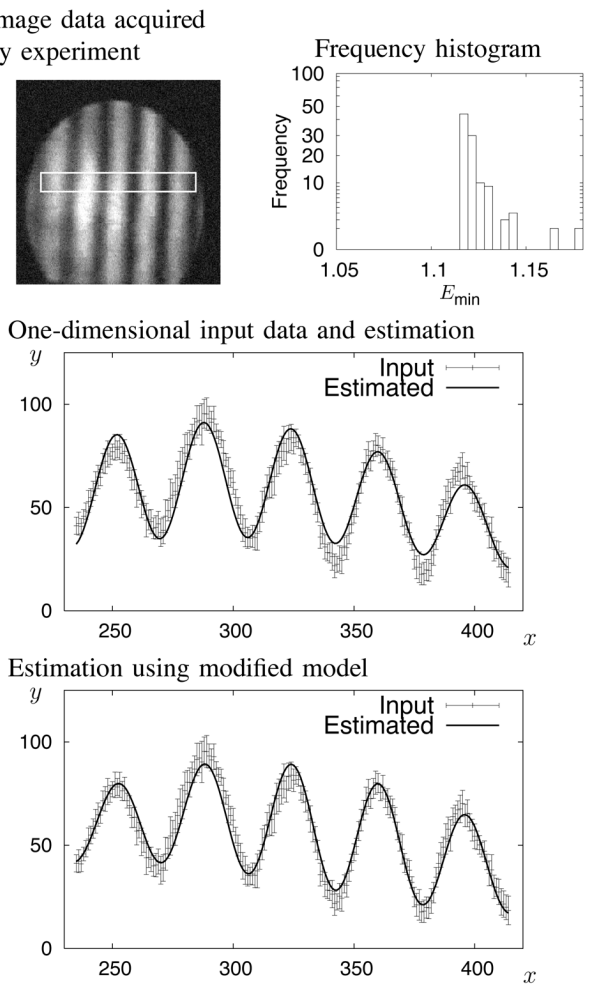


Fig. 10. Input data of "**Exp**" and its estimated result. The image data in left-top figure is obtained by an interferometer system with two light beams. The bounded area is used for the proposed estimation method. The input data for the proposed estimation method is prepared by computing the average and the standard deviation along the vertical axis for the every horizontal pixel. Both the input data and its estimated function obtained by the adaptive domain method of "**ADM 4 GAs**" are shown in the middle. The estimated function is of 50th rank in 100 trials. Each input data is plotted with an error bar that shows the standard deviation of the data itself. The top-right figure shows a frequency histogram of the least mean squares in 100 trials. The bottom figure shows the estimated result for modified model which assumes that the center and the width of two beams are different.

TABLE V
INITIAL PARAMETER DOMAINS AND ESTIMATED RESULT FOR **Exp** DATA

parameter	initial domain lower, upper	estimated result in best 50 trials	estimated result in all (100) trials
A_0	[0 , 20 ⟩	7.74± 0.02	7.60± 0.26
Δk	⟨ 0.2 , 0.3 ⟩	0.1730±0.0000	0.1730± 0.0001
$\Delta\theta$	[−3.2 , 3.2]	−0.486± 0.015	−0.486± 0.019
α	[0 , 1]	0.231± 0.002	0.242± 0.023
w_0	[1 , 300 ⟩	229.8± 1.6	225.1± 8.0
x_0	[240 , 420]	293.4± 0.3	293.7± 0.6
$I_{\rm bg}$	[0 , 10 ⟩	0.26± 0.33	2.22± 3.38

The marks [and] indicate hard boundaries that act as limits during the adaptive domain method. Also the marks ⟨ and ⟩ show the initial limits that are only applicable in the first loop of the adaptive domain method.

data may have had systematic errors (e.g., the image is slanted), and also there may be effects from some minor parameters that are not included in the estimation model in (20). To increase the accuracy of the model, we assume that the center and the width of the two beams' envelopes are different. The modified model is as follows:

$$\begin{aligned} y(x) &= A_1(x)^2 + A_2(x)^2 \\ &\quad + 2A_1(x)A_2(x)\cos(\Delta k x - \Delta\theta) + I_{\rm bg} \\ A_1(x) &= A_0 e^{-\frac{(x-x_1)^2}{w_1^2}}, \quad A_2(x) = \alpha A_0 e^{-\frac{(x-x_2)^2}{w_2^2}}. \end{aligned} \tag{22}$$

The result of this modification is also shown at the bottom of Fig. 10. Some of the systematic error is reduced. The improved model can be realized by only a few lines modification to the program source code.

Finally, we discuss the topic of optimizing computation time. In our numerical test for which the number of parameters to be estimated is seven , the computational time is almost 6 seconds for each estimation with $G_{\max} = 3000$, using PC with an Intel Pentium-4 processor operating 2 GHz. Thus, even with retrying ten times with different random system to obtain a more reliable solution, one can obtain the solution in only 1 minute.

VI. CONCLUSION

In conventional LS regressions for nonlinear problems, it is difficult to obtain the *normal equation* analytically. As an alternative, a GA may be used as a search method to determine the optimum value in the parameter space. It is expected that by the application of this technique to problems of LS regression, we can find solutions without relying on the *normal equation*.

In a simple real-coded GA, the domain of parameters with respect to the phenotype of a gene is fixed in the system. If the parameter domain is too wide the estimation is not effective. Conversely, it is difficult to choose a narrower parameter domain that is certain to contain an acceptable solution.

In this paper, we proposed a new method, ADM, to change the domain dynamically using several GAs with short lifetimes. The merits of ADM include the following points: the estimation is almost always reliable, and the computational cost for estimations is relatively small. Although general GAs are typically difficult to manage from the aspects of reliability and computational cost, ADM requires no manual tuning or selection of parameters to optimize these aspects. Hence, ADM is especially useful for general researchers who may not have expert knowledge of GA simulation models.

We demonstrated a nonlinear LS fitting problem concerning the interference pattern from two light beams. In the simulation of this problem, an acceptable estimation was always obtained when the assumed solution was within the predefined domain. Furthermore, even if the predefined domain did not contain the assumed solution, the dynamic domain redefinition proved to work effectively as an extrapolation method. Only in the case where the predefined domain contains only a local optimum, did some of the estimations fail. However, such cases can be prevented by defining a wider initial domain. This may be accomplished without significantly sacrificing the convergence performance because the dynamic redefinition of ADM is effective. In the statistical fitting for a set of real experimental data, we can

obtain reasonable results without any tuning of the factors that are required to control GAs and ADM.

The computational cost of ADM will likely be more expensive than that of solving the normal equation. However, enhancements for more detailed models of the function to be estimated can be easily applied with only slight modifications to the source program.

Consequently, we conclude that the proposed ADM is effective and reliable for nonlinear LS regressions, and also advantageous since it is easy to use without any detailed knowledge of GAs or their tuning. Therefore, the ADM method may be useful for general scientists in many fields.

Moreover, in the development of the ADM, we impose no restriction on potential users except that the applied problem should be a minimization problem. Thus, the ADM is applicable not only to LS regressions but also to many other kinds of optimization problems in which the parameters to be optimized are real numbers.

REFERENCES

[1] W. H. Press, S. A. Teukolsky, W. T. Vetterling, and B. P. Flannery, *Numerical Recipes in Fortran 77: The Art of Scientific Computing*, 2nd ed. Cambridge, U.K.: Cambridge Univ. Press, 1992.

[2] M. Maeder, Y.-M. Neuhold, and G. Puxty, "Application of a genetic algorithm: Near optimal estimation of the rate and equilibrium constants of complex reaction mechanisms," *Chemometrics Intell. Lab. Syst.*, vol. 70, pp. 193–203, 2004.

[3] I. Rechenberg, "Cybernetic solution path of an experimental problem," in *Royal Aircraft Establishment,* Transl.: 1122. Piscataway, NJ: IEEE Press, 1965, Reprint in: D. B. Fogel (Ed.), "Evolutionary Computation," The Fossil Record, pp. 301–309, 1995.

[4] J. H. Holland*, Adaption in Natural and Artificial Systems*. Ann Arbor, MI: Univ. Michigan, 1975; Cambridge, MA: MIT Press, 1992.

[5] D. E. Goldberg*, Genetic Algorithm in Search, Optimization and Learning*. Reading, MA: Addison-Wesley, 1989.

[6] Z. Michalewicz*, Genetic Algorithms + Data Structures = Evolution Programs*. Berlin, Germany: Springer-Verlag, 1996.

[7] F. Li, R. Morgan, and D. Williams, "Hybrid genetic approaches to ramping rate constrained dynamic economic dispatch," *Electric Power Syst. Res.*, vol. 43, pp. 97–103, 1997.

[8] S. Baskar, P. Subbaraj, and M. V. C. Rao, "Hybrid real coded genetic algorithm solution to economic dispatch problem," *Comput. Elec. Eng.*, vol. 29, pp. 407–419, 2003.

[9] J.-H. Kim and H. Myung, "Evolutionary programming techniques for constrained optimization problems," *IEEE Trans. Evol. Comput.*, vol. 1, pp. 129–140, 1997.

[10] M. Arakawa and I. Hagiwara, "Development of adaptive real range (ARRange) genetic algorithms," *JSME Int. J.*, ser. C, vol. 41, no. (4), pp. 969–977, 1997.

[11] A. Oyama, S. Obayashi, and T. Nakamura, "Real-coded adaptive range genetic algorithm applied to transonic wing optimization," *Appl. Soft Comput.*, vol. 1, pp. 179–187, 2001.

[12] D. H. Wolpert and W. G. Macready, "No free lunch theorems for optimization," *IEEE Trans. Evol. Comput.*, vol. 1, pp. 67–82, 1997.

[13] A. H. Wright, "Genetic algorithms for real parameter optimization," in *Foundations of Genetic Algorithms*. San Mateo, CA: Morgan Kaufman, 1991, pp. 205–218.

[14] C. L. Karr, B. Weck, D. L. Massart, and P. Vankeerberghen, "Least median squares curve fitting using a genetic algorithm," *Eng. Appl. Artif. Intell.*, vol. 8, pp. 177–188, 1995.

[15] P. Vankeerberghen, J. Smeyers-Verbeke, R. Leardi, C. L. Karr, and D. L. Massart, "Robust regression and outlier detection for non-linear models using genetic algorithms," *Chemometrics Intell. Lab. Syst.*, vol. 28, pp. 73–87, 1995.

[16] N. Yi, T. Dechun, and L. Yuzheng, "Genetic algorithm diagnosis of individual cell frequencies in a coupled cavity chain," *Nuclear Instrum. Method in Phys. Res.*, vol. A 462, pp. 356–363, 2001.

[17] Y.-D. Kwon, S.-B. Kwon, S.-B. Jin, and J.-Y. Kim, "Convergence enhanced genetic algorithm with successive zooming method for solving continuous optimization problems," *Comput. Structures*, vol. 81, pp. 1715–1725, 2003.

[18] A. B. Djurisčić, J. M. Elazar, and A. D. Rakić, "Genetic algorithms for continuous optimization problems—A concept of parameter-space size adjustment," *J. Physics A: Math. Gen.*, vol. 30, pp. 7849–7861, 1997.

[19] F. Herrera, M. Lozan, and J.L. Verdegay, "Tackling real-coded genetic algorithms: Operators and tools for behavioral analysis," *Artif. Intell. Rev.*, vol. 12, pp. 265–319, 1998.

[20] A. Blanco, M. Delgado, and M.C. Pegalajar, "A real-coded genetic algorithm for training recurrent neural networks," *Neural Netw.*, vol. 14, pp. 93–105, 2001.

[21] H. Mühelenbein and D. Schlierkamp-Voosen, "Predictive models for the breeder genetic algorithm I. Continuous parameter optimization," *Evol. Comput.*, vol. 1, pp. 25–49, 1993.

[22] S. Tomioka, S. Nisiyama, and T. Enoto, "Identification of electromagnetic mode excited by electron beam in waveguide using genetic algorithm," in *Comput. Eng. I; Advances in Continuum Mechanics and Electromagnetics*. Japan: Japan Society for Computational Method in Engineering, 2004, pp. 251–260.

[23] K. A. De Jong, "An analysis of the behavior of a class of genetic adaptive systems," Ph.D. dissertation, Univ. Michigan, Ann Arbor, MI, 1975, Univ. Microfilms No. 76-9381.

[24] *Genetic Algorithms and Neural Networks—Scheduling and Combinatorial Optimization* (in Japanese) , The Institute of Electrical Engineers of Japan, Ed. Tokyo, Japan: Corona Publishing, 1998.

[25] B. Kröger, P. Schwenderling, and O. Vornberger, "Parallel genetic packing of rectangles," in *Parallel Problem Solving from Nature*. Berlin, Germany: Springer-Verlag, 1991, pp. 160–164.

[26] T. Starkweather, D. Whitley, and K. Mathias, "Optimization using distributed genetic algorithms," in *Parallel Problem Solving from Nature*. Berlin, Germany: Springer-Verlag, 1991, pp. 176–185.

Satoshi Tomioka was born in Nara, Japan, in 1962. He received the B.Eng. degree in nuclear engineering from the Department of Engineering, Hokkaido University, Sapporo, Japan, in 1986, and the M.Eng. and Dr. Eng. degrees in nuclear engineering from the Graduate School of Engineering, Hokkaido University, in 1988 and 1996, respectively.

He worked at Fujitsu LTD., Japan, before joining Hokkaido University in 1991. At Fujitsu LTD., he developed on-board computers on solar observation satellite and X-ray observation satellite. Currently, he is an Associate Professor in the Division of Quantum Science and Engineering, Graduate School of Engineering, Hokkaido University. At Hokkaido University, his current research interests include plasma diagnostics, boundary element method for electromagnetic field, and infrared holography.

Shusuke Nisiyama was born in Yokohama, Japan, in 1974. He received the B.Eng. degree in nuclear engineering from the Department of Engineering, Hokkaido University, Sapporo, Japan, in 1994, and the M.Eng. degree in nuclear engineering and the Dr. Eng. degree in quantum energy engineering from the Graduate School of Engineering, Hokkaido University, in 1996 and 2000, respectively.

From 2000 to 2001, he was a Research Fellow at the Center for Advanced Research of Energy Technology, Hokkaido University. He is currently a Research Associate at the Division of Quantum Science and Engineering, Graduate School of Engineering, Hokkaido University. His research interests include numerical analysis of electromagnetic field and measurement of accelerated electron beam.

Takeaki Enoto (S'67–M'71) was born in Sapporo, Japan, in 1944. He received the B.Eng. degree in electronics engineering from the Department of Engineering, Hokkaido University, Sapporo, Japan, in 1966, and the M.Eng. and Dr. Eng. degrees in electronics engineering from the Graduate School of Engineering, Hokkaido University, in 1968 and 1971, respectively.

From 1971 to 1975, he was a Research Associate at the Faculty of Engineering, Hokkaido University. From 1975 to 1990, he was an Associate Professor at the Faculty of Engineering, Hokkaido University. He is currently a Professor at the Division of Quantum Science and Engineering, Graduate School of Engineering, Hokkaido University, and Director at the Center for Advanced Research of Energy Conversion Material, Hokkaido University. His research interests include plasma diagnostics, ultra-high-speed camera system, and measurement of accelerated electron beam.