# A Derivative-Free Trust-Region Algorithm for Unconstrained Optimization with Controllable Error

Guidance

Associate Professor Nobuo YAMASHITA

Jun TAKAKI

2006 Graduate Course

in

Department of Applied Mathematics and Physics

Graduate School of Informatics

Kyoto University

February 2008

## **Abstract**

In this paper, we consider the unconstrained optimization problem under the following situation: (S1) The objective function is evaluated with a certain amount of error, (S2) the error is controllable, that is, the objective function can be evaluated in any desirable accuracy, and (S3) the evaluation with higher accuracy requires more computation time. This situation arises in many fields such as engineering and financial problems, where the objective function value results from a huge amount of numerical calculation or simulation.

Under (S1)—(S3), algorithms using derivatives of the objective function are not applicable. To date, there exist several algorithms which use the objective function values only. Conn *et al.* proposed the Derivative-Free Optimization method (DFO) as such an algorithm. The DFO generates a sequence of minimizers of quadratic model functions derived from sample points and their objective function values. Moreover, it adopts a trust-region framework for global convergence. Conn *et al.* shows its global convergence when the objective values at sample points are evaluated accurately. To apply DFO for the problem under (S1) and (S2), the objective function should be calculated with high accuracy in the whole step of DFO. However, computations with high accuracy require a large amount of computation time under (S3). Therefore the DFO algorithm cannot be applied directly to such a problem that requires a large number of function evaluations to find a solution.

Under (S2) and (S3), it would be reasonable to set the accuracy of the evaluation to be low at a point far from a solution, and to set it to be high at a point in the neighborhood of a solution. In this paper, we propose a derivative-free trust-region algorithm based on this idea. For this purpose, we consider (i) how to construct a quadratic model function by exploiting pointwise errors and (ii) how to control the accuracy of function evaluations to reduce the total computation time of the algorithm. For (i), we propose a method based on support vector regression. For (ii), we give some updating formula of the accuracy that is related to the trust-region radius. We present numerical experiments with the proposed algorithm for several test problems taken from CUTEr as well as a financial problem which estimates implied volatilities from some option prices. The results show that the proposed algorithm not only works much faster than the DFO with high-accuracy function evaluations at all time, but also finds some accurate solutions. In particular, for relatively easy small-scale problems, the proposed algorithm can obtain optimal solutions with computational costs which amount to those required for several times evaluations of an accurate function value.

# Contents

## 1 Introduction

In this paper, we consider the unconstrained minimization problem

$$\underset{\boldsymbol{x}\in\mathbb{R}^n}{\text{minimize}}\, f(\boldsymbol{x}), \tag{1.1}$$

where $f$ is a nonlinear function from $\mathbb{R}^n$ into $\mathbb{R}$. When the function values and the derivatives of $f$ can be exactly computed, we can solve the problem (1.1) by Newton's method, quasi-Newton method and others. However, there exist many practical problems where objective function values cannot be accurately computed. In this paper, we suppose the following situation, in which the function value $f(\boldsymbol{x})$ can be evaluated for all $\boldsymbol{x} \in \mathbb{R}^n$ with an arbitrary accuracy.

**(S1)** Evaluated objective function value $\hat{f}(\boldsymbol{x})$ contains a certain amount of error $e$,

$$\hat{f}(\boldsymbol{x}) = f(\boldsymbol{x}) + e,$$

**(S2)** the error $e$ is controllable, that is, we can compute $\hat{f}$ such that $|e| \leq \epsilon$ for any accuracy $\epsilon$, and

**(S3)** to obtain the more highly accurate $\hat{f}$, the more computation time is required.

This situation (S1)—(S3) often arises in engineering [2, 16], bilevel programming [3] and financical problems [23]. Typically, such problems arises when one needs to minimize a function measured by some numerical calculate, or some experiment or by a complicated simulation. Therefore, the evaluated objective function value contains some error (S1). On the other hand, the accuracy of the evaluated objective value in such a problem is set arbitrarily (S2). Moreover, to evaluate the function with the higher accuracy, the more computation time is required in general (S3). For instance, consider the case where the objective function contains the integration and it is calculated by a Monte Carlo method. It is known that the computation time $O(1/\epsilon^2)$ is required to guarantee that the error of the integration is less than $\epsilon$ [23].

Under (S1), not only the derivative of $f$ can not be computed, but also the finite difference approximation based on the function evaluations is unreliable. Therefore, the optimization algorithms using derivative of the objective function cannot be applied for such a problem. To date, several computational methods using the objective function value only have been proposed. Especially, direct search methods, Meta heuristics and model-based methods are studied well.

The direct search methods are classical and one of the most popular minimization technique in practice. The simplex reflection algorithm of Nelder and Mead [18], or its modern variants such as the Parallel Direct Search algorithm of Dennis and Torczon [14, 24] are its typical methods. These methods are easy to be implemented so that they are used by many practitioners. However, they do not exploit the information of the objective function well, and thus, its computation time becomes relatively larger than that of model-based methods. A good introduction to these techniques can be found in the book [17].

The Meta heuristics are the methods which have high possibility of finding a global minimum. The Genetic algorithm, the Particle Swarm Optimization algorithm and the Ant Colony Optimization algorithm are well studied. These techniques suffer, in general, from the following two drawbacks. Termination criteria is unclear and the larger number of function evaluations for finding a local optimal solution than that of direct search methods or model-based methods.

The model-based methods successively construct a surrogate model function from sample points and its objective function values, and minimize the model function to find a local solution of the problem. This idea of model-based methods is used in Design of Experiments [15] for a long time. Winfield [25, 26] proposed a model-based algorithm built in the idea of trust-region methods for global convergence. We will call *derivative-free trust-region* algorithms the model-based algorithms of this type. In [25], the quadratic model function was constructed by solving the system of linear equations of composed of $N := \frac{1}{2}(n+1)(n+2)$ sample points. However, he did not give the concrete method for generating sample points. Then, Powell [22] proposed a generation method of sample points and the construction of the model were based on Lagrange interpolation function for improving the quality of the model function as an approximation of the objective function. However, constructing Lagrange interpolation function required a large amount of computation time so that his algorithm can not be applied to relatively large problems. For the purpose of reducing computation time, Conn *et al.* [5] proposed to use Newton fundamental polynomial function for constructing the model. Furthermore, Conn *et al.* [8] established global convergence for the derivative-free trust-region algorithms under some assumptions for the first time. In [7], he discussed on an algorithm, in which the quadratic model function was constructed by Support Vector Regression (SVR), for a case where the objective function values are evaluated with uniformly random errors.

On the studies described above, the function is evaluated exactly, or even if an error in the evaluation exists, it is assumed to be very small. Thus, to apply these algorithms under (S1) and (S2), the function must be evaluated with high accuracy. However, since the function evaluation with high accuracy needs a large amount of computation time under (S3) so that these algorithms cannot be applied to problems which require the large number of function evaluations for finding a solution. Under (S3) where the function evaluation time and its accuracy are in the relation of trade-off, it would be reasonble to set the accuracy to be low at point far from a solution, and to set it to be high at a point in the neighborhood of a solution. In this paper, we propose a derivative-free trust-region algorithm based on this idea, which is a kind of the model-based algorithm proposed by Winfield.

For this purpose, we have to consider how to construct a quadratic model function exploiting pointwise errors and how to control the accuracies of the function evaluations for reducing total computation time of the algorithm. For the first, we develop a constructing method of a quadratic model function which is suitable under (S1)—(S3). The method is based on SVR. SVR is a technique for obtaining the regression curve that enters width $\hat{\epsilon}$ or less from the sample. Conn *et al.* have already discussed on the model-based algorithm using SVR in [7]. However, in [7], $\hat{\epsilon}$ is treated as a certain constant without any relation to sample points $\boldsymbol{x}_i,\ i = 1, \ldots, l$. In this paper, we consider that width $\hat{\epsilon}$ corresponds to pointwise accuracy of function evaluation. Then, we can construct a quadratic model function exploiting pointwise accuracy from sample points. Furthermore, we propose update formulae of function evaluation accuracy for reducing the total computation time to find a solution.

The paper is organized as follows. In the next section, we introduce basic concepts for constructing a quadratic model function from sample points, and the result on the quality of the quadratic model function. We also present the derivative-free trust-region algorithm, which is base of our algorithm, and a general SVR. In Section 3, we present our derivative-free trust-region algorithm. In Section 4, we give some numerical results for a numerical test set CUTEr and financial problem. Finally, Section 5 concludes the paper.

We use the following notations throughout the paper. For a vector $\boldsymbol{x} \in \mathbb{R}^n$, $\|\boldsymbol{x}\|$ denotes the Euclidean norm defined by $\|\boldsymbol{x}\| := \sqrt{\boldsymbol{x}^\top \boldsymbol{x}}$. For a symmetric matrix $H \in \mathbb{R}^{n\times n}$, $\lambda_{\min}(H)$ denotes the minimum eigenvalue of $H$, similarly $\lambda_{\max}(H)$ denotes the maximum eigenvalue of $H$. Furthermore, $\|H\|$ denotes the $\ell_2$ norm of $H$ defined by $\|H\| := \lambda_{\max}(H)$. For a set $X$, $|X|$ denotes the number of the elements of $X$.

# 2 Preliminary

In this section, we introduce basic concepts for constructing a quadratic model function from sample points. We also present results on the quality of the constructed quadratic model function as an approximation of the objective function in some region. Moreover, we explain the derivative-free trust-region algorithm proposed by Winfield and a support vector regression, which are key tools for constructing our algorithm proposed in the next section.

In the following subsection, $X := \{\boldsymbol{x}_1, \ldots, \boldsymbol{x}_l\} \subseteq \mathbb{R}^n$ denotes the *sample point set*, and $y_i = f(\boldsymbol{x}_i),\ i = 1, \ldots, l$. We denote the *sample set* composed of them by

$$\hat{S} := \{(\boldsymbol{x}_1, y_1), \ldots, (\boldsymbol{x}_l, y_l)\}. \tag{2.1}$$

## 2.1 Quadratic Model Function from Sample Points

### 2.1.1 Construction of a quadratic model function from sample points

We consider constructing a quadratic model function of $f : \mathbb{R}^n \to \mathbb{R}$ from the sample set $\hat{S}$ by interpolation or regression. First, note that any quadratic function $m : \mathbb{R}^n \to \mathbb{R}$ is written in the form

$$m(\boldsymbol{x}) := \boldsymbol{w}^\top \phi(\boldsymbol{x}),$$

where $\boldsymbol{w} \in \mathbb{R}^N$ is a coeffient vector with $N := \frac{1}{2}(n+1)(n+2)$, and $\phi : \mathbb{R}^n \to \mathbb{R}^N$ is a mapping defined by

$$\phi(\boldsymbol{x}) := \left(1, x_1, \ldots, x_n, x_1^2, x_1x_2, \ldots, x_{n-1}x_n, x_n^2\right)^\top \in \mathbb{R}^N. \tag{2.2}$$

For constructing a reasonable function $m$ from the sample set $\hat{S}$, we require $\boldsymbol{w} \in \mathbb{R}^N$ to satisfy the following condition.

$$\begin{pmatrix} m(\boldsymbol{x}_1) \\ \vdots \\ m(\boldsymbol{x}_l) \end{pmatrix} = \Phi(X)\boldsymbol{w} = \boldsymbol{y} \tag{2.3}$$

where

$$\Phi(X) := \begin{pmatrix} \phi(\boldsymbol{x}_1)^\top \\ \vdots \\ \phi(\boldsymbol{x}_l)^\top \end{pmatrix} \in \mathbb{R}^{l \times N}, \tag{2.4}$$

$$\boldsymbol{y} := (y_1, \ldots, y_l)^\top \in \mathbb{R}^l.$$

In the case of interpolation, the number $l$ of sample points equals to $N$ in general. Then, the matrix $\Phi(X)$ becomes square. Moreover, if $\Phi(X)$ is nonsingular, then there exists a unique $\boldsymbol{w}$ that satisfies (2.3). Hence, we can construct the quadratic model function $m$ by solving the system of linear equations (2.3) with respect to $\boldsymbol{w}$. In addition, we can construct the model instead of solving (2.3) by using Lagrange polynomial interpolation [21] or Newton polynomial interpolation [8].

In the case of regression, $l \geq N$ in general. Thus there may not exist $\boldsymbol{w}$ satisfying the condition (2.3). Then, least squares regression is usually used to obtain a reasonable function $m$. Least squares regression solves the following problem.

$$\begin{aligned} \underset{\boldsymbol{w},\boldsymbol{\xi}}{\text{minimize}} \quad & \sum_{i=1}^{l} \xi_i^2, \\ \text{subject to} \quad & \Phi(X)\boldsymbol{w} = \boldsymbol{y} + \boldsymbol{\xi}. \end{aligned} \tag{2.5}$$

where $\boldsymbol{\xi} = (\xi_1, \dots, \xi_l)^\top \in \mathbb{R}^l$. If rectangular matrix $\Phi(X) \in \mathbb{R}^{l\times N}$ is full column rank, there exists a unique solution of the problem (2.5). When $\Phi(X)$ is not full column rank, Ridge regression is useful. Ridge regression solves the following problem whose objective function is composed of that of least squares regression and the regularized term.

$$\begin{aligned} \underset{\boldsymbol{w},\boldsymbol{\xi}}{\text{minimize}} \quad & \lambda\|\boldsymbol{w}\|^2 + \sum_{i=1}^{l} \xi_i^2, \\ \text{subject to} \quad & \Phi(X)\boldsymbol{w} = \boldsymbol{y} + \boldsymbol{\xi}, \end{aligned} \tag{2.6}$$

where $\lambda > 0$ is a regularized parameter. There exists a unique solution of the problem (2.6) regardless of the condition of $\Phi(X)$.

### 2.1.2 Quality of the quadratic model function

Now we introduce the results in [9, 10] for the quality of the quadratic model function, which is constructed by interpolation or least squares regression.

Let $\Delta$ denote the radius of the smallest enclosing hypersphere containing all sample points $\boldsymbol{x}_i \in X$. In the following discussion, we assume that the hypersphere is centered at origin without loss of generality, and we denote the hypersphere by $B(\Delta) \subseteq \mathbb{R}^n$. Furthermore, let $\hat{X}$ denote the normalized sample points set by the radius $\Delta$, that is,

$$\hat{X} = \{\boldsymbol{x}_1/\Delta, \dots, \boldsymbol{x}_l/\Delta\}.$$

The following theorem shows the error bound between the objective function $f$ and the quadratic model function $m$ constructed by interpolation.

**Theorem 2.1** *[9, Theorem 4.2.] Assume that $f$ is twice continuously differentiable in an open domain $\Omega$ containing $B(\Delta)$ and that $\nabla^2 f$ is Lipschitz continuous in $\Omega$ with constant $L_f > 0$.*

*Then, for all points $\boldsymbol{x} \in B(\Delta)$, we have that*

$$|f(\boldsymbol{x}) - m(\boldsymbol{x})| \le \left( \frac{6 + 9\sqrt{2}}{4} L_f \sqrt{N-1} \|\Phi(\hat{X})^{-1}\| + \frac{1}{6} L_f \right) \Delta^3. \tag{2.7}$$

Theorem 2.1 shows that the quality of the quadratic model function constructed by interpolation depends on the radius $\Delta$ and the singular values of the matrix $\Phi(\hat{X})$.

Next, we introduce a similar result for least squares regression. The discussion for least squares regression is complex a little more than that for interpolation, because $\Phi(\hat{X})$ is not necessarily a square matrix. As in the case of interpolation, $\hat{X}$ denotes the normalized sample points set by $\Delta$. The rectangular matrix $\Phi(\hat{X}) \in \mathbb{R}^{l\times N}$ forms the following reduced singular value decomposition.

$$\Phi(\hat{X}) = U\Sigma V^\top,$$

where $\Sigma \in \mathbb{R}^{N\times N}$ is a diagonal matrix formed by the singular values, and $U \in \mathbb{R}^{l\times N}$ and $V \in \mathbb{R}^{l\times l}$ are the corresponding orthonormal matrices. Then, the following result is known.

**Theorem 2.2** *[10, Theorem 3.2.] Assume that $f$ is twice continuously differentiable in an open domain $\Omega$ containing $B(\Delta)$ and that $\nabla^2 f$ is Lipschitz continuous in $\Omega$ with constant $L_f > 0$.*

*Then, for all points $\boldsymbol{x} \in B(\Delta)$, we have that*

$$|f(\boldsymbol{x}) - m(\boldsymbol{x})| \le \left( \frac{6+9\sqrt{2}}{2} L_f N \sqrt{l}\, \|\Sigma^{-1}\| + \frac{1}{6} L_f \right) \Delta^3. \tag{2.8}$$

Theorem 2.2 shows that the quality of the quadratic model function constructed by least squares regression also depends on the radius $\Delta$ and the singular values of $\Phi(\hat{X})$. Note that the singular value matrix $\Sigma$ is nonsingular if the rectangular matrix $\Phi(\hat{X})$ is full column rank.

From Theorem 2.1, 2.2, we see that, since $\Delta$ and the singular values of $\Phi(\hat{X})$ depend on the position of sample points $X$, it is important to generate a adequate sample points for the quality of the model. For instance, if sample points are generated on a line or on a quadratic curve, then the matrix $\Phi(\hat{X})$ is singular, and hence, the quadratic model function constructed from the sample points is unreliable.

## 2.2 Derivative-Free Trust-Region methods without error

The derivative-free trust-region algorithm proposed by Winfield is built in the idea of the trust-region methods [4]. The outline of the algorithm is as follows.

**Outline of derivative-free trust-region algorithms**

**Step 0** : Initialization for the sample set $\hat{S} \subseteq \mathbb{R}^{n+1}$, initial iterate $\boldsymbol{x}_0$ and trust-region radius $\Delta_0$.

**Step 1** : Construction of the quadratic model $m_k : \mathbb{R}^n \to \mathbb{R}$ from the sample set $\hat{S}$.

**Step 2** : Obtain the minimum $\boldsymbol{x}_k^+$ of the model $m_k$ within the trust region $\{\boldsymbol{x} \mid \|\boldsymbol{x}_k - \boldsymbol{x}\| \le \Delta_k\}$.

**Step 3** : Evaluation of the quality of the current model $m_k$ at $\boldsymbol{x}_k^+$.

**Step 4** : Update of the trust-region radius $\Delta_k$ based on the evaluation in step 3 and on the property of the matrix $\Phi(\hat{X})$.

**Step 5** : Update of the sample set $\hat{S}$.

**Step 6** : Update of the current iterate $\boldsymbol{x}_k$, and go to Step 1.

The algorithm described above is different from general *derivative-based* trust-region methods at **Step 1** in which a quadratic model function is constructed, at **Step 4** in which the trust-region radius is updated, and at **Step 5** in which the sample set is updated. Notice that Step 5 in the algorithm does not exist in ordinary derivative-based trust-region algorithms. Here we explain derivative-free trust-region algorithm while comparing with the derivative-based trust-region algorithm.

In **Step 1**, derivative-based trust-region algorithm constructs a quadratic model function $m_k$ from the information of $f$ at the current point $\boldsymbol{x}_k$ only, that is, $m_k$ is given by

$$m_k(\boldsymbol{x}_k + \boldsymbol{p}) = f(\boldsymbol{x}_k) + \boldsymbol{g}_k^\top \boldsymbol{p} + \boldsymbol{p}^\top \boldsymbol{H}_k \boldsymbol{p}$$

where $\boldsymbol{g}_k = \nabla f(\boldsymbol{x}_k) \in \mathbb{R}^n$, $\boldsymbol{H}_k = \nabla^2 f(\boldsymbol{x}_k) \in \mathbb{R}^{n \times n}$. However, when the derivatives of the objective function are not available, the model function must be constructed only from the information of the objective function values. Then, model-based algorithm constructs a quadratic model function $m_k$ from the sample set $\hat{S}$ by using some techniques explained in Section 2.1.

The operations in **Steps 2 and 3** are found in the derivative-base trust-region algorithms. So we give a brief description for them.

In Step 2, the minimum $\boldsymbol{x}_k^+$ of the quadratic model function $m_k$ is obtained within the region $\mathcal{B}_k$ where the model is regarded as an adequate representation of the objective function, i.e.,

$$\boldsymbol{x}_k^+ = \arg\min_{\boldsymbol{x}\in\mathcal{B}_k} m_k(\boldsymbol{x}), \tag{2.9}$$

and

$$\mathcal{B}_k = \{\boldsymbol{x} \in \mathbb{R}^n \mid \|\boldsymbol{x} - \boldsymbol{x}_k\| \leq \Delta_k\}.$$

The hypersphere $\mathcal{B}_k$ is called the *trust region*, and $\Delta_k > 0$ is the *trust-region radius.* For solving (2.9) there exist approximate methods such as the dogleg method, two-dimensional subspace method and Steihaug's approach, and the exact method [4, Algorithm 4.4]. In the followings, we denote the objective function value at $\boldsymbol{x}_k^+$ by $y_k^+ := f(\boldsymbol{x}_k^+)$.

In Step 3, the algorithm calculate the ratio

$$\rho_k := \frac{y_k - y_k^+}{m_k(\boldsymbol{x}_k) - m_k(\boldsymbol{x}_k^+)} = \frac{\text{acutual reduction}}{\text{predicted reduction}}$$

to evaluate the quality of the model function. We note that the denominator of $\rho_k$, namely, the predicted reduction, is always nonnegative since the point $\boldsymbol{x}_k^+$ is the minimizer of the model over a region that includes the point $\boldsymbol{x}_k$. In fact, we can evaluate the quality of the model function using $\rho_k$ . For instance, when the ratio $\rho_k$ is negative, or positive but close to zero, we may consider the quality of the model is worse.

Recall that the mapping $\Phi$ and the set $\hat{X}$ are defined in Section 2.1, and we assume that the quadratic model function is constructed by some interpolation method[1].

In **Step 4**, the trust-region radius $\Delta_k$ is adjusted according to the ratio $\rho_k$ and the regularity of the matrix $\Phi(\hat{X})$. In derivative-based trust-region algorithms, since Taylor's theorem indicates that the quadratic model function better fits $f(\cdot)$ in a smaller neighborhood of $\boldsymbol{x}_k$. Hence, the quality of the model can be improved by reducing the trust-region radius. However, in the case of interpolation, Theorem 2.1 indicates that the quality of the model depends on not only the $\Delta$, which is the radius of the hypersphere containing all sample points, but also $\|\Phi(\hat{X})^{-1}\|$. Thus, the quality of the model can not be improved only by reducing the trust-region radius $\Delta_k$ without updating the sample set. Therefore, the trust-region radius is updated in consideration of $\rho_k$ and $\|\Phi(\hat{X})^{-1}\|$ as follows. Let $\eta_0$ and $\eta_1$ be constants such that $0 < \eta_0 < \eta_1 < 1$.

- $\rho_k \geq \eta_1$: Expand the trust-region radius $\Delta_k$.
- $\rho_k < \eta_0$ and $\|\Phi(\hat{X})^{-1}\|$ is sufficiently small: Reduce the trust-region radius $\Delta_k$.
- $\rho_k \leq \eta_0$ and $\Phi(\hat{X})$ is close to singular: Improve the quality of the model by updating the sample set to reduce the value of $\|\Phi(\hat{X})^{-1}\|$ in Step 5.

[1]To our knowledge, there is no concrete discussion on model-based algorithm via a regression technique.

In **Step 5**, the sample set is updated. First of all, the trial point $\boldsymbol{x}_k^+$, solution of trust-region subproblem (2.9), is added to the sample set $\hat{S}$ if one of the following three statements holds.

- The number $|\hat{S}|$ of sample points is less than an upper bound of the number of sample points[2].
- The reduction of the objective function is sufficient, that is, $\rho_k \geq \eta_0$.
- $\rho_k < \eta_0$ but $\|\Phi(\hat{X})^{-1}\|$ becomes small by adding the trial point $\boldsymbol{x}_k^+$.

Next, the sample set is updated to improve the quality of the model as follows:

- $\rho_k < \eta_0$ and the matrix $\Phi(\hat{X})$ is close to singular: Update the sample set $\hat{S}$ for reducing the value $\|\Phi(\hat{X})^{-1}\|$ by some suitable method (see [5] for details).
- $\rho_k < \eta_0$ and $\|\Phi(\hat{X})^{-1}\|$ is sufficiently small: In this case the trust region is shrinked in Step 4. If some sample point are too far away from the current point $\boldsymbol{x}_k$, remove them and create new sample points in the trust region by suitable method.

Now, we give a concrete derivative-free trust-region algorithm as follows.

**a derivative-free trust-region model-based algorithm**

**Step0: Initialization.**
Set the trust-region parameters $\eta_0, \eta_1, \gamma_{dec}, \gamma_{inc}$ such as

$$0 < \eta_0 \leq \eta_1 < 1, \quad \text{and } 0 < \gamma_{dec} < 1 \leq \gamma_{inc},$$

Choose a starting point $\boldsymbol{x}_s$. Choose an initial sample set $\hat{S}$ containing $\boldsymbol{x}_s$. Determine $\boldsymbol{x}_0 \in X$ such that $f(\boldsymbol{x}_0) = \min_{\boldsymbol{x}_i \in X} f(\boldsymbol{x}_i)$. Choose an initial trust-region radius $\Delta_0 > 0$. Set $k = 0$.

**Step 1: Model building.**
Using the sample set $\hat{S}$, build the model $m_k$.

**Step 2: Minimization of the model within the trust-region.**
Compute the point $\boldsymbol{x}_k^+$ such that

$$m_k(\boldsymbol{x}_k^+) = \min_{\boldsymbol{x} \in \mathcal{B}_k} m_k(\boldsymbol{x}).$$

Compute $y_k^+ := f(\boldsymbol{x}_k^+)$.

**Step 3: Evaluation of the quality of the current model** Compute the ratio

$$\rho_k := \frac{y_k - y_k^+}{m_k(\boldsymbol{x}_k) - m_k(\boldsymbol{x}_k^+)}.$$

**Step 4: Update the trust-region radius.**

- *Successful step*: If $\rho_k \geq \eta_1$, then set $\Delta_{k+1} = \gamma_{inc}\Delta_k$.

[2]In the case of quadratic interpolation, the number of sample set must be equals to or less than the number of the coefficients of a quadratic function $\frac{1}{2}(n+1)(n+2)$

- *Unsuccessful step*: If $\rho_k < \eta_0$ and $\|\Phi(\hat{X})^{-1}\|$ is small enough, then set $\Delta_{k+1} = \gamma_{dec}\Delta_k$.
- Otherwise, set $\Delta_{k+1} = \Delta_k$.

**Step 5: Update the sample set.**
If $\rho_k \geq \eta_1$, then *insert* $(\boldsymbol{x}_k^+, y_k^+)$ in $\hat{S}$, and drop one of the existing sample points if $|\hat{S}| = N$.
If $\rho_k < \eta_1$ and $\Phi(\hat{X})$ is nearly singular, *improve* the model $m_k$ by updating sample points in $\hat{S}$.

**Step 6: Update the current iterate.**
Determine $\hat{\boldsymbol{x}}_k$ such that

$$\hat{\boldsymbol{x}}_k = \arg \min_{\boldsymbol{x}_i \in X, \boldsymbol{x}_i \neq \boldsymbol{x}_k} f(\boldsymbol{x}_i).$$

Then, calculate the ratio

$$\hat{\rho}_k := \frac{f(\boldsymbol{x}_k) - f(\hat{\boldsymbol{x}}_k)}{m_k(\boldsymbol{x}_k) - m_k(\boldsymbol{x}_k^+)}.$$

- *Successful step:* If $\hat{\rho}_k \geq \eta_0$, then set $\boldsymbol{x}_{k+1} = \hat{\boldsymbol{x}}_k$.
- *Unsuccessful step:* Otherwise, set $\boldsymbol{x}_{k+1} = \boldsymbol{x}_k$.

Increment $k$ by one, and go to Step 1.

**End of algorithm**

Conn *et al.* established global convergence of the algorithm described in [8] under suitable assumptions, in which he assumed that the objective function $f$ was bounded below and twice continuously differentiable and others. In the proof, $\|\nabla f(\boldsymbol{x}_k)\|$ converges to zero as long as the trust-region radius $\Delta_k$ converges to zero. Due to the fact, $\Delta_k \leq \epsilon_\Delta$ with a small positive constant $\epsilon_\Delta$ is employed as the termination criterion.

Conn and Toint claim that the model-based algorithm is robust for small errors in objective function from numerical results in [5, 7], in which the errors was generated in uniformly random number. However, to our knowledge, there exists no algorithm that controls the accuracy of function evaluation in the culculation.

## 2.3 Support Vector Regression

In this subsection, we introduce Support Vector Regression (SVR) that is one of the methods constructing a quadratic model function of $f$ from sample sets. We assume that the sample set $\hat{S}$ is given as (2.1), which is composed of sample points $X$ and corresponding objective function values $\{y_1, \ldots, y_l\}$.

We consider *linear $\epsilon$-insensitive loss function* as follows:

$$|y - m(\boldsymbol{x})|_\epsilon := \max(0, |y - m(\boldsymbol{x})| - \epsilon). \tag{2.10}$$

As in Figure 1, the linear $\epsilon$-insensitive loss function linearly penalizes the residual between the model function value $m(\boldsymbol{x})$ and true function value $y$ by more than $\epsilon$. As in Section 2.1, a quadratic model function can be written as $m(\boldsymbol{x}) = \boldsymbol{w}^\top \phi(\boldsymbol{x})$ where $\boldsymbol{w} \in \mathbb{R}^N$ is a coefficient vector and $\phi$ is defined

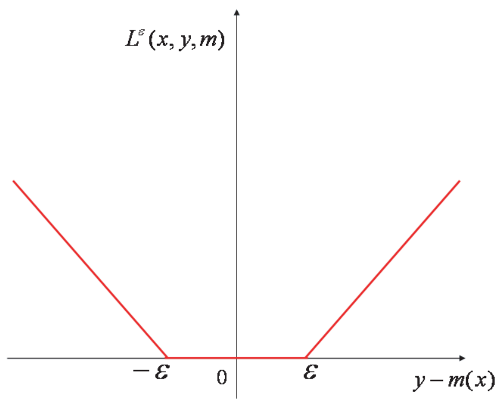


Figure 1: Linear $\epsilon$-insensitive loss function

by (2.2). Then, the problem of finding the model function $m$ which minimize the sum of linear $\epsilon$-insensitive loss function values at all sample points can be formulated as

$$\underset{\boldsymbol{w}\in\mathbb{R}^N}{\text{minimize}} \quad \frac{1}{2}\|\boldsymbol{w}\|^2 + C\sum_{i=1}^{l} |\boldsymbol{w}^\top\phi(\boldsymbol{x}_i) - y_i|_\epsilon, \tag{2.11}$$

where $\epsilon$ and $C$ are positive parameters decided respectively according to the character of sample points and regression purpose (See Figure 2). The method obtaining a regression function $m$ by solving

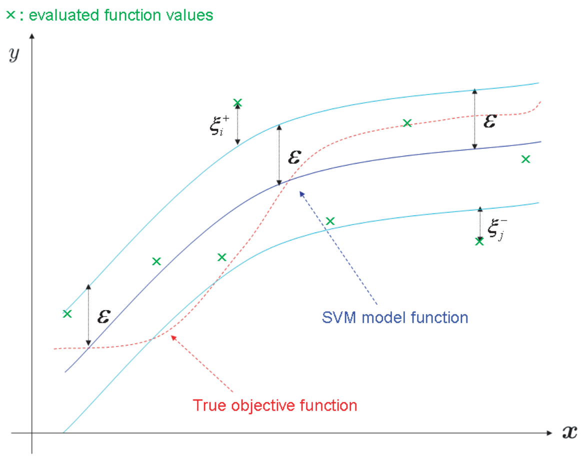


Figure 2: Image of SVR

the minimization problem (2.11) is called Support Vector Regression (SVR). By introducing the artificial variables $\boldsymbol{\xi}^+,\ \boldsymbol{\xi}^- \in \mathbb{R}^l$, the problem (2.11) can be reformulated into the following quadratic programming problem.

$$\begin{aligned} \underset{\boldsymbol{w},\boldsymbol{\xi}^+,\boldsymbol{\xi}^-}{\text{minimize}} \quad & \frac{1}{2}\|\boldsymbol{w}\|^2 + C\sum_{i=1}^{l}(\xi_i^+ + \xi_i^-), \\ \text{subject to} \quad & y_i - \boldsymbol{w}^\top \phi(\boldsymbol{x}_i) \le \epsilon + \xi_i^+, \\ & \boldsymbol{w}^\top \phi(\boldsymbol{x}_i) - y_i \le \epsilon + \xi_i^-, \\ & \xi_i^+,\ \xi_i^- \ge 0, \quad i = 1,\dots,l. \end{aligned} \tag{2.12}$$

Moreover, the dual problem of (2.12) is written as follows [13].

$$\begin{aligned} \underset{\boldsymbol{\alpha}^+,\boldsymbol{\alpha}^-}{\text{minimize}} \quad & \frac{1}{2}\begin{pmatrix} \boldsymbol{\alpha}^+ \\ \boldsymbol{\alpha}^- \end{pmatrix}^\top \begin{pmatrix} \boldsymbol{K} & -\boldsymbol{K} \\ -\boldsymbol{K} & \boldsymbol{K} \end{pmatrix} \begin{pmatrix} \boldsymbol{\alpha}^+ \\ \boldsymbol{\alpha}^- \end{pmatrix} + \begin{pmatrix} \epsilon\boldsymbol{1} - \boldsymbol{y} \\ \epsilon\boldsymbol{1} + \boldsymbol{y} \end{pmatrix}^\top \begin{pmatrix} \boldsymbol{\alpha}^+ \\ \boldsymbol{\alpha}^- \end{pmatrix}, \\ \text{subject to} \quad & \boldsymbol{0} \le \boldsymbol{\alpha}^+, \boldsymbol{\alpha}^- \le C\boldsymbol{1}, \end{aligned} \tag{2.13}$$

where, $\boldsymbol{y} := (y_1,\dots,y_l)^\top \in \mathbb{R}^l$, $\boldsymbol{1} = (1,\dots,1)^\top \in \mathbb{R}^l$, $\boldsymbol{K} = \boldsymbol{P}\boldsymbol{P}^\top$, and

$$\boldsymbol{P} := \begin{pmatrix} \phi(\boldsymbol{x}_1)^\top \\ \vdots \\ \phi(\boldsymbol{x}_l)^\top \end{pmatrix} \in \mathbb{R}^{l \times N}.$$

Since the dual problem (2.13) is also quadratic programming, it can be solved by some existing solver. Let $\hat{\boldsymbol{\alpha}}^+, \hat{\boldsymbol{\alpha}}^-$ be the solution of the problem (2.13), then the coefficients of the quadratic model function $m$ is given as $\boldsymbol{w}^* = \boldsymbol{P}^\top(\hat{\boldsymbol{\alpha}}^+ - \hat{\boldsymbol{\alpha}}^-)$ [13].

At end of this subsection, we note the relation among SVR, least squares regression, and ridge regression. There also exists quadratic SVR that minimizes the sum of the quadratic $\epsilon$-insensitive loss function, $\frac{1}{2}\|\boldsymbol{w}\|^2 + C\sum_{i=1}^{l}\{(\xi_i^+)^2 + (\xi_i^-)^2\}$, at all sample points. When $\epsilon = 0$, the quadratic SVR is equivalent to ridge regression. Furthermore, when $C = \infty$, it reduced to least squares regression.

## 3 Derivative-Free Optimization with Controllable Error

In this section, we propose a derivative-free trust-region algorithm for the problem (1.1) in the situation of (S1)—(S3).

To express the situation of (S1) and (S2) mathematically, we assume that for given a point $\boldsymbol{x} \in \mathbb{R}^n$ and an accuracy $\epsilon$ of the function evaluation, we can obtain estimation value $f_\epsilon(\boldsymbol{x})$ of the objective value such that

$$f(\boldsymbol{x}) - \epsilon \le f_\epsilon(\boldsymbol{x}) \le f(\boldsymbol{x}) + \epsilon. \tag{3.1}$$

Under (S3), the time and the accuracy of function evaluation are in the relation of the trade-off. In that case, it would be reasonable to set the accuracy to be low at a point far from a solution, and to set it to be high at a point in the neighborhood of a solution. Based on this idea, we propose a derivative-free trust-region algorithm that updates the accuracy of objective function evaluation at each iteration. For this purpose, the following two techniques must be developed.

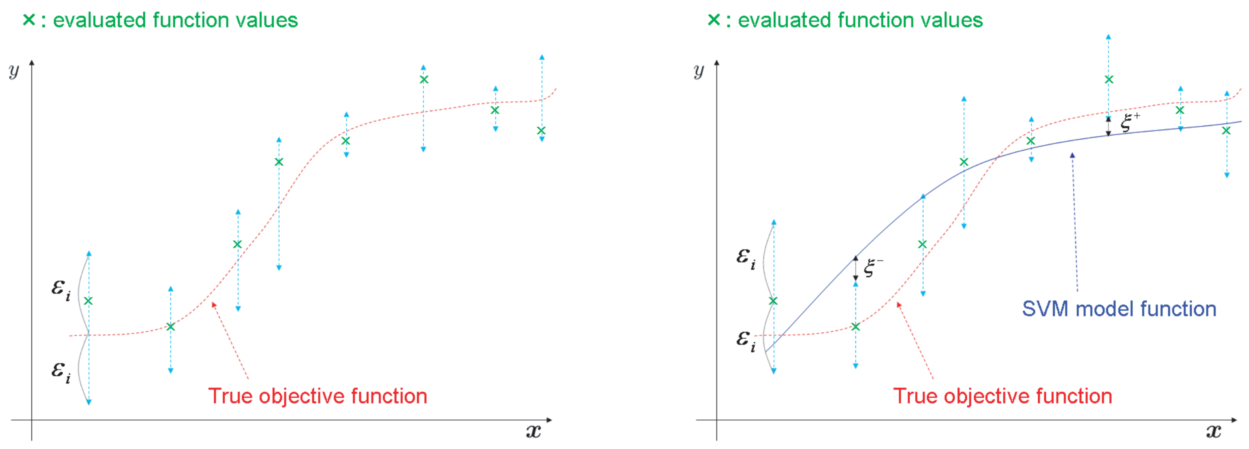
1. Construction of the model function exploiting the pointwise accuracies.

2. Control of the accuracy of the evaluated objective function value.

In the following two subsections, we consider the two issues.

## 3.1 Construction of a Quadratic Model from Sample Points with Pointwise Errors by SVR

In order to exploit different accuracies of each sample points, we construct a quadratic model function by the SVR.



(a) Evaluation of function values (b) Support Vector Regression with various errors

Figure 3: Image of PSVR

Now, the sample points set $X = \{\boldsymbol{x}_1, \ldots, \boldsymbol{x}_l\}$ is given. Moreover, we assume that the evaluated objective values $f_{\epsilon_i}(\boldsymbol{x}_i)$ are calculated with the accuracy $\epsilon_i$ for all $\boldsymbol{x}_i \in X$ (Figure 3 (a)).

$$f(\boldsymbol{x}_i) - \epsilon_i \le f_{\epsilon_i}(\boldsymbol{x}_i) \le f(\boldsymbol{x}_i) + \epsilon_i, \ i = 1, \ldots, l, \tag{3.2}$$

Since the inequalities (3.2) can be written as

$$f_{\epsilon_i}(\boldsymbol{x}_i) - \epsilon_i \le f(\boldsymbol{x}_i) \le f_{\epsilon_i}(\boldsymbol{x}_i) + \epsilon_i, \ i = 1, \ldots, l, \tag{3.3}$$

we may see that the true objective values $f(\boldsymbol{x}_i)$ lie in $[f_{\epsilon_i}(\boldsymbol{x}_i) - \epsilon_i, \ f_{\epsilon_i}(\boldsymbol{x}_i) + \epsilon_i]$. Therefore, we impose the condition (3.3) on the model function $m$, that is,

$$f_{\epsilon_i}(\boldsymbol{x}_i) - \epsilon_i \le m(\boldsymbol{x}_i) \le f_{\epsilon_i}(\boldsymbol{x}_i) + \epsilon_i, \ i = 1, \ldots, l. \tag{3.4}$$

As the idea of SVR (2.11), we consider the regression problem which minimizes the linear $\epsilon_i$-insensitive loss function for each sample $\boldsymbol{x}_i$. The problem is formulated as follows.

$$\underset{\boldsymbol{w} \in \mathbb{R}^N}{\text{minimize}} \quad \frac{1}{2}\|\boldsymbol{w}\|^2 + \sum_{i=1}^{l} C_i |\boldsymbol{w}^\top \phi(\boldsymbol{x}_i) - y_i|_{\epsilon_i},$$

where $|\cdot|_\epsilon$ is the linear $\epsilon$-insensitive loss function defined by (2.10), and $y_i := f_{\epsilon_i}(\boldsymbol{x}_i), \ i = 1, \ldots, l$. Note that the $C_i$ and $\epsilon_i$ are distinct for each $i = 1, \ldots, l$ while $C$ and $\epsilon$ are same for $i$ in the original SVR (2.11)

As the discussion in section 2.3, this problem can be reformulated as follows.

$$\begin{aligned} \underset{\boldsymbol{w},\boldsymbol{\xi}^+,\boldsymbol{\xi}^-}{\text{minimize}} \quad & \frac{1}{2}\|\boldsymbol{w}\|^2 + \sum_{i=1}^{l} C_i(\xi_i^+ + \xi_i^-), \\ \text{subject to} \quad & y_i - \boldsymbol{w}^\top \phi(\boldsymbol{x}_i) \le \epsilon_i + \xi_i^+, \\ & \boldsymbol{w}^\top \phi(\boldsymbol{x}_i) - y_i \le \epsilon_i + \xi_i^-, \\ & \xi_i^+,\ \xi_i^- \ge 0, \quad i = 1, \ldots, l. \end{aligned} \tag{3.5}$$

Here $C_i \ge 0,\ i = 1, \ldots, l$ are the penalty parameters corresponding to the constraints (3.4) of sample points. For some index $i$, the bigger penalty parameter $C_i$ makes the constraints (3.4) for $i$ the more satisfied. Thus, by adjusting the values of $C_i$, we can weight a high-accuracy sample more than low-accuracy sample in the problem (3.5). Furthermore, we may also weight samples that lie in the trust region more than ones which lies outside it. One of concrete settings for $C_i$ is given in Section 3.3.

As the discussion in Section 2.3, the dual problem of (3.5) is written as follows.

$$\begin{aligned} \underset{\boldsymbol{\alpha}^+,\boldsymbol{\alpha}^-}{\text{minimize}} \quad & \frac{1}{2}\begin{pmatrix} \boldsymbol{\alpha}^+ \\ \boldsymbol{\alpha}^- \end{pmatrix}^\top \begin{pmatrix} \boldsymbol{K} & -\boldsymbol{K} \\ -\boldsymbol{K} & \boldsymbol{K} \end{pmatrix} \begin{pmatrix} \boldsymbol{\alpha}^+ \\ \boldsymbol{\alpha}^- \end{pmatrix} + \begin{pmatrix} \boldsymbol{\epsilon} - \boldsymbol{y} \\ \boldsymbol{\epsilon} + \boldsymbol{y} \end{pmatrix}^\top \begin{pmatrix} \boldsymbol{\alpha}^+ \\ \boldsymbol{\alpha}^- \end{pmatrix}, \\ \text{subject to} \quad & \boldsymbol{0} \le \boldsymbol{\alpha}^+, \boldsymbol{\alpha}^- \le \boldsymbol{C}, \end{aligned} \tag{3.6}$$

where $\boldsymbol{C} := (C_1, \ldots, C_l)^\top \in \mathbb{R}^l$ and $\boldsymbol{\epsilon} := (\epsilon_1, \ldots, \epsilon_l)^\top \in \mathbb{R}^l$, $\boldsymbol{K} = \boldsymbol{P}\boldsymbol{P}^\top$, and

$$\boldsymbol{P} = \begin{pmatrix} \phi(\boldsymbol{x}_1)^\top \\ \vdots \\ \phi(\boldsymbol{x}_l)^\top \end{pmatrix} \in \mathbb{R}^{l \times N}.$$

Since the dual problem (3.6) is a quadratic programming, we can obtain its solution $(\hat{\boldsymbol{\alpha}}^+, \hat{\boldsymbol{\alpha}}^-)$ by using some existing solver. With this solution, the quadratic model function of the objective function is determined as follows:

$$\begin{aligned} \boldsymbol{w}^* &= \boldsymbol{P}^\top(\hat{\boldsymbol{\alpha}}^+ - \hat{\boldsymbol{\alpha}}^-), \\ m(\boldsymbol{x}) &= (\boldsymbol{w}^*)^\top \phi(\boldsymbol{x}) \end{aligned}$$

From the above discussion, we see that not only $(\boldsymbol{x}_i, y_i),\ i = 1, \ldots, l$ but also the accuracy $\epsilon_i$ and the penalty parameters $C_i,\ i = 1, \ldots, l$ are important to construct the quadratic model function $m$. Hence, the sample set $S$ used in the proposed algorithm is composed of four factors: " sample point ", " objective value ", " accuracy " and " penalty parameter".

$$S := \{(\boldsymbol{x}_1, y_1, \epsilon_1, C_1), \ldots, (\boldsymbol{x}_l, y_l, \epsilon_l, C_l)\}.$$

## 3.2 Control of Errors

In this subsection, we consider how to control the accuracy for reducing the total computation time of the algorithm. As mentioned in introduction, we want the accuracy to be low for a point far from a solution and to be high for a point near the solution. Since the derivative of $f$ is not available, we cannot measure the distance between a point $\boldsymbol{x}$ and a solution by $\|\nabla f(\boldsymbol{x})\|$. Conn *et al.* showed that $\|\nabla f(\boldsymbol{x})\| \to 0$ and $\Delta_k \to 0$ as $k \to \infty$ when $\boldsymbol{x}_k$ is generated by their derivative-free trust-region algorithm with exact function evaluations. This fact indicates that we may use the trust-region radius $\Delta_k$ for the measure. So we control $\epsilon_k \to 0$ as $\Delta_k \to 0$. However, how fast do we make $\epsilon_k$ reduced when $\Delta_k$ goes to zero?

Now, we give the answer. Let $\boldsymbol{x}_0$ denote the solution of the trust-region subproblem (2.9). Suppose that $\boldsymbol{x}_0$ is a stationary point of the model $m$, that is, $\nabla m(\boldsymbol{x}_0) = \mathbf{0}$, and that there exist $n+1$ sample points $\boldsymbol{x}_0, \boldsymbol{x}_1, \ldots, \boldsymbol{x}_n$ in the trust region with radius $\Delta_k$. Let $\epsilon$ denote the largest accuracies of the samples in the trust region, i.e., $\epsilon = \max_{i \in \{0,\ldots,n\}} \epsilon_i$. Then, under some assumptions Theorem A.1 in Appendix A shows that there exist constants $M_1 > 0,\ M_2 > 0$ such that

$$\|\nabla f(\boldsymbol{x}_0)\| \leq M_1 \Delta_k + M_2(\epsilon/\Delta_k). \tag{3.7}$$

Suppose that $\epsilon$ is large enough compared to $\Delta_k$. Then, (3.7) does not guarantee that $\boldsymbol{x}_0$ is a stationary point, even if $\Delta_k \to 0$. Contrary, suppose that $\epsilon = o(\Delta_k)$. Then, if $\Delta_k$ is small enough, then $\|\nabla f(\boldsymbol{x}_0)\|$ is also small enough from (3.7). Hence, we may suppose that $\boldsymbol{x}_0$ is a stationary point when $\Delta_k$ is small. From the above discussion, we recommend that $\epsilon_k$ is set to be $o(\Delta_k)$. We consider concrete control method of $\epsilon_k$ by numerical experiments in Section 4.

## 3.3 Derivative-Free Trust-Region algorithm with Controlling Errors

We propose the following algorithm DFTR-CE.

**Algorithm DFTR-CE**

**Step 0: Initialization.**

Set the trust-region parameters $\eta_0, \eta_1, \gamma_{dec}, \gamma_{inc}$ such that

$$0 < \eta_0 \leq \eta_1 < 1, \quad \text{and } 0 < \gamma_{dec} < 1 \leq \gamma_{inc}.$$

Set an initial accuracy $\epsilon_0$ and an initial penalty parameter $C_0$. Set an upper bound $N_u$ for the number of sample set. Choose an initial sample set $S$ and sample point set $X$. Determine $\boldsymbol{x}_0 \in X$ such that $\boldsymbol{x}_0 = \arg\min_{\boldsymbol{x}_i \in X} f_{\epsilon_0}(\boldsymbol{x}_i)$. Choose an initial trust-region radius $\Delta_0 > 0$. Set $k = 0$.

**Step 1: Construction of the quadratic model with pointwise errors by SVR.**

Using the sample set $S$, build the model $m_k$ by solving the dual problem (3.6). If

$$\max\left\{\|\nabla m_k(\boldsymbol{x}_k)\|, -\lambda_{\min}(\nabla^2 m_k(\boldsymbol{x}_k))\right\} < \epsilon_m, \tag{3.8}$$

then set $\rho_k = -1$ and go to step 5.

**Step 2: Minimization of the model within the trust-region.**

Compute the point $\boldsymbol{x}_{k^*}$ such that

$$\boldsymbol{x}_{k^*} = \arg\min_{\boldsymbol{x}_k \in \mathcal{B}_k} m_k(\boldsymbol{x}).$$

Evaluate $y_{k^*} := f_{\epsilon_{k^*}}(\boldsymbol{x}_{k^*})$ with accuracy $\epsilon_{k^*} := \epsilon_k$.

**Step 3: Evaluation of the quality of the current model.**

Compute the ratio

$$\rho_k := \frac{y_k - y_{k^*}}{m_k(\boldsymbol{x}_k) - m_k(\boldsymbol{x}_{k^*})}.$$

**Step 4: Update of the trust-region radius.**

- *Successful step*: If $\rho_k \geq \eta_1$, then set $\Delta_{k+1} = \gamma_{inc}\Delta_k$.
- *Unsuccessful step*: If $\rho_k < \eta_0$, then set $\Delta_{k+1} = \gamma_{dec}\Delta_k$.
- Otherwise, set $\Delta_{k+1} = \Delta_k$.

**Step 5: Adjustment of the accuracy $\epsilon_k$ and the penalty parameter $C_k$.**

Update the parameters $\epsilon_k$ and $C_k$ by

$$\begin{aligned} \epsilon_{k+1} &= o(\Delta_k), \\ C_{k+1} &= C_0/\epsilon_{k+1}. \end{aligned}$$

**Step 6: Update of the current iterate.**

Set the next iterate $\boldsymbol{x}_{k+1} = \boldsymbol{x}_{\tilde{\jmath}}$, where $\tilde{\jmath} = \arg\min_{j\in\{1,\dots,l\}\cup\{k^*\}}(y_j + \epsilon_j)$.

**Step7: Update of the sample set.**

- If $|S| < N_u$, then *insert* $(\boldsymbol{x}_{k^*}, y_{k^*}, \epsilon_{k^*}, C_k)$ in $S$, otherwise if $y_{\tilde{\imath}} + \epsilon_{\tilde{\imath}} \geq y_{k^*} + \epsilon_k$, insert the sample of $\boldsymbol{x}_{k^*}$ and remove the $\tilde{\imath}$ th sample, where $\tilde{\imath} := \arg_{i\in\{1,\dots,l\}\setminus\{k\}} \max(y_i + \epsilon_i)$.
- If the the number of sample points in the new trust region is less than $n+1$, remove the sample point which is the farest from the current point $\boldsymbol{x}_{k+1}$ and generate new sample in the new trust region until there are at least $n+1$ points in the trust region.
- Set the penalty patameters $C_i$ corresponding to the points $\boldsymbol{x}_i$ such that $\|\boldsymbol{x}_i - \boldsymbol{x}_{k+1}\| > \Delta_{k+1}$ to,

$$C_i = 10^{-\frac{\|x_i - x_k\|}{\Delta_k}} C_0.$$

Increment $k$ by one and go to step 1.

**<u>End of algorithm.</u>**

We explain the details of the algorithm DFTR-CE.

In **Step 1**, if (3.8) hold, then the current model function is almost a constant function. When the model function is constant for all $\boldsymbol{x} \in \mathbb{R}^n$, its gradient equals zero. In that case, if $\Delta_k$ is sufficiently small, then the current iterate $\boldsymbol{x}_k$ can be considered to be a solution of the problem. Otherwise the trust-region radius is reduced and the sample set is reconstructed in the resulted trust region.

In **Step 5**, the penalty parameter $C_k$ is set to be $C_0/\epsilon_k$ so that penalty parameter becomes the larger for the higher accurate sample. As a result, the constraint (3.4) of SVR tends to be satisfied for a high accurate sample. Of course, we may use another update method of setting $C_k$.

In **Step 6**, the current point is updated taking into account both estimation $y_i$ and accuracy $\epsilon_i$.

Note that after the current iterate and the trust-region radius is updated, the sample set is updated in Step 7.

The first update in Step 7 is insertion of the solution $\boldsymbol{x}_{k^*}$ of the trust-region subproblem to the sample set. In the case of interpolation, the number of the sample set $l$ must be equal to the number $N$ of the coefficients of the quadratic model function. On the other hand, in the case that the model function is constructed via regression, $l$ may be larger than $N$. Therefore, if possible, we add $\boldsymbol{x}_{k^*}$ to the sample set $S$ in order to use the information on the objective function obtained by the algorithm. However, the more the number of the elements of $S$ increases, the larger dual problem (3.6) becomes. Then, it becomes harder to solve the problem (3.6). From the viewpoint of applications, it is natural to set some upper bound for the number of the sample set. According to this, we set the bound $N_u$ depending on the dimension $n$ of the variable. We add the trial sample $(\boldsymbol{x}_{k^*}, y_{k^*}, \epsilon_k, C_k)$ to the sample set $S$ when the number of the sample $l$ is less than $N_u$. When $l = N_u$, a bad point in the sample set $S$ is removed and the trial sample is added to $S$. For example, in Step 7, the sample with the largest evaluation value

$$\tilde{\imath} := \arg_{i\in\{1,\dots,l\}\setminus\{k\}} \max(y_i + \epsilon_i),$$

is removed if its value $y_{\tilde{\imath}} + \epsilon_{\tilde{\imath}}$ is larger than that of the trial point $y_{k^*} + \epsilon_k$. In addition, we replace the sample which is the furthest from $\boldsymbol{x}_{k+1}$ with the trial sample. Of course, it may be effective to replace them using the property of the matrix $\Phi(\hat{X})$ defined in Section 2.1 in order to improve the quality of the model function, which is not included in Algorithm DFTR-CE.

The second update in Step 7 is done for improving the quality of the model function at next iteration. When the trust-region radius is reduced, some sample points may lie outside the new trust region. It indicates that the number of the samples within the neighborhood of $\boldsymbol{x}_{k+1}$ becomes small. Then, the quality of the model function around $\boldsymbol{x}_{k+1}$ might deteriorate. For this reason, the sample set is updated so that there exist at least $n + 1$ sample points in the new trust region. If there does not exist at least $n + 1$ sample points in the trust region updated in Step 4, a new sample point is created in the trust region, and the function value at the point is evaluated with the accuracy $\epsilon_{k+1}$. After this operation, the number of the sample set may becomes larger than $N_u$. Then, the sample which is the furthest from $\boldsymbol{x}_{k+1}$ is removed. We repeat these operations until $n + 1$ sample points exist in the new trust region.

When the number of the sample is larger than $N$, the samples which are far away from $\boldsymbol{x}_{k+1}$ may deteriorate the quality of the model function within the trust region. Thus, we consider reducing the penalty parameters of the sample points which are outside of the trust region to reduce the weight of the samples. In third update of Step 7, we reduce $C_i$ for a sample point $\boldsymbol{x}_i$ outside of the trust region as follows:

$$C_i := 10^{-\frac{\|\boldsymbol{x}_i - \boldsymbol{x}_{k+1}\|}{\Delta_k}} C_0.$$

Of course, a variety of update types can be thought additionally.

In this paper, we concentrate on the unconstrained optimization problem (1.1). We note that the proposed algorithm DFTR-CE can be applied to the easy constrained problem such as box-constrained optimization problem.

$$\begin{array}{rl} \text{minimize} & f(\boldsymbol{x}), \\ \text{subject to} & \boldsymbol{x} \in F \subseteq \mathbb{R}^n, \end{array}$$

where $F := \{\boldsymbol{x} \in \mathbb{R}^n \mid l_i \leq x_i \leq u_i,\ i = 1, \ldots, n\}$ with $l_i < u_i,\ i = 1, \ldots, n$. For this problem, as in [6], we can construct an algorithm just replacing the trust-region subproblem in Step 2 by

$$\boldsymbol{x}_{k^*} = \arg \min_{\boldsymbol{x}_k \in \mathcal{B}_k \cap F} m_k(\boldsymbol{x}).$$

## 4 Numerical Experiences

In this section, we show some numerical experiences with the algorithm DFTR-CE. The algorithm is coded by MATLAB 7.0 and run on a machine with 3.20GHz CPU and 2G memory. First, the influences of the settings of the parameters in the algorithm were examined with several problems chosen from CUTEr [1]. Furthermore, we examined the performance of the algorithm into the financial problem in which the implied volatility was estimated.

In the following, we explain the details of the algorithm DFTR-CE we implemented.

**Settings for trust-region method** We set the trust-region parameters as

$$\eta_0 = 0.1, \quad \eta_1 = 0.75, \quad \gamma_{dec} = 0.5, \quad \gamma_{inc} = 2.0, \quad \Delta_0 = 1.0.$$

Since we suppose that an objective function evaluation is very costly, we adapt the exact method (see [19], pp. 81, Algorithm. 4.4) for solving the trust-region subproblem (2.9).

**How to create the initial sample set** As for the creation of initial sample set in Step 0, we generated $n$ samples randomly within the initial trust-region centered at initial point $\boldsymbol{x}_0$, and we set these sample points as an initial sample set.

**Improvement of the sample set for the next iterate** In Step 7, we have to generate a new sample point in the trust region. In the experiments, we randomly generated a point in the trust region.

### 4.1 Numerical Results for CUTEr

We assume the situation of (S1)—(S3). However, in fact, it takes a great amount of time to test the performance of the algorithm into such problems. From this reason, first, we examined the algorithm into several problems from CUTEr collection. In the problems of CUTEr, an objective function value $f(\boldsymbol{x})$ is accurately calculated at every point $\boldsymbol{x} \in \mathbb{R}^n$. Therefore, for given accuracy $\epsilon$, generating the uniformly random number $e$ in $[-\epsilon, \epsilon]$ or the Gaussian random number $e$ with $N(0, \epsilon/1.96)$, we added it to the true function value $f(\boldsymbol{x})$ in order to simulate inaccurate evaluation of $f_\epsilon(\boldsymbol{x}) = f(\boldsymbol{x}) + e$. Note that the probability that the Gaussian random number with $N(0, \epsilon/1.96)$ falls in $[-\epsilon,\ \epsilon]$ is 95 %. Since we use random number for generating some sample points, we report the average results taken over 10 runs.

The time of evaluation of objective functions in CUTEr is extent that can be almost disregarded. To suppose the situation of (S3), we assume that the time of the objective function evaluation guaranteeing the accuracy $\epsilon$ is $g(\epsilon)$. For instance, $g(\epsilon) = O(1/\epsilon^2)$ in the Monte Carlo simulation. The time required to evaluate an objective value is much larger than that of other operation of the algorithm for the problem we consider. Hence, we define the following "Total time os Function Evaluations (TFE)" as the evaluation scale of the algorithm of the total computation time.

$$\text{TFE} := \sum_{i=1}^{\mathcal{N}} g(\epsilon_i),$$

where $\mathcal{N}$ is the total number of the function evaluations the algorithm performed, and $\epsilon_i,\ i = 1, \dots, \mathcal{N}$ are the accuracies for each function evaluation.

Table 1 lists the CUTEr problems used in the numerical experiments. For the reference of the problems of CUTEr and its characters, we also give the numerical results examined by Conn *et al.* [6, 7] in Table 1. Note that in [6, 7] the error is always given by the uniformly random error in $[-10^{-3}, 10^{-3}]$.

Table 1: Numerical results in [6, 7]

| Problem | # | $\mathcal{N}$ | | f. val. | |
|---|---|---|---|---|---|
| name | var. | DFO[6] | SVM[7] | DFO | SVM |
| SISSER | 2 | 27 | 32 | 7.33e-04 | -5.48e-04 |
| CLIFF | 2 | 38 | 77 | 2.00e-01 | 1.99e-01 |
| ROSENBR | 2 | 95 | 99 | -6.19e-04 | 3.49e-03 |
| HAIRY | 2 | 53 | 46 | 1.99e+01 | 1.99e+01 |
| GROWTHLS | 3 | 155 | 35 | 1.23e+01 | 2.93e+03 |
| GULF | 3 | 52 | 33 | 6.81e+00 | 6.76e+00 |
| PFIT1LS | 3 | 210 | 98 | 4.47e-03 | 6.12e-02 |
| BROWNDEN | 4 | 113 | 87 | 8.57e+04 | 8.59e+04 |
| HART6 | 6 | 134 | 60 | -3.32e+00 | -3.14e+00 |
| MANCINO | 10 | 211 | 104 | -5.19e-04 | 2.54e-03 |
| POWER | 10 | 251 | 179 | 2.27e-02 | 6.91e-03 |
| MOREBV | 10 | 167 | 34 | 1.34e-02 | 1.57e-02 |
| BRYBND | 10 | 394 | 156 | 3.53e-03 | 1.52e-02 |

### 4.1.1 Experiment on the influence of the computation time of function evaluation $g(\epsilon)$

We examine the relation between the total computation time of the proposed algorithm and the time of the function evaluations $g(\epsilon)$. We consider the following three cases.

**(i)** $g(\epsilon) = 1/\sqrt{\epsilon}$: The increase rate at computation time is calm as the accuracy is raised.

**(ii)** $g(\epsilon) = 1/\epsilon$: The computation time is in inverse proportion to the height of the accuracy (corresponding to a quasi-Monte Carlo method).

**(iii)** $g(\epsilon) = 1/\epsilon^2$: The computation time is in inverse proportion to the square fo the height of accuracy (corresponding to a Monte Carlo method).

We updated the accuracy by $\epsilon_k = 0.5\Delta_k^2$. The upper bound of the number of the sample is set to $N_u = 2N$, where $N$ is the number of the coefficients of $n$-dimensional quadratic function. The termination criterion is given as $\Delta_k < 10^{-3}$. Table 2 reports the TFE required for each (i)—(iii).

Table 2: Numerical results for various $g$ with uniform random error

| PROBLEM | TFE | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| name | $g(\epsilon) = 1/\sqrt{\epsilon}$ | | $g(\epsilon) = 1/\epsilon$ | | $g(\epsilon) = 1/\epsilon^2$ | | $\mathcal{N}$ | f. val. |
| SISSER | 2.46e+04 | (19.2) | 1.62e+07 | (9.9) | 1.81e+13 | (6.7) | 133 | 2.22e-06 |
| CLIFF | 3.08e+03 | (4.0) | 1.15e+06 | (2.0) | 4.07e+11 | (1.2) | 36 | 2.02e-01 |
| ROSENBR | 5.50e+04 | (76.0) | 1.64e+07 | (31.3) | 3.23e+12 | (11.7) | 315 | 2.69e-01 |
| HAIRY | 7.35e+03 | (8.4) | 3.22e+06 | (4.2) | 1.42e+12 | (2.4) | 54 | 2.00e+01 |
| GROWTHLS | 5.44e+03 | (6.9) | 2.97e+06 | (4.8) | 1.58e+12 | (4.1) | 39 | 1.45e+03 |
| GULF | 1.04e+04 | (14.4) | 5.82e+06 | (11.1) | 2.60e+12 | (9.5) | 47 | 6.56e+00 |
| PFIT1LS | 7.35e+03 | (10.4) | 4.12e+06 | (8.2) | 1.84e+12 | (7.3) | 49 | 6.87e-01 |
| BROWNDEN | 1.81e+04 | (14.9) | 7.34e+06 | (5.0) | 7.28e+12 | (3.3) | 620 | 8.60e+04 |
| HART6 | 1.98e+04 | (27.3) | 9.83e+06 | (18.7) | 4.32e+12 | (15.7) | 153 | -3.32e+00 |
| MANCINO | 2.66e+04 | (36.7) | 7.48e+06 | (14.3) | 1.93e+12 | (7.0) | 255 | 5.42e-01 |
| POWER | 1.08e+05 | (97.8) | 6.89e+07 | (56.5) | 5.45e+13 | (36.6) | 468 | 4.71e-05 |
| MOREBV | 1.38e+05 | (190.2) | 7.57e+07 | (144.4) | 3.31e+13 | (120.3) | 373 | 4.94e-03 |
| BRYBND | 6.59e+04 | (91.0) | 3.13e+07 | (59.7) | 1.22e+13 | (44.4) | 336 | 1.65e-01 |

Numbers in the brackets shows TFE$/g(\epsilon^*)$, where $\epsilon^*$ is the highest accuracy of all accuracies of the function evaluations the algorithm performed. That is, TFE$/g(\epsilon^*)$ corresponds to the number of the function evaluations with the highest accuracy $\epsilon^*$. "f. val." in the table denotes the minimum values obtained by the algorithm. $\mathcal{N}$ gives the total number of the function evaluations of the algorithm.

Table 2 shows that TFE increases as the time of a function evaluation increases. On the other hand, TFE$/g(\epsilon^*)$ does not change so much. This is because it hardly takes time at the early stage of the algorithm and because it takes much time to evaluate highly accurate function value around the neighborhood of a solution. We also see that the proposed algorithm is equally effective for (i)–(iii) from the view point of TFE$/g(\epsilon^*)$. Thus, TFE was evaluated with $g(\epsilon) = 1/\epsilon$ only in the following experiments to CUTEr.

### 4.1.2 Experiment on the influence of the algorithmic parameters

The performance of the algorithm DFTR-CE depends on the following two matters:

- An upper bound of the number of the sample $N_u$
- Update of the accuracy

We seek suitable settings for these two things via experiments for CUTEr.

**Experiments on upper bound $N_u$** First, we examined in the three settings of $N_u$ as $N$, $2N$, $3N$. The termination criterion was set to $\Delta_k < 10^{-3}$. The accuracy was updated as $\epsilon_k = \min(0.5\Delta_k^2,\ 0.1)$. Tables 3 and 4 report the results for the uniformly random number and the Gaussian random number respectively. "f. val." of these tables gives the minimum values obtained by the algorithm. The shadowed number *.**e+** means "f. val." is relatively large compared with those of other settings.

Table 3: Numerical results for various $N_u$ with uniform random errror

| | $N_u = N$ | | | $N_u = 2N$ | | | $N_u = 3N$ | | |
|---|---|---|---|---|---|---|---|---|---|
| PROBLEM | $\mathcal{N}$ | TFE | f. val. | $\mathcal{N}$ | TFE | f. val. | $\mathcal{N}$ | TFE | f. val. |
| SISSER | 53 | 4.92e+06 | 6.54e-06 | 64 | 8.15e+06 | -5.75e-08 | 60 | 7.66e+06 | 4.78e-06 |
| CLIFF | 34 | 2.92e+06 | 2.01e-01 | 32 | 3.33e+06 | 2.01e-01 | 33 | 3.12e+06 | 2.01e-01 |
| ROSENBR | 306 | 3.84e+07 | 1.49e-01 | 268 | 3.10e+07 | 1.40e-01 | 250 | 3.67e+07 | 5.76e-01 |
| HAIRY | 82 | 7.68e+06 | 6.82e+01 | 56 | 5.37e+06 | 2.00e+01 | 91 | 7.42e+06 | 2.00e+01 |
| GROWTHLS | 52 | 6.24e+06 | 1.71e+03 | 48 | 8.32e+06 | 1.35e+03 | 50 | 5.99e+06 | 1.69e+03 |
| GULF | 52 | 7.46e+06 | 6.56e+00 | 49 | 7.16e+06 | 5.93e+00 | 50 | 6.54e+06 | 6.56e+00 |
| PFIT1LS | 44 | 5.56e+06 | 7.92e-01 | 47 | 7.69e+06 | 3.86e-01 | 44 | 4.22e+06 | 2.65e-01 |
| BROWNDEN | 517 | 5.85e+06 | 8.62e+04 | 509 | 6.78e+06 | 8.64e+04 | 552 | 6.63e+06 | 8.62e+04 |
| HART6 | 146 | 1.50e+07 | -3.30e+00 | 126 | 9.47e+06 | -3.32e+00 | 118 | 8.30e+06 | -3.32e+00 |
| MANCINO | 201 | 3.44e+07 | 2.88e-01 | 296 | 3.67e+07 | 4.97e-02 | 225 | 3.29e+07 | 3.88e-01 |
| POWER | 444 | 5.31e+07 | 2.92e-05 | 415 | 4.61e+07 | 4.20e-05 | 457 | 5.69e+07 | 4.14e-05 |
| MOREBV | 329 | 5.61e+07 | 4.51e-03 | 340 | 5.96e+07 | 4.83e-03 | 339 | 5.83e+07 | 4.75e-03 |
| BRYBND | 383 | 5.12e+07 | 1.88e-01 | 534 | 8.16e+07 | 6.93e-02 | 394 | 4.55e+07 | 1.84e-01 |

Table 4: Numerical results for various $N_u$ with normal distribution error

| | $N_u = N$ | | | $N_u = 2N$ | | | $N_u = 3N$ | | |
|---|---|---|---|---|---|---|---|---|---|
| PROBLEM | $\mathcal{N}$ | TFE | f. val. | $\mathcal{N}$ | TFE | f. val. | $\mathcal{N}$ | TFE | f. val. |
| SISSER | 43 | 3.98e+06 | -1.71e-02 | 52 | 4.58e+06 | -9.82e-05 | 45 | 3.50e+06 | -2.06e-02 |
| CLIFF | 36 | 4.04e+06 | 2.01e-01 | 32 | 3.64e+06 | 2.01e-01 | 35 | 4.38e+06 | 2.01e-01 |
| ROSENBR | 269 | 4.19e+07 | 1.25e+00 | 230 | 3.44e+07 | 1.52e-01 | 244 | 4.07e+07 | 6.08e-01 |
| HAIRY | 121 | 7.77e+06 | 2.00e+01 | 118 | 7.34e+06 | 2.00e+01 | 78 | 4.52e+06 | 2.00e+01 |
| GROWTHLS | 50 | 5.50e+06 | 2.13e+03 | 48 | 5.11e+06 | 1.41e+03 | 54 | 9.70e+06 | 1.96e+03 |
| GULF | 62 | 1.09e+07 | 6.59e+00 | 90 | 2.22e+07 | 6.28e+00 | 50 | 1.02e+07 | 6.55e+00 |
| PFIT1LS | 72 | 2.23e+07 | 7.94e-01 | 48 | 6.38e+06 | 4.59e-01 | 50 | 7.28e+06 | 1.11e+00 |
| BROWNDEN | 532 | 5.75e+06 | 8.62e+04 | 519 | 5.44e+06 | 8.63e+04 | 536 | 6.09e+06 | 8.63e+04 |
| HART6 | 125 | 1.18e+07 | -3.32e+00 | 135 | 1.46e+07 | -3.32e+00 | 172 | 1.83e+07 | -3.25e+00 |
| MANCINO | 224 | 2.76e+07 | 4.13e-01 | 264 | 5.65e+07 | 6.38e-02 | 236 | 3.15e+07 | 2.65e-01 |
| POWER | 354 | 2.50e+07 | 8.78e-04 | 400 | 4.54e+07 | 3.63e-05 | 387 | 3.94e+07 | -1.02e-03 |
| MOREBV | 325 | 5.40e+07 | 5.30e-03 | 286 | 3.52e+07 | 6.00e-03 | 318 | 4.95e+07 | -9.91e-04 |
| BRYBND | 436 | 6.84e+07 | 1.61e-01 | 380 | 7.56e+07 | 4.50e-02 | 452 | 8.00e+07 | 1.85e-01 |

Tables 3 and 4 show that there is no big difference on the number of the function evaluations $\mathcal{N}$ and the time evaluation scale TFE. Therefore, we focus on "f. val.". The setting $N_u = N$ is not so good in the 4 problems HAIRY and MANCINO in Table 3, and ROSENBR, MANCINO in Table 4. This is because $N$ samples is too few to construct the objective function well for the ill-conditioned problems. Similarly, the setting $N_u = 3N$ is not so good in 5 problems. This is because there exist many samples outside of the trust region so that they negatively affect the quality of the model function in the trust region. On the other hand, the setting $N_u = 2N$ never greatly compares unfavorably in all problems compared with others. This numerical experiment recommends that the upper bound $N_u$ should be set to $2N$.

**Experiment on the control $\epsilon_k$** Next, it experimented concerning the adjusting of accuracy $\epsilon_k$ of the function evaluation in each iteration. The termination criterion was set to $\Delta_k < 10^{-3}$. The upper bound of the number of the sample was set to $N_u = 2N$. We examined in the following 3 cases as adjusting methods satisfying $\epsilon_k = o(\Delta_k)$:

**(Update 1)** $\epsilon_k = \min(\Delta_k^2,\ 0.1)$

**(Update 2)** $\epsilon_k = \min(0.5\Delta_k^2,\ 0.1)$

**(Update 3)** $\epsilon_k = \min(\Delta_k^{1.1},\ 0.1)$

Tables 5 and 6 report the results for the uniformly random number and the Gaussian random number respectively.

Table 5: Numerical results for various updating of $\epsilon_k$ with uniform random error

| | $\epsilon_k = \min(\Delta_k^2, 0.1)$ | | | $\epsilon_k = \min(0.5\Delta_k^2, 0.1)$ | | | $\epsilon_k = \min(\Delta_k^{1.1}, 0.1)$ | | |
|---|---|---|---|---|---|---|---|---|---|
| PROBLEM | $\mathcal{N}$ | TFE | f. val. | $\mathcal{N}$ | TFE | f. val. | $\mathcal{N}$ | TFE | f. val. |
| SISSER | 44 | 1.85e+06 | 2.24e-05 | 50 | 5.40e+06 | 7.60e-06 | 31 | 6.84e+03 | 1.96e-03 |
| CLIFF | 36 | 1.80e+06 | 2.01e-01 | 35 | 5.10e+06 | 2.01e-01 | 32 | 5.76e+03 | 2.00e-01 |
| ROSENBR | 250 | 2.35e+07 | 9.97e-01 | 204 | 1.79e+07 | 6.17e-01 | 136 | 3.29e+04 | 2.31e+00 |
| HAIRY | 83 | 4.02e+06 | 2.00e+01 | 120 | 1.02e+07 | 2.00e+01 | 85 | 1.51e+04 | 2.00e+01 |
| GROWTHLS | 52 | 2.63e+06 | 1.52e+03 | 54 | 4.56e+06 | 2.00e+03 | 79 | 1.91e+04 | 1.74e+03 |
| GULF | 66 | 7.42e+06 | 6.61e+00 | 56 | 8.13e+06 | 6.59e+00 | 40 | 6.90e+03 | 6.65e+00 |
| PFIT1LS | 65 | 4.69e+06 | 1.48e+00 | 51 | 1.05e+07 | 3.03e-01 | 43 | 7.82e+03 | 5.25e-01 |
| BROWNDEN | 492 | 3.27e+06 | 8.93e+04 | 454 | 5.84e+06 | 8.64e+04 | 586 | 3.99e+04 | 8.58e+04 |
| HART6 | 151 | 5.65e+06 | -3.32e+00 | 121 | 8.79e+06 | -3.32e+00 | 124 | 1.91e+04 | -3.06e+00 |
| MANCINO | 240 | 2.25e+07 | 1.96e-01 | 206 | 2.42e+07 | 1.54e-01 | 239 | 6.77e+04 | 1.36e-01 |
| POWER | 423 | 2.31e+07 | 1.01e-04 | 405 | 5.35e+07 | 2.78e-05 | 231 | 3.65e+04 | 4.46e-02 |
| MOREBV | 350 | 2.77e+07 | 4.97e-03 | 259 | 3.63e+07 | 5.82e-03 | 79 | 1.33e+04 | 1.20e-02 |
| BRYBND | 432 | 3.10e+07 | 1.46e-01 | 405 | 7.38e+07 | 1.18e-01 | 441 | 1.31e+05 | 7.35e-02 |

Table 6: Numerical results for various updating of $\epsilon_k$ with normal distribution error

| | $\epsilon_k = \min(\Delta_k^2, 0.1)$ | | | $\epsilon_k = \min(0.5\Delta_k^2, 0.1)$ | | | $\epsilon_k = \min(\Delta_k^{1.1}, 0.1)$ | | |
|---|---|---|---|---|---|---|---|---|---|
| PROBLEM | $\mathcal{N}$ | TFE | f. val. | $\mathcal{N}$ | TFE | f. val. | $\mathcal{N}$ | TFE | f. val. |
| SISSER | 54 | 2.62e+06 | -2.49e-02 | 50 | 3.62e+06 | -7.37e-05 | 33 | 7.29e+03 | -2.43e-03 |
| CLIFF | 34 | 1.67e+06 | 2.01e-01 | 32 | 2.75e+06 | 2.01e-01 | 36 | 6.59e+03 | 1.99e-01 |
| ROSENBR | 228 | 1.57e+07 | 6.25e-01 | 229 | 2.94e+07 | 1.29e-01 | 172 | 3.52e+04 | 1.18e+00 |
| HAIRY | 80 | 2.63e+06 | 2.00e+01 | 73 | 8.10e+06 | 2.00e+01 | 100 | 2.18e+04 | 2.34e+01 |
| GROWTHLS | 44 | 3.00e+06 | 2.09e+03 | 53 | 7.79e+06 | 2.11e+03 | 68 | 1.90e+04 | 1.83e+03 |
| GULF | 64 | 8.71e+06 | 6.58e+00 | 68 | 1.24e+07 | 6.57e+00 | 38 | 9.65e+03 | 6.61e+00 |
| PFIT1LS | 43 | 2.35e+06 | 1.04e+00 | 47 | 6.07e+06 | 3.93e-01 | 47 | 1.22e+04 | 9.50e+01 |
| BROWNDEN | 543 | 3.16e+06 | 8.61e+04 | 477 | 7.53e+06 | 8.65e+04 | 647 | 5.76e+04 | 8.58e+04 |
| HART6 | 133 | 5.49e+06 | -3.32e+00 | 124 | 9.15e+06 | -3.31e+00 | 121 | 1.66e+04 | -3.30e+00 |
| MANCINO | 228 | 2.07e+07 | 4.18e-01 | 231 | 3.06e+07 | 1.96e-01 | 230 | 6.64e+04 | 2.70e-01 |
| POWER | 360 | 1.54e+07 | 5.64e-04 | 372 | 3.91e+07 | 4.35e-04 | 236 | 2.83e+04 | 4.56e-02 |
| MOREBV | 317 | 2.87e+07 | 3.49e-04 | 308 | 4.71e+07 | -4.74e-03 | 89 | 1.51e+04 | -1.22e-03 |
| BRYBND | 402 | 2.35e+07 | 1.53e-01 | 426 | 6.11e+07 | 1.22e-01 | 485 | 1.62e+05 | 6.29e-02 |

From the tables, we see that Update 2 is better than Update 1 in the accuracy of the optimal value, while there is not much difference between Update 1 and Update 2 in TFE. Update 3 is worse than the others in the accuracy of the optimal value, while it is much faster than the others from the view point of TFE. If the computation time is valued more than the accuracy of the optimal value, it might be good to use Update 3. In contrast, if the accuracy of the optimal value is important, it might be good to use Update 2.

### 4.1.3 Experiment on the comparison with constant accuracy

We examined the algorithmic performance by Update 2, comparing with the case in which the function evaluation accuracy $\epsilon_k$ was fixed to a small value, that is, $\epsilon_k = 10^{-6}$ for all $k$. Note that this fixed case corresponds to using existing derivative-free trust-region algorithms.

In the following, we set $g$ as $g(\epsilon) = 10^{-6}/\epsilon$ in order to set $g(10^{-1}) = 1$ for the fixed high accuracy. The termination criterion was set to $\Delta_k < 10^{-3}$. The upper bound of the number of the sample was set to $N_u = 2N$. We give the result in Table 7

Table 7 shows that the proposed algorithm with updating $\epsilon_k$ takes smaller time to obtain the same accuracy solution of the fixed version. In addition, TFE of the update version is 2 to 5 times of the number of the problem's variables for some problems such as SISSER, CLIFF, HAIRY, HART6 and BRYBND. For instance, in the problem CLIFF, TFE = $5.96 \approx 3n = 6$. This indicates that the algorithm can obtain the solution of the problem by the time required for only 6 times evaluations of the objective function with the accuracy $10^{-6}$.

Table 7: Numerical results for updating $\epsilon_k$ vs. fixed $\epsilon_k = 10^{-6}$ with uniform random error

|  | # | $\epsilon_k$ =1e-06 |  |  | $\epsilon_k = 0.5\Delta_k^2$ |  |  |
|---|---|---|---|---|---|---|---|
| PROBLEM | var. | $\mathcal{N}$ | TFE | f. val. | $\mathcal{N}$ | TFE | f. val. |
| SISSER | 2 | 31 | 31.7 | -5.69e-07 | 56 | 10.1 | 3.54e-06 |
| CLIFF | 2 | 30 | 30.9 | 2.01e-01 | 34 | 5.9 | 2.01e-01 |
| ROSENBR | 2 | 341 | 341.0 | 1.11e-01 | 273 | 37.8 | 3.24e-02 |
| HAIRY | 2 | 58 | 58.8 | 2.00e+01 | 155 | 7.5 | 2.00e+01 |
| GROWTHLS | 3 | 45 | 45.1 | 1.77e+03 | 41 | 5.8 | 1.41e+03 |
| GULF | 3 | 60 | 60.0 | 6.45e+00 | 306 | 14.7 | 5.70e+00 |
| PFIT1LS | 3 | 44 | 44.7 | 6.58e-01 | 59 | 7.5 | 2.79e-01 |
| BROWNDEN | 4 | 227 | 226.9 | 8.59e+04 | 461 | 13.3 | 8.61e+04 |
| HART6 | 6 | 161 | 161.1 | -3.32e+00 | 134 | 14.1 | -3.32e+00 |
| MANCINO | 10 | 220 | 220.4 | 9.67e-02 | 188 | 18.5 | 1.12e-01 |
| POWER | 10 | 329 | 329.4 | 2.60e-06 | 420 | 79.9 | 6.12e-05 |
| MOREBV | 10 | 265 | 264.8 | 6.18e-03 | 298 | 64.2 | 3.36e-03 |
| BRYBND | 10 | 463 | 463.2 | 1.49e-01 | 262 | 28.3 | 1.09e-02 |

## 4.2 Numerical Results for Implied Valatility Estimation

In this subsection, we consider options traded in financial markets [12]. A *put* (*call*) *option* is the contract right to sell (buy) a specified amount of asset (real or financial) at a fixed price on or before a fixed date. The price of the option is called the option *premium*. The date on which the contract expires is called the *maturity*. We aim to estimate the volatility of an asset and the risk-free rate from the option price.

In option trading, it is important to estimate the volatility from the premium for measurement of the validity of the premium. We formulate the estimation problem as the unconstrained minimization problem (1.1). Let $c : \mathbb{R}^2 \to \mathbb{R}$ and $p : \mathbb{R}^2 \to \mathbb{R}$ denote the call and put option premium of the volatility $\sigma$ and the risk-free $r$ rate, respectivery. Suppose that the call option premium $c^*$ and the put option premium $p^*$ are given. Then, the problem that estimates the volatility and the risk-free rate from the premium can be formulate as

$$\underset{\sigma\in\mathbb{R}, r\in\mathbb{R}}{\text{minimize}} \quad f(\sigma, r) := \{c(\sigma, r) - c^*\}^2 + \{p(\sigma, r) - p^*\}^2 ,$$

which is the least squared problem (1.1).

In numerical experiments, we consider the *look-back* option and the *Asian* option. A look-back option is a path dependent option where the option owner has the right to buy (sell) the underlying instrument at its lowest (highest) price over some preceding period. The Asian option is an option where the payoff is not determined by the underlying price at maturity but by the average underlying price over some pre-set period of time. However, it is impossible to calculate the premiums of the look-back option and the Asian option in closed-form. In general, these premiums $c(\sigma, r),\ p(\sigma, r)$ are often evaluated via a Monte Carlo simulation. The Monte Carlo simulation determines the option premium for a set of randomly generated economic scenarios [23]. The resulting sample set yields an expectation value for the option. In Appendix B.1, we give the detail of the evaluation method of these option premiums via the Monte Carlo simulation.

We consider a certain asset whose price process follows the geometric Brownian motion (B.1). We suppose that the current price $S(0)$ of the asset is 62 and the period $T$ is 1/12.

**Problem 1** For the underlying asset, the strike price $K_c$ and the premium $c_B^*$ of the look-back type maximum call option are given as 60 and 4.7085 respectively. Similarly, the strike price $K_p$ and the premium $p_B^*$ of the look-back type minimum put option are given as 64 and 4.1276 respectively. Determine the volatility $\sigma$ and risk-free rate $r$ of the asset from the premiums. (The solution $(r, \sigma)$ of this problem is $(0.1, 0.2)$.)

**Problelm 2** For the underlying asset, the strike price $K_c$ and the premium $c_B^*$ of the Asian type call option are given as 60 and 2.3710 respectively. Similarly, the strike price $K_p$ and the premium $p_B^*$ of the Asian type put option are given as 64 and 1.9602 respectively. Determine the volatility $\sigma$ and risk-free rate $r$ of the asset from the premiums. (The solution $(r, \sigma)$ of this problem is $(0.1, 0.2)$.)

**Numerical results for Problems 1 and 2** We update the accuracy by $\epsilon_k = 0.5\Delta_k^2$ and set the accuracy of the option premium as (B.9). The upper bound of the number of the sample is set to $N_u = 2N$. The termination criterion is given as $\Delta_k < 10^{-3}$. Tables 8 and 9 report the numerical results for Problems 1 and 2, respectively. In these tables, "NQ" denotes the total number of the economic scenarios generated by the Monte Carlo simulation.

Table 8: Numerical result of problem 1.

| algorithm | $r$ | $\|\sigma\|$ | $\mathcal{N}$ | Time [s] | NQ [times] | f. val. |
|---|---|---|---|---|---|---|
| proposed ($\epsilon_k = 0.5\Delta_k^2$) | 0.0980 | 0.1999 | 40 | 3.99e+03 | 4.07e+08 | 6.20e-05 |
| fixed ($\epsilon_k = 10^{-6}$) | 0.1005 | 0.2004 | 28 | 6.24e+04 | 6.00e+09 | 5.58e-05 |

Table 8 shows that the proposed algorithm can solve the problem almost 15 times faster than the fixed case. For Problem 1, the trial number of the Monte Carlo simulation required to evaluate an objective value with the high accuracy: $1.00 \times 10^{-6}$, is $(6.00 \times 10^9)/28 = 2.14 \times 10^8$. Note that NQ of the proposed algorithm is $4.07 \times 10^8$, and $(4.07 \times 10^8)/(2.14 \times 10^8) \approx 1.9$. This indicates that the proposed algorithm find a solution in the time required for 1.9 times evaluation of an objective value with the high accuracy $1.00 \times 10^{-6}$.

Table 9: Numerical result of problem 2.

| algorithm | $r$ | $\|\sigma\|$ | $\mathcal{N}$ | Time [s] | NQ [times] | f. val. |
|---|---|---|---|---|---|---|
| proposed ($\epsilon_k = 0.5\Delta_k^2$) | 0.1017 | 0.1968 | 40 | 2.41e+03 | 2.61e+08 | 2.02e-04 |
| fixed ($\epsilon_k = 10^{-6}$) | 0.0982 | 0.1983 | 32 | 4.00e+04 | 4.35e+09 | 5.20e-05 |

Table 9 shows that the proposed algorithm can solve the problem almost 16 times faster than the fixed case. For Problem 2, the trial number of the Monte Carlo simulation required to evaluate an objective value with the high accuracy: $1.00 \times 10^{-6}$, is $(4.35 \times 10^9)/32 = 1.36 \times 10^8$. Note that NQ of the proposed algorithm is $2.61 \times 10^8$, and $(2.61 \times 10^8)/(1.36 \times 10^8) \approx 2$. This indicates that the proposed algorithm find a solution in the time required for 2 times evaluation of an objective value with the high accuracy $1.00 \times 10^{-6}$.

## 5 Concluding Remarks

In this paper, we proposed the derivative-free trust-region algorithm for the problems with controllable error in the objective function evaluation. To this end, we developed the method to construct the model function by applying a SVR, and the control method of the accuracy. In Section 4, we have applied the proposed algorithm to the several problems taken from CUTEr collection and the financial problems estimating the implied volatility. The numerical experiments showed that the proposed algorithm could find a reasonable solution much faster than the algorithm, in which the accuracy of the function evaluation was maintained in high. In particular, for relatively easy small-scale problems, the proposed algorithm can obtain optimal solutions with computational costs which amount to those required for several times evaluations of an accurate function value.

This work has focused on development of the practical algorithm. It is certainly important to describe a class of algorithms which has the global convergence property for the problem with controllable error. We hope that this work contributes to future development of efficient algorithms and establishment of the global convergence for the problem we consider.

**Acknowledgements**

The author wishes to express his sincerest thanks and appreciation to Associate Professor Nobuo Yamashita for his kind guidance and direction in this study, invaluable discussions, constructive criticisms in the writing of the manuscripts, extreme patience, and encouragement throughout the course of this work. The author wishes to tender his acknowledgments to Professor Masao Fukushima for his constructive comments and kind guidance. The author also wishes to express his thanks to Assistant Professor Shunsuke Hayashi for his invaluable advice. The author greatly appreciates the help of all members of System Optimization Laboratory. Finally, but not the least, precious thanks are due to his family, for their strong support and an ectionate encouragement throughout the course of this study.

## References

[1] Bongartz, I., Conn, A. R., Gould, N. I. M. and Toint, Ph. L., CUTE: Constrained and unconstrained testing environment, *ACM Transactions on Mathematical Software*, 21 pp. 123–160, 1995.

[2] Booker, A. J., Dennis, J. E., Frank, P. D., Serafini, D. B., Torczon, V., Trosset, M. W., A rigorous framework for optimization of expensive functions by surrogates, *Structural and Multidisciplinary Optimization*, 17 pp. 1–13, 1999.

[3] Colson, B, Marcotte, P and Savard, G., Bilevel programming: A survey, *4OR: A Quarterly Journal of Operations Research*, 3 pp. 87–107, 2005.

[4] Conn, A. R., Gould, N. I. M., Toint, Ph. L., *Trust-Region Methods*, SIAM, Philadelphia, 2000.

[5] Conn, A. R., Toint, Ph. L., An algorithm using quadratic interpolation for unconstrained derivative free optimization, *Nonlinear Optimization and Applications*, Plenum Publishing, pp. 27–47, New York, 1996.

[6] Conn, A. R., Scheinberg, K., and Toint, Ph. L., A derivative free optimization algorithm in practice, *the American Institute of Aeronautics and Astronautics Conference*, St Louis, 1998.

[7] Conn, A. R., Scheinberg, K., and Toint, Ph. L., A derivative free optimization method via support vector machines, 1999. `www.cas.mcmaster.ca/~oplab/seminar/conn/conn_deriv_free.ps`

[8] Conn, A. R., Scheinberg, K., and Toint, Ph. L., On the convergence of derivative-free methods for unconstrained optimization. *Approximation Theory and Optimization*, pp. 83–108, Cambridge University Press, Cambridge, UK, 1997.

[9] Conn, A. R., Scheinberg, K., and Vicente, L. N., Geometry of interpolation sets in derivative free optimization, *Mathematical Programming*, 111, pp. 141–172, 2008.

[10] Conn, A. R., Scheinberg, K., and Vicente, L. N., Geometry of sample sets in derivative free optimization. part II: Polynomial regression and underdetermined interpolation, *Preprint* 05-15, Department of Mathematics, University of Coimbra, Portugal, 2005.

[11] Conn, A. R., Scheinberg, K., and Vicente, L. N., Global convergence of general derivative-free trust-region algorithms to first and second order critical points, *Preprint* 06-49, Department of Mathematics, University of Coimbra, 2006.

[12] Cox, C. and Rubinstein, M., *Option Markets*, Prentice-Hall, 1985.

[13] Cristianini, N. and Shawe-Taylor, J., *An introduction to support vector machines and other kernel-based methods*, Cambridge University Press, Cambridge, UK, 2000.

[14] Dennis, J. E. and Torczon, V., Direct search methods on parallel machines, *SIAM Journal on Optimization*, 1 pp. 448–474, 1991.

[15] Fisher, R. A., *The Design of Experiments*, Oliver and Boyd Ltd., 1951.

[16] Karasözen, B., Survey of trust-region derivative free optimization methods, *Journal of Industrial and Management Optimization*, 3, pp. 321–334, 2007.

[17] Kolda, T. G., Lewis, R. M., Torzcon, V., Optimization by direct search: new perspectives of some classical and modern methods, *SIAM Review*, 45 pp. 385–482, 2003.

[18] Nelder, J. A. and Mead, R., A simplex method for function minimization, *Computer Journal*, 7 pp. 308–313, 1965.

[19] Nocedal, J. and Wright, S. J., *Numerical Optimization*, Springer-Verlag, New York, USA, 1999.

[20] Powell, M. J. D., A direct search optimization method that models the objective and constraint functions by linear interpolation, *Advances in optimization and numerical analysis*, Kluwer Academic Publishers, 1994.

[21] Powell, M. J. D., Trust region methods that employ quadratic interpolation to the objective function, *Presentation at the 5th SIAM Conference on Optimization*, 1996.

[22] Powell, M. J. D., UOBYQA: unconstrained optimization by quadratic approximation, *Mathematical Programming*, 92 pp. 555–582, 2002.

[23] Tilley, J. A., Valuing American options in a path simulation model, *Transactions of the Society of Actuaries*, 45, pp. 83–104, 1993.

[24] Torczon, V., On the convergence of the multidirectional search algorithm, *SIAM Journal on Optimization*, 1 pp. 123–145, 1991.

[25] Winfield, D., Function and functional optimization by interpolation in data tables, *PhD thesis*, Harvard University, Cambridge, USA, 1969.

[26] Winfield, D., Functional minimization by interpolation in a data table, *Journal of the Institute of Mathematics and its Applications*, 12 pp. 339–347, 1973.

## A Error bound for the gradient

We give an error bound for the true derivative as follows.

**Theorem A.1** *Let $\boldsymbol{x}_1, \dots, \boldsymbol{x}_n \in \mathbb{R}^n$ be belong to the ball $\mathcal{B} = \{\boldsymbol{x} \in \mathbb{R}^n \mid \|\boldsymbol{x} - \boldsymbol{x}_0\| \leq \Delta\}$ with given $\boldsymbol{x}_0 \in \mathbb{R}^n$ and $\Delta > 0$. Suppose that the following assumptions (i)–(iv) hold.*

*(i) The objective function $f$ is continuously differentiable and $\nabla f$ is Lipschitz continuous with Lipschitz constant $L_f$.*

*(ii) There exist function $m$ and positive constant $\Lambda$ such that*

$$|f(\boldsymbol{x}) - m_k(\boldsymbol{x})| \leq \Lambda(\Delta_k^2 + \epsilon), \ ^\forall \boldsymbol{x} \in \mathcal{B}.$$

*(iii) The gradient of the function $\nabla m$ is Lipschitz continuous with Lipschitz constant $L_m$.*

*(iv) $\boldsymbol{x}_{k^*}$ is a stationary point of the function $m$.*

$$\nabla m(\boldsymbol{x}_0) = \mathbf{0}.$$

*$\epsilon$ denotes the largest function evaluation accuracies corresponding to the sample points $\boldsymbol{x}_i, \ i = 1, \dots, n$ within the trust region.*

*Then, the following relation hold.*

$$\|\nabla f(\boldsymbol{x}_0)\| \leq \sqrt{n}(2L_f + 2L_m + \Lambda)\|\hat{\boldsymbol{D}}^{-1}\|\Delta_k + \sqrt{n}(3 + \Lambda)\|\hat{\boldsymbol{D}}^{-1}\|\epsilon/\Delta_k$$

*where $\hat{\boldsymbol{D}}$ is a matrix defined as*

$$\hat{\boldsymbol{D}} := \begin{pmatrix} (\boldsymbol{x}_1 - \boldsymbol{x}_0)^\top/\Delta_k \\ \vdots \\ (\boldsymbol{x}_n - \boldsymbol{x}_0)^\top/\Delta_k \end{pmatrix} \in \mathbb{R}^{n \times n}.$$

**Proof.** From the mean-value theorem, we have for all $i = 1, \dots, n$

$$\begin{aligned} |(\boldsymbol{x}_i - \boldsymbol{x}_0)^\top \nabla f(\boldsymbol{x}_0)| &= \left| f(\boldsymbol{x}_i) - f(\boldsymbol{x}_0) - \int_0^1 \Big(\nabla f\big(\boldsymbol{x}_0 + t\,(\boldsymbol{x}_i - \boldsymbol{x}_0)\big) - \nabla f(\boldsymbol{x}_0)\Big)^\top (\boldsymbol{x}_i - \boldsymbol{x}_0) dt \right| \\ &\leq |f(\boldsymbol{x}_i) - f(\boldsymbol{x}_0)| + \frac{1}{2} L_f \|\boldsymbol{x}_i - \boldsymbol{x}_0\|^2. \end{aligned}$$

Since $\boldsymbol{x}_i, \boldsymbol{x}_0 \in \mathcal{B}$, it then follows that

$$\begin{aligned} |(\boldsymbol{x}_i - \boldsymbol{x}_0)^\top \nabla f(\boldsymbol{x}_0)| &\leq |f(\boldsymbol{x}_i) - f(\boldsymbol{x}_0)| + 2L_f\Delta_k^2 \\ &\leq |f_{\epsilon_i}(\boldsymbol{x}_i) - f_{\epsilon_k}(\boldsymbol{x}_0)| + 2L_f\Delta_k^2 + 2\epsilon. \end{aligned}$$

From Assumption (ii) and (iv), we have that

$$\begin{aligned} |(\boldsymbol{x}_i - \boldsymbol{x}_0)^\top \nabla f(\boldsymbol{x}_0)| &\leq |m_k(\boldsymbol{x}_i) - m_k(\boldsymbol{x}_0)| + (2L_f + \Lambda)\Delta_k^2 + (3 + \Lambda)\epsilon \\ &\leq \left| \nabla m_k(\boldsymbol{x}_0)^\top (\boldsymbol{x}_i - \boldsymbol{x}_0) + \int_0^1 \Big(\nabla m_k\big(\boldsymbol{x}_0 + t\,(\boldsymbol{x}_i - \boldsymbol{x}_0)\big) - \nabla m_k(\boldsymbol{x}_0)\Big)^\top (\boldsymbol{x}_i - \boldsymbol{x}_0) dt \right| \\ &\qquad + (2L_f + \Lambda)\Delta_k^2 + (3 + \Lambda)\epsilon \\ &\leq (2L_f + 2L_m + \Lambda)\Delta_k^2 + (3 + \Lambda)\epsilon. \end{aligned}$$

Then, it then follows from easy calculation on the properties of the matrix norms that

$$\begin{aligned}\|\hat{\boldsymbol{D}}\nabla f(\boldsymbol{x}_0)\|_2 &\le \sqrt{n}\|\hat{\boldsymbol{D}}\nabla f(\boldsymbol{x}_0)\|_\infty \\ &\le \sqrt{n}(2L_f + 2L_m + \Lambda)\Delta_k + \sqrt{n}(3+\Lambda)\epsilon/\Delta_k,\end{aligned}$$

and hence

$$\|\nabla f(\boldsymbol{x}_0)\| \le \sqrt{n}(2L_f + 2L_m + \Lambda)\|\hat{\boldsymbol{D}}^{-1}\|\Delta_k + \sqrt{n}(3+\Lambda)\|\hat{\boldsymbol{D}}^{-1}\|\epsilon/\Delta_k,$$

which is the desired inequarity. □

## B Estimation of option premium and its estimation error

### B.1 Estimation of the Option Premium via Monte Carlo Simulation

In finance, the price process of a certain financial asset such as a stock is represented by the stochastic process. For instance, in the black-scholes model, which is famous in the option pricing theory, the price process $S(t)$ is supposed to follow the following Geometric Brownian Motion:

$$dS(t) = rS(t)dt + \sigma S(t)dw(t), \tag{B.1}$$

where $r$ is the risk-free rate, $\sigma$ is the implied volatility and $dw(t)$ is the Brownian motion. A Monte Carlo simulation determines the option premium for a set of randomly generated economic scenarios following this stochastic process [23]. The resulting sample set yields an expectation value for the option premium. To evaluate a path dependent option premium, it is necessary to generate data of the underlying price for the period.

The price process is represented by the stochastic process (B.1), however, it is impossible to generate continuous data on the computer. Then, we consider the discrete-time approximation of this process. Let the current time be 0 and the maturity be $T$. We approximate the process with $\nu$ capitation of the time interval $[0, T]$ as follows:

$$\begin{aligned} dt &\cong \Delta t = T/\nu, \\ dS(t) &\cong \Delta S_t = S_{t+\Delta t} - S_t, \\ dz(t) &\cong w\sqrt{\Delta t}, \\ w &\sim N(0,1). \end{aligned} \tag{B.2}$$

Then, the continuous time model (B.1) can be written as

$$\Delta S_t = \nu S_t \Delta t + \sigma S_t w \sqrt{\Delta t}$$

so that using (B.2) we obtain the following recurrence relations:

$$S_{t+1} = (1 + \nu\Delta t)S_t + \sigma S_t w\sqrt{\Delta t}. \tag{B.3}$$

The approximation accuracy of the expression (B.3) to (B.1) becomes higher as $\nu$ is increasing. However, the cost of the simulation is also increasing as do so. A Monte Carlo simulation generates the data sequence of the underlying prices $\{S_0, \dots, S_\nu\}$ according to the recurrence relation (B.3) and calculate the option premium from the sequence.

For instance, because the strike price of Asian option is based on the average of the price process in the period, the gain of Asian-type call option for the sequence $\{S_0, \ldots, S_\nu\}$ is calculated as

$$c_A(\nu, \sigma) = \max \left\lfloor \frac{1}{\nu+1} \sum_{t=0}^{\nu} \{S_t\} - K, 0 \right\rfloor .$$

Similarly, the gain of Asian-type put option is calculated as

$$p_A(\nu, \sigma) = \max \left\lfloor K - \frac{1}{\nu+1} \sum_{t=0}^{\nu} \{S_t\}, 0 \right\rfloor .$$

In addition, the gains of look-back type call and put options are calculated as

$$c_L(\nu, \sigma) = \max \left\lfloor \max_{0 \le t \le \nu} \{S_t\} - K, 0 \right\rfloor$$

and

$$p_L(\nu, \sigma) = \max \left\lfloor K - \min_{0 \le t \le \nu} \{S_t\}, 0 \right\rfloor ,$$

respectively.

The Monte Carlo simulation determines the expected value of the option premium by repeating the operations described above. The error of the evaluated option premium is dependent on the number of the trial number $q$ of the Monte Carlo simulation. When the accuracy of the option premium $\epsilon_o$ is given, we set the trial number $q$ as follows.

The error $e_o$ of the evaluated value by the Monte Carlo simulation follows the Gaussian distribution $N(0, s)$, and trial number $q$ and the standard deviation $s$ have the relation: $s = O(1/\sqrt{q})$ [23]. From this fact, when we set $s = \epsilon_o/1.96$, the probability that the error $e_o$ falls in the interval $[-\epsilon_o, \epsilon_o]$ becomes 95 %. Now, we suppose that the standard deviation $s$ can be represented as $s = a/\sqrt{q+b}$ with some constants $a, b \in \mathbb{R}$. Then, the trial number $q$ is written as $q = a^2/s^2 - b$. Hence, when the accuracy $\epsilon_o$ of the option premium is given, we set the trial number $q$ as follows:

$$q = \frac{(1.96a)^2}{\epsilon_o^2} - b. \tag{B.4}$$

The constants $a$ and $b$ are estimated by pri-ori numerical experiments. For examples, $a, b$ for Problem 1 and 2 in Section 4.2 are estimated as Table 10.

Table 10: Setting for $a$ and $b$

| OPTION | $a$ | $b$ |
|---|---|---|
| Look back option | 159.7067 | 2.3560 |
| Asian option | 50.5936 | 1.7687 |

## B.2 Relation between the accuracy and the trial number of Monte Carlo simulation

We explain how to set the accuracy $\epsilon_o$ of the option premium obtained by a Monte Carlo simulation for given accuracy $\epsilon$ of the objective function evaluation.

In numerical experience of section 4.2, we consider the following problem which estimates implied volatility $\sigma$ and the risk-free rate $r$ from given option premiums.

$$f(\sigma, r) := \{c(\sigma, r) - c^*\}^2 + \{p(\sigma, r) - p^*\}^2,$$

Now, let us denote the true call abd put option premiums with $(\sigma, r)$ by $p$ and $c$, respectively, and denote the errors of the evaluated option premiums $c(\sigma, r)$, $p(\sigma, r)$ by $e_c$, $e_p$ respectively. Then, we have that

$$f_\epsilon(\sigma, r) = (p(\sigma, r) + e_p - p^*)^2 + (c(\sigma, r) + e_c - c^*)^2 = (p - p^*)^2 + (c - c^*)^2 + e_p^2 + e_c^2 + e_p(p - p^*) + e_c(c - c^*)$$
$$f(\sigma, r) + e_p^2 + e_c^2 + e_p(p - p^*) + e_c(c - c^*).$$

Thus, the error $e$ of the evaluated objective value is

$$e = e_p^2 + e_c^2 + e_p(p - p^*) + e_c(c - c^*).$$

In order to satisfy the relation $\epsilon \geq |e|$, for given accuracy $\epsilon$, it is sufficient that

$$\epsilon/2 \geq e_p^2 + e_p|p - p^*|, \tag{B.5}$$
$$\epsilon/2 \geq e_c^2 + e_c|c - c^*|, \tag{B.6}$$

hold. Let $\epsilon_p$ and $\epsilon_c$ denote the maximum values of $e_p$ and $e_c$ satisfying the inequalities (B.5) and (B.6), respectively. Then, they are

$$\epsilon_p = \frac{-|p - p^*| + \sqrt{|p - p^*|^2 + 2\epsilon}}{2}, \tag{B.7}$$
$$\epsilon_c = \frac{-|c - c^*| + \sqrt{|c - c^*|^2 + 2\epsilon}}{2}. \tag{B.8}$$

The objective value can be evaluated with the accuracy $\epsilon$ by setting $\epsilon_p$ and $\epsilon_c$ given by (B.7) and (B.8), respectively. However, if $|c - c^*|$ or $|p - p^*|$ is large, the accuracies $\epsilon_p$ and $\epsilon_c$ of the option premiums becomes much smaller than $\epsilon$. Then, the computation time of a Monte Carlo simulation becomes too large to perform. In that case, in view of that $|c - c^*| \approx 0$, $|p - p^*| \approx 0$ in the neighborhood of a solution, it would be reasonable to set $\epsilon_p$ and $\epsilon_c$ as

$$\epsilon_p = \epsilon_c = \sqrt{\frac{\epsilon}{2}} \tag{B.9}$$