SURVEY OF DERIVATIVE-FREE OPTIMIZATION

MIN XI*
School of Mathematical Sciences, Jiangsu Key Laboratory for NSLSCS
Nanjing Normal University, Nanjing 210023, China
and
School of Mathematics and Statistics, Guangdong University of Foreign Studies
Guangzhou 510006, China

WENYU SUN
School of Mathematical Sciences, Jiangsu Key Laboratory for NSLSCS
Nanjing Normal University, Nanjing 210023, China

JUN CHEN
School of Information Engineering, Nanjing Xiaozhuang University
Nanjing 211171, China

Abstract. In this survey paper we present an overview of derivative-free optimization, including basic concepts, theories, derivative-free methods and some applications. To date, there are mainly three classes of derivative-free methods and we concentrate on two of them, they are direct search methods and model-based methods. In this paper, we first focus on unconstrained optimization problems and review some classical direct search methods and model-based methods in turn for these problems. Then, we survey a number of derivative-free approaches for problems with constraints, including an algorithm we proposed for spherical optimization recently.

1. Introduction. In this paper, we first consider the following unconstrained minimization problem:

\[ \min_{x \in \mathbb{R}^n} f(x), \]

where \( f : \mathbb{R}^n \to \mathbb{R} \) is a nonlinear function. When the accurate first derivative of \( f \) can be obtained, a wide selection of derivative-based methods is available to solve the problem (1). However, there exist many optimization problems where the derivatives of the objective functions are unavailable or unreliable. In this paper, we assume that the function value \( f(x) \) can be computed at any \( x \in \mathbb{R}^n \) but the information about the gradient of \( f \) is unavailable.

In many scientific and engineering applications, it is common to optimize a function obtained from an experiment or a complex computer simulation, and the value of the objective function is evaluated through some unavailable source code which makes the calculation of the derivatives impossible or impractical. Moreover, as

2020 Mathematics Subject Classification. Primary: 90C56; Secondary: 90C30.

Key words and phrases. Derivative-free optimization, direct search methods, model-based methods, unconstrained optimization.

This research was supported by the National Natural Science Foundation of China under grants 11571178.

* Corresponding author: Min Xi.
the scale and difficulty of the applications increased, the function evaluations become costly and noisy, we can not trust derivatives any more or approximate them by finite difference. Thus, we need to resort to the methods that deal with such problems without explicitly using derivatives. These methods are referred to as derivative-free methods and the area of nonlinear optimization that is concerned with the design and analysis derivative-free methods is referred to as derivative-free optimization (in brief, DFO).

Derivative-free optimization is an area has long history and its development dates back to the works of Hooke and Jeeves [27] with their “direct search” method and Winfield [66] with his model-based method. Derivative-free methods, especially the so called direct search methods, were popular and widely applied in the 1960s for they were relatively easy to implement. However, after dozen years of development, derivative-free methods fell out of favor with the mathematical optimization community for some of them converge very slowly or even can not guarantee global convergence. Meanwhile, some more sophisticated methods, like the quasi-Newton method, had matured and replaced a large extend of derivative-free methods. Nevertheless, in the last decades, due to the growth of applications, derivative-free optimization has experienced a renewed interest and has received considerable attention within the optimization community. The recent progress in this field includes establishing solid mathematical foundation for some of the methods [1, 5, 15, 17, 33, 40], developing effective derivative-free algorithms and softwares [18, 52, 53, 58] and offering the first textbook devoted to derivative-free methods [18].

In the literature, there are mainly three classes of derivative-free methods: direct search methods, model-based methods and implicit filtering methods. Implicit filtering methods [25] use the simplex gradients (a kind of generalization of finite-difference gradient) to approximate the actual gradients of the objective function and then perform a line search along the direction of the approximate gradient. In its simplest unconstrained form, implicit filtering method is the steepest descent algorithm with difference gradients. It has been shown that these methods are robust for problems with many local minima due to a noise. However, implicit filtering methods are essentially gradient-based approaches and we will not go into further details on these methods, and refer readers to [9, 32] for more details. In the rest of this paper, we focus on direct search methods and model-based methods which do not form explicit approximations of derivatives.

Direct search methods are zero-order methods that rely exclusively on values of the objective function and do not require derivative information. These methods sample the objective function at a set of points with certain predefined geometric pattern and decide what actions to take next solely based on those function values. More precisely, as Lewis et al. [39] pointed, direct search methods for unconstrained optimization depend on the relative rank of function values instead of numerical function values. There are many direct search methods, some classic and typical examples are Hooke and Jeeves’ pattern search, generalized pattern search (GPS), Nelder-Mead simplex method and mesh adaptive direct search (MADS). Direct search methods are classical and remain popular because they are easy to implement and many of them work well in practice. However, direct search methods do not fully exploit the available information from the objective function, and thus require a relatively large amount of function evaluations and some of these methods are very slow. We will discuss direct search methods at length in Section 2.
Another major class of derivative-free methods is model-based methods. At each iteration, these methods construct a model of the objective function, and then minimize the model function in a vicinity of the current iterate to find a new point for the next iteration. In short, these model-based methods approximate the objective function directly instead of its unavailable derivatives. Lacking the first derivatives, the model functions are established by interpolating $f$ at a set of appropriately chosen sample points. Also, model-based methods usually use a trust-region approach to minimize the model function because the model functions are reliable only in a small neighbourhood of the current iterate. Some of the better known model-based algorithms include DFO [18], NEWUOA [53] and BOBYQA [54]. Model-based methods try to exploit the underlying properties of the objective function rather than just use the function values alone. It has been shown by some research [46] that this class of methods is frequently superior in practical performance to the direct search methods. A detailed discussion on model-based methods will be presented in Section 3.

As mentioned before, this paper will first concentrate on unconstrained problems and overview two main classes of DFO methods: direct search methods and model-based methods, particularly the latter one. We discuss, in Section 2, the direct search methods and their global convergence. In Section 3, we describe the model-based methods and their global convergence. In Section 4, we discuss some derivative-free approaches for handling constrained problems, including an algorithm we proposed for spherical optimization recently. We describe two examples of applications of derivative-free optimization in Section 5 and then finish this paper by a short conclusion in Section 6.

1.1. **Notation.** Throughout this paper, $\| \cdot \|$ is used to denote the 2-norm for a vector and the induced 2-norm for a matrix. $\mathcal{B}$ denotes the closed ball in $\mathbb{R}^n$ and $\mathcal{B}(z, \Delta)$ denotes the closed ball in $\mathbb{R}^n$ of radius $\Delta > 0$ centered at $z$. By $P_d^n$, we denote the space of polynomials of degree less than or equal to $d$ in $\mathbb{R}^n$.

The natural basis $\bar{\phi}$ of $P_d^n$ is the following basis of normalized monomial functions

$$\bar{\phi} = \{1, x_1, x_2, \ldots, x_n, x_1^2/2, x_1x_2, \ldots, x_n^d/d!\}.$$

2. **Direct Search Methods.** Direct search methods appeared much earlier than the phrase “direct search” was first introduced by Hooke and Jeeves [29] in 1961. In fact, the coordinate search algorithm of Fermi and Metropolis was used on a digital computer [23] in 1952. Direct search methods are popular in the scientific and engineering communities because they are straightforward to implement and have been proved to be robust and reliable in practice. For instance, the Hooke and Jeeves’ pattern search and Nelder-Mead simplex methods are widely used from past till now. In the 1960s, direct search methods enjoyed attention in numerical optimization community and most basic direct search methods were proposed in those years. However, classical direct search methods were based on heuristics and their convergence analysis has not been established in the early period. By the early 1970s, numerical optimizers became less interested in heuristics and the direct search methods were dismissed. Until the early 1990s, Torczon [63, 64] gave the convergence analysis for generalized pattern search (GPS) methods and a renaissance of direct search methods started. Over the last decades, the interest in direct search methods has shifted to the theoretical analysis of this class of methods. Although a
wealth of new direct search methods have been proposed, most of them are variants of those basic approaches that already exist since 1971.

Nowadays, “direct search methods” is a catchall term for a large and disparate class of approaches that minimizing a function $f$ without explicitly using derivatives. Follow [18], we divide these methods into two classes: directional direct search and simplicial direct search. The former class of methods sample the objective function with the guide of sets of directions. These methods include coordinate search, pattern search, generalized pattern search (GPS), generating set search (GSS), mesh adaptive direct search (MADS), etc. Simplicial direct search methods are those based on simplices and operations over simplices to drive the search, include Spendley et al. simplex method [60], Nelder-Mead simplex method and some modern variants of Nelder-Mead simplex method. Take the limitation of the length into consideration, this paper will focus only on some classic direct search methods and give a brief sketch of them.

2.1. Coordinate Search. Coordinate search, also known as compass search or alternating variable search, is the ancestor of pattern search and is perhaps the simplest direct search method. Based on coordinate search, a large amount of variants have been developed, and by investigating this early version, we could get a perceptual understanding of pattern search methods.

Let $D^\oplus$ denote the set of $2n$ coordinate directions $D^\oplus = \{ e_1, e_2, ..., e_n, -e_1, -e_2, ..., -e_n \}$, where $e_i$ is the $i$th unit coordinate vector in $\mathbb{R}^n$. At each iteration, coordinate search algorithm evaluates the objective function at trail points of the form $x_k + \Delta_k d$ for each $d \in D^\oplus$, where $x_k \in \mathbb{R}^n$ is the $k$th iterate and $\Delta_k$ is the step-length control parameter. If there exists $d_k \in D^\oplus$ satisfying $f(x_k + \Delta_k d_k) < f(x_k)$, then move along the direction $d_k$ with step-length $\Delta_k$ and we say that the iteration is successful. The subsequence of successful iterations is denoted by $S$. Otherwise, the iteration is unsuccessful and the iterate remain unchanged while the value $\Delta_k$ has to be reduced. The subsequence of unsuccessful iterations is denoted by $U$. After any unsuccessful iteration, compare $\Delta_k$ with the preset tolerance $\Delta_{tol}$, once $\Delta_k$ falls below $\Delta_{tol}$, the algorithm terminates with $x^* = x_k$. We now describe the coordinate search algorithm formally as follows.

**Coordinate Search Algorithm**

**Initialization:**
- $x_0 \in \mathbb{R}^n$: initial guess point such that $f(x_0) < \infty$.
- $\Delta_{tol} > 0$: tolerance used for convergence.
- $\Delta_0 > \Delta_{tol}$: initial value of the step-length control parameter.

**Poll Step:** For $k = 0, 1, 2, ...$
- If there exists $d_k \in D^\oplus$, such that $f(x_k + \Delta_k d_k) < f(x_k)$, then set $x_{k+1} = x_k + \Delta_k d_k$, $\Delta_{k+1} = \Delta_k$.
- Otherwise, $f(x_k + \Delta_k d_k) \geq f(x_k)$ for all $d \in D^\oplus$, then $x_{k+1} = x_k$, $\Delta_{k+1} = \frac{1}{2}\Delta_k$.
- If $\Delta_k < \Delta_{tol}$, then terminate.

As we can see from above algorithm, coordinate search is really simple to describe and implement. Also, this method has been observed to be very reliable. However,
coordinate search may quickly approach a minimizer, but may be slow to detect this fact and thus sometimes is quite tardy to find the solutions.

2.2. Hooke and Jeeves’ Pattern Search (HJPS). In 1961, Hooke and Jeeves [29] provided a formal definition of direct search methods and also proposed the Hooke and Jeeves’ pattern search (HJPS). HJPS is a variant of coordinate search that accelerates the progress of the algorithm by exploiting information gained from previous successful iterations and it involves two types of moves: exploratory moves and pattern moves.

The exploratory move is a very local search to find an improving direction (not necessarily the best improving direction) in which to move. During the exploratory move, the current point is perturbed sequentially by small amounts in each of the coordinate directions and to observe whether improvement in the objective function value can be achieved.

Given the current point \( x_0 = (x_1^0, x_2^0, \ldots, x_n^0) \), the perturbation vector \( p_0 = (\Delta x^1, \Delta x^2, \ldots, \Delta x^n) \) which defines the sizes of perturbation steps and the perturbation tolerance vector \( \eta = (\eta^1, \eta^2, \ldots, \eta^n) \), an exploratory move around the point \( x_0 \) is performed as follows:

**step 1:** Let \( x_1 = x_0, \ p = p_0 \) and \( f_{\text{best}} = f(x_0) \).

**step 2:** For each component \( x^j \) in turn:

1. \( x_1 = x_1 + \Delta x^j e_j, \ e_j \) is the \( j \)-th unit coordinate vector.
2. If \( f(x_1) < f(x_0) \), \( f_{\text{best}} = f(x_1) \), go to step 2 for next component. Otherwise, \( x_1 = x_1 - \Delta x^j e_j \).
3. If \( f(x_1) < f(x_0) \), \( f_{\text{best}} = f(x_1) \), go to step 2 for next component. Otherwise \( x_1^1 = x_0^1 \), go to step 2 for next component.

**step 3:** If \( x_1 = x_0 \), then the exploratory move failed and the step size must be reduced: \( p = p/2 \). If \( p < \eta \), \( x_0 \) is the solution. Otherwise, go to step 2.

**step 4:** Otherwise, i.e. \( x_1 \neq x_0 \), the improving direction is \( x_1 - x_0 \) and go to Pattern Move.

As an improving direction \( x_1 - x_0 \) has been obtained, the pattern move performs a more aggressive step, that is, conducting exploratory move around the tentative iterate \( x_0 + 2(x_1 - x_0) \) then around the current iterate \( x_1 \). The idea is to move larger in the improving direction to see whether further progress is possible. If the exploratory move around \( x_0 + 2(x_1 - x_0) \) is successful, then the point returned by exploratory move is accepted as \( x_2 \). If not, i.e. the exploratory move failed, then the pattern move is deemed unsuccessful and the method reduces to exploratory move around \( x_1 \). Hooke and Jeeves [29] had pointed that the major reduction in \( f(x) \) is made by pattern moves.

As a special case of the generalized pattern search, HJPS is reliable to find the solutions. Furthermore, taking advantage of the pattern move, HJPS can rapidly approach the minimum and may be beneficial in situations where an optimum is far from the initial guess.

2.3. Generalized Pattern Search (GPS). Generalized pattern search (GPS) was proposed by Torczon [64] in 1997 to provide an abstract unifying framework for pattern search methods. All the existing pattern search algorithms, including the coordinate search, the Hooke and Jeeves’ pattern search [29] and the multidirectional search algorithm of Dennis and Torczon [20, 62] are instances of the abstract pattern search framework.
The common process of all pattern search methods is that they define the construction of a mesh and explore the mesh points according to some rules. If no decrease in objective function is obtained on mesh points around the current iterate, then the mesh is refined and the process is repeated until some stopping criterion is satisfied (typically, the mesh size).

Torczon [64] used a pattern \( P_k \) to describe the structure of the mesh which is explored in the \( k \)-th iteration. \( P_k \) is defined by a basis matrix and a generating matrix. The basis matrix can be any nonsingular matrix \( B \in \mathbb{R}^{n \times n} \). The generating matrix is a matrix \( C_k \in \mathbb{Z}^{n \times p} \), where \( p > 2n \). \( C_k \) can be partitioned into components

\[
C_k = [M_k - M_k L_k] = [\Gamma k L k],
\]

where \( M_k \in \mathbb{M} \subset \mathbb{Z}^{n \times n} \) (\( \mathbb{M} \) is a finite set of nonsingular integer matrices) and \( L_k \in \mathbb{Z}^{n \times (p-2n)} \) contains at least one column (the column of zeros). Then \( P_k \) is defined by the columns of the matrix \( P_k = BC_k \). Since both \( B \) and \( C_k \) are full row rank, the columns of \( P_k \) span \( \mathbb{R}^n \).

Given mesh size parameter \( \Delta_k \in \mathbb{R}^+ \), a trial step \( s_k \) can be defined as

\[
s_k = \Delta_k B c_k,
\]

where \( c_k \) denotes a column of \( C_k \), and \( B c_k \) is the direction of the step, while \( \Delta_k \) serves as a step length parameter.

The processes of sampling the mesh points around the current iterate are called exploratory moves. In order to establish the convergence of the pattern search methods, exploratory moves need to satisfy the following two hypotheses.

1. \( s_k \in \Delta_k P_k = \Delta_k B C_k \equiv \Delta_k [B \Gamma k B L_k], \) i.e. the direction of any step \( s_k \) accepted at iteration \( k \) is defined by the pattern \( P_k \), and its length is determined by \( \Delta_k \).
2. If \( \min\{f(x_k + y), y \in \Delta_k B \Gamma_k\} < f(x_k) \), then \( f(x + s_k) < f(x_k) \).

The following is the framework of generalized pattern search method.

**The Generalized Pattern Search Method**

**step 1:** Let \( x_0 \) and \( \Delta_0 \) be given.

**step 2:** For \( k = 0, 1, \ldots \)

1. Compute \( f(x_k) \).
2. Determine a step \( s_k \) using an exploratory moves algorithm.
3. If \( f(x_k) > f(x_k + s_k) \), then set \( x_{k+1} = x_k + s_k \). Otherwise \( x_{k+1} = x_k \).
4. Update \( C_k \) and \( \Delta_k \).

To define a particular pattern search method, the basis matrix \( B \), the generating matrix \( C_k \), the exploratory moves and the algorithms for updating \( C_k \) and \( \Delta_k \) must be specified.

For example, the coordinate search method uses the identity matrix as the basis matrix, i.e. \( B = I \). The generating matrix is fixed across all iterations of the method and contains in its columns all possible combinations of \( \{-1, 0, 1\} \). Thus, take \( n = 2 \) for example, we have

\[
C = \begin{bmatrix}
1 & 0 & -1 & 0 & 1 & 1 & -1 & -1 & 0 \\
0 & 1 & 0 & -1 & 1 & -1 & -1 & 1 & 0
\end{bmatrix}.
\]

From “Poll Step” in the algorithm of coordinate search (Subsection 2.1), the method of updating step length \( \Delta_k \) is obvious, that is, if no decrease in objective
function is obtained, shrink the mesh size by $\Delta_{k+1} = \frac{1}{2}\Delta_k$. The exploratory moves algorithm for coordinate search is as follows:

**Exploratory Moves Algorithm for Coordinate Search**

1. **step 1:** Given $x_k$, $\Delta_k$, set $s_k = 0$ and $f_{best} = f(x_k)$.
2. **step 2:** For $j = 1, 2, ..., n$
   - (1) $s^j_k = s_k + \Delta_k B e_j$ and $x^j_k = x_k + s^j_k$.
   - (2) If $f(x^j_k) < f_{best}$ then set $f_{best} = f(x^j_k)$ and $s_k = s^j_k$.
   - (3) Otherwise,
     - (i) $s^j_k = s_k - \Delta_k B e_j$ and $x^j_k = x_k + s^j_k$.
     - (ii) If $f(x^j_k) < f_{best}$, $f_{best} = f(x^j_k)$ and $s_k = s^j_k$.
3. **step 3:** Return.

By exploiting the characterization of pattern search methods, Torczon [64] established a global convergence theory for generalized pattern search methods, which is the first convergence result for some of these methods and the first general convergence theory for all of them.

First, under a very mild condition on the objective function $f$, the step length control parameter $\Delta_k$ has following limiting behavior:

**Theorem 2.1.** Assume that $L(x_0) = \{x : f(x) \leq f(x_0)\}$ is compact. Then $\lim \inf_{k \to +\infty} \Delta_k = 0$.

Assume that $f$ is continuously differentiable on a neighborhood of $L(x_0)$, the following global convergence can be established:

**Theorem 2.2.** Assume that $L(x_0)$ is compact and that $f$ is continuously differentiable on a neighborhood of $L(x_0)$. Then for the sequence of iterates $\{x_k\}$ produced by the generalized pattern search method,

$$\lim \inf_{k \to +\infty} \|\nabla f(x_k)\| = 0.$$  

In general, on the positive side, pattern search methods are reliable methods that could guarantee the convergence to the solutions. And if the evaluations of the objective function are not too expensive, pattern search methods are easy to implement. However, on the negative side, pattern search methods sometimes converge slowly and the number of optimization variables that they can handle is not large.

2.4. The Nelder-Mead Simplex Methods. The Nelder-Mead simplex method, also known as downhill simplex method [48], is one of the most popular direct search method for nonlinear unconstrained optimization. It has been extensively applied in many fields of science and technology.

Starts with a set of finite points that form a simplex, in each iteration, the Nelder-Mead simplex method evaluates the objective function at the vertices of the simplex and finds the worst one. The algorithm then replace the worst vertex by introducing a new point in a way that the remaining vertices and the new point forms a new simplex. The replacement points are obtained by reflecting, expanding, or contracting the worst vertex over the centroid of the remaining vertices.

The Nelder-Mead method is simplex-based and a simplex $S$ in $\mathbb{R}^n$ is defined as the convex hull of $n + 1$ affinely independent points $Y = \{y^0, y^1, ..., y^n\}$. For example, a simplex in $\mathbb{R}^2$ is a triangle, and a simplex in $\mathbb{R}^3$ is a tetrahedron.
The following is the Nelder-Mead algorithm.

**The Nelder-Mead Algorithm**

**Initialization:**
Choose an initial simplex of vertices $Y_0 = \{y^0, y^1, ..., y^n\}$. Choose constants:

\[ -1 < \rho^c < 0 < \rho^{oc} < \rho^r < \rho^e, \quad 0 < \delta^e < 1. \]

**For:** $k = 0, 1, 2, ...$

0. Set $Y = Y_k$.

1. **Order:** Order the $n + 1$ vertices of $Y = \{y^0, y^1, ..., y^n\}$ so that

\[ f(y^0) \leq f(y^1) \leq ... \leq f(y^n). \]

2. **Reflect:** Reflect the worst vertex $y^n$ over the centroid $y^c = \sum_{i=0}^{n-1} y^i/n$ of the remaining vertices:

\[ y^r = y^c + \rho^r (y^n - y^c). \]

If $f(y^r) < f(y^n)$, then replace $y^n$ by $y^r$ and update the set of vertices to $Y_{k+1} = \{y^0, y^1, ..., y^{n-1}, y^r\}$. Otherwise, replace $y^n$ by $y^r$ and update the set of vertices to $Y_{k+1} = \{y^0, y^1, ..., y^{n-1}, y^n\}$.

3. **Expand:** If $f(y^r) < f(y^0)$, compute the expansion point $y^e$

\[ y^e = y^c + \rho^e (y^n - y^c). \]

If $f(y^e) < f(y^r)$, then replace $y^n$ by $y^e$ and update the set of vertices to $Y_{k+1} = \{y^0, y^1, ..., y^{n-1}, y^e\}$. Otherwise, replace $y^n$ by $y^r$ and update the set of vertices to $Y_{k+1} = \{y^0, y^1, ..., y^{n-1}, y^n\}$.

4. **Contract:** If $f(y^r) \geq f(y^{n-1})$, perform a contraction.

a. **Outside.** If $f(y^{n-1}) \leq f(y^r) < f(y^n)$ (i.e. $y^r$ strictly better than $y^n$), then perform an outside contraction

\[ y^{oc} = y^c + \rho^{oc} (y^n - y^c). \]

If $f(y^{oc}) \leq f(y^r)$, then replace $y^n$ by $y^{oc}$ and update the set of vertices to $Y_{k+1} = \{y^0, y^1, ..., y^{n-1}, y^{oc}\}$. Otherwise, perform a shrink.

a. **Inside.** If $f(y^r) > f(y^n)$, perform an inside contraction

\[ y^{ic} = y^c + \rho^{ic} (y^n - y^c). \]

If $f(y^{ic}) < f(y^n)$, then replace $y^n$ by $y^{ic}$ and update the set of vertices to $Y_{k+1} = \{y^0, y^1, ..., y^{n-1}, y^{ic}\}$. Otherwise, perform a shrink.

5. **Shrink:** Retain the best point $y^0$, throw away all the other points, and evaluate $f$ at $n$ new points

\[ v^i = y^0 + \delta^e (y^i - y^0), \quad i = 1, 2, ..., n. \]

We obtain the simplex with vertices $Y_{k+1} = \{y^0, v^1, v^2, ..., v^n\}$ for the next iteration.

The Nelder-Mead simplex algorithm is simple to understand and very easy to implement. Typically, it needs only one or two function evaluations per iteration, while many other direct search methods use $n$ or even more function evaluations. However, Nelder-Mead simplex algorithm has been observed to be unreliable since the shape of the simplices may deteriorate. For instance, McKinnon [45] constructed a family of strictly convex functions in two dimension to show that the Nelder-Mead simplex algorithm can converge to a non-minimizing point.
To ensure convergence, numerous variants of Nelder-Mead simplex algorithm have been proposed [65, 47, 55]. These variants generally improve the original algorithm by imposing the following requirements, either enforce a sufficient decrease condition for accepting new iterates or monitor and improve the quality of the geometry of the simplices, or take safeguards against the failure in determining reflection, expansion, or contraction. For example, Tseng [65] proposed a variant that requires a “fortified descent” condition along with uniform linear independence of the simplex edges. And this method was proved to converge to a stationary point under some mild conditions.

3. Model-Based Methods. The model-based methods were introduced by Winfield in his thesis during the late of 1960’s. In [66] and [67], Winfield constructed a quadratic model of the objective function by interpolating with the available objective function values. This model is assumed to be a good approximation to the objective function in a neighborhood of the current iterate, which is known as a trust region. The model is then minimized within the trust region to locate the next trial point. As the algorithm proceeds, the set of sample points used to interpolate and the model is updated iteratively. This method has been proved to be efficient and reliable, but Winfield did not provide any convergence theorem for this method nor a concrete method for generating the sample sets. About twenty-five years later, Powell [50] proposed a model-based method for constrained optimization, where the objective function and the constrains are approximated by linear models. Then, Powell [51] described a new model-based method for unconstrained optimization using a quadratic model of the objective function. In this method, the set of sample points is updated in a manner that maintains its geometric properties so that to ensure the model established from it is sufficiently accurate. Following ideas expressed in [51], Conn et al. [19] proposed their model-based method, and the first convergence theorem for methods of this type was established by Conn et al. [15]. Furthermore, some research [15, 17] established the global convergence for model-based trust-region methods under certain assumptions and many variants of model-based derivative-free methods are proposed for various optimization problems [26, 56, 59]. Efficient implementations and commercial codes are also proposed, such as the oldest DFO package developed by Conn and coworkers [15, 19], the UOBYOA and NEWUOA packages presented by Powell [52, 53] and the packages UDFO and PSDFO of Colson and Toint [10, 11]. Within the past few years, many researchers have become preoccupied with using stochastic methods to solve optimization problems and some stochastic derivative-free optimization methods have been proposed [6, 34].

The model functions used in model-based methods are usually smooth and easy to evaluate, typical choices are linear and quadratic functions. Lacking derivative information, the model functions are typically established via interpolation. Linear models are appealing for they are simple to construct and update. If we choose properly, only \( n + 1 \) ( \( n \) is the number of variables) interpolation points are needed to build a linear model and the amount of routine work of each iteration is only of magnitude \( n^2 \). However, with the absence of curvature information the algorithm can hardly be efficient. In comparison with linear models, quadratic models take account of the curvature of the objective function, so they can approximate the objective function more accurate and are often very efficient and useful in many
optimization algorithms. However, at least $\frac{1}{2}(n+1)(n+2)$ sample points are needed to establish a complete quadratic model and the complexity of each iteration is $O(n^4)$. Thus, the use of complete quadratic models becomes intolerable and impractical for large $n$. In this paper, we use quadratic model functions to describe the model-based methods.

Since the model function is usually non-convex and only reliable in a region around the current iterate, model-based methods typically use trust-region approach to minimize the model function. Trust-region technique is a development of the line search and is superior to it. Though it is equivalent to the Levenberg-Marquart method, trust-region technique is more efficient in practice. Furthermore, in some extent, trust-region technique is also equivalent to the regularization approach, which guarantee the robustness of this technique. For a comprehensive introduction to trust-region techniques we refer readers to [13] and [49].

3.1. Description of the Model-Based Trust-Region Methods. In this subsection, we describe in detail the general model-based trust-region methods which take quadratic functions as the model functions.

3.1.1. Construction of a quadratic model function from sample points. First, let us take a close look at the construction of the model function. At $k$-th iteration, we model the objective function $f$ by a quadratic function

$$m_k(x_k + s) = c_k + g_k^T s + \frac{1}{2} s^T H_k s,$$

(4)

where $x_k$ denotes the $k$-th iterate, the vector $s$ represents a step from the iterate $x_k$. $c_k \in \mathbb{R}$, $g_k \in \mathbb{R}^n$ and the symmetric matrix $H_k \in \mathbb{R}^{n \times n}$ are parameters to be determined. Assume we have a sample set $Y = \{y^0, y^1, ..., y^p\} \subset \mathbb{R}^n$, the model function $m_k$ then can be determined from the interpolation conditions

$$m_k(y^i) = f(y^i), \quad i = 0, 1, ..., p.$$  

(5)

This is a linear system with $p$ equations and $\frac{1}{2}(n+1)(n+2)$ elements to be determined. Therefore, in the interpolation case, at least $\frac{1}{2}(n+1)(n+2)$ sample points are needed to determine $g_k$ and $H_k$ uniquely. Generally, we assume $p = \frac{1}{2}(n+1)(n+2)$.

However, the condition (5) alone can not ensure the existence and the uniqueness of the interpolation or the good quality of the model. Conn et al. [16] showed that the error bound between an interpolating polynomial and the true function is connected with a constant that reflects the geometric quality (known as poisedness) of the sample set. In other words, in the model-based algorithm, we should focus on how to maintain the sample set well-poised so that the corresponding constant remains small (at least uniformly bounded) and to guarantee the model good enough.

Let $\phi = \{\phi_0(x), \phi_1(x), ..., \phi_p(x)\}$ be a basis of $P^2_n$, then the system of linear equations (5) can be written as

$$M(\phi, Y) \alpha_\phi = f(Y),$$  

(6)

where

$$M(\phi, Y) = \begin{bmatrix}
\phi_0(y^0) & \phi_1(y^0) & \cdots & \phi_p(y^0) \\
\phi_0(y^1) & \phi_1(y^1) & \cdots & \phi_p(y^1) \\
\vdots & \vdots & \ddots & \vdots \\
\phi_0(y^p) & \phi_1(y^p) & \cdots & \phi_p(y^p)
\end{bmatrix}.$$
\[ \alpha = \begin{bmatrix} \alpha_0 \\ \alpha_1 \\ \vdots \\ \alpha_p \end{bmatrix}, \quad f(Y) = \begin{bmatrix} f(y^0) \\ f(y^1) \\ \vdots \\ f(y^p) \end{bmatrix}. \]

If the matrix \( M(\phi, Y) \) is nonsingular, then the above system has a unique solution and the sample set \( Y \) is said to be poised. From practical point of view, it is necessary to have a quantitative measure for poisedness. To define the most commonly used measure of poisedness in polynomial interpolation, we need the basis of Lagrange polynomials.

**Definition 3.1.** Given a set of sample points \( Y = \{ y^0, y^1, \ldots, y^p \} \), a basis of \( p + 1 \) polynomials \( l_j(x) \), \( j = 0, \ldots, p \), in \( P^d_n \) is called a basis of Lagrange polynomials if

\[ l_j(y^i) = \delta_{ij} = \begin{cases} 1 & \text{if } i = j, \\ 0 & \text{if } i \neq j. \end{cases} \]

Then we have the following definition of \( \Lambda \)-poisedness.

**Definition 3.2.** Let \( \Lambda > 0 \) and a set \( B \subset \mathbb{R}^n \) be given. Let \( \phi = \{ \phi_0(x), \phi_1(x), \ldots, \phi_p(x) \} \) be a basis in \( P^d_n \). Then a poised set \( Y = \{ y^0, y^1, \ldots, y^p \} \) is said to be \( \Lambda \)-poised in \( B \) for \( P^d_n \) (in the interpolation sense) if and only if, for any \( x \in B \) there exists \( \lambda(x) \in \mathbb{R}^{p+1} \) such that

\[ \sum_{i=0}^{p} \lambda_i(x) \phi(y^i) = \phi(x) \quad \text{with} \quad \| \lambda(x) \|_\infty \leq \Lambda. \]

\( \Lambda \)-poisedness is an important measure of poisedness and reflects the geometric property of a interpolation set \( Y \subset \mathbb{R}^n \). The definition we mentioned above and two other equivalent ones can be found in [18].

3.1.2. Trust-region Subproblem. Once \( m_k \) has been constructed around the current iterate \( x_k \), we compute a step \( s_k \) by approximately solving the following trust-region subproblem

\[ \min_s m_k(x_k + s), \quad \text{s.t.} \quad \|s\| \leq \Delta_k, \quad (7) \]

where \( \Delta_k > 0 \) is the trust-region radius.

If \( x_k + s_k \) gives a sufficient reduction of \( f \), then set \( x_{k+1} = x_k + s_k \), update \( \Delta_k \), and add \( x_{k+1} \) into \( Y_k \). Otherwise, the step \( s_k \) is rejected, and \( Y_k \) maybe improved or trust region shrunk or model-improving performed.

To reduce the cost of the algorithm, the model function usually be improved, rather than be recomputed from scratch at every iteration.

3.1.3. Model-Based Trust-Region Algorithm. Let the ratio of the actual reduction in the objective function versus the predicted reduction in model be

\[ \rho_k = \frac{f(x_k) - f(x_k^+)}{m_k(x_k) - m_k(x_k^+)}. \quad (8) \]

Then, we state the general form of the model-based trust-region algorithm:

**step 0:** Choose a poised sample set \( Y = \{ y^0, y^1, \ldots, y^p \} \). Set \( k = 0 \), choose \( x_0 \in Y \), such that \( f(x_0) \leq f(y^i), \forall y^i \in Y \). Given \( \Delta_0, \eta \in (0, 1) \).
step 1: Repeat until a convergence test is satisfied:

Construct the model function \( m_k(x_k + s) \) that satisfies (5); Compute a step \( s_k \) by approximately solving subproblem (7); Set the trial point as \( x_k = x_k + s_k \), compute the ratio \( \rho \) by (8);

- if \( \rho \geq \eta \)
  Replace an element of \( Y \) by \( x^+_k \), choose \( \Delta_{k+1} \geq \Delta_k \), set \( x_{k+1} := x^+_k, k := k + 1 \) and go to the next iteration;
- else if the set \( Y \) need not to be improved
  Choose \( \Delta_{k+1} < \Delta_k \), set \( x_{k+1} := x_k, k := k + 1 \) and go to the next iteration;
- else (\( \rho < \eta \) and that the quality of \( Y \) is not good)
  Perform a geometry-improving procedure to update \( Y \), set \( \Delta_{k+1} := \Delta_k \);
  Choose \( \hat{x} \in Y \) with lowest function value, set \( x^+_k = \hat{x} \), compute \( \rho \);
  If \( \rho \geq \eta \), set \( x_{k+1} = x^+_k \). Otherwise \( x_{k+1} = x_k \), set \( k := k + 1 \) and go to the next iteration.

3.2. Convergence Analysis. In this subsection, first-order and second-order global convergence of model-based trust-region methods are stated. We just only list the main results, the readers can refer to [17, 18] for more details. First, we describe some assumptions on the objective function and models.

Given the initial point \( x_0 \) and the upper bound on the size of the trust regions \( \Delta_{\text{max}} \), we restrict in this paper all potential iterates to the region

\[
L_{\text{enl}}(x_0) = L(x_0) \cup \bigcup_{x \in L(x_0)} B(x, \Delta_{\text{max}}),
\]

where

\[
L(x_0) = \{x \in \mathbb{R}^n : f(x) \leq f(x_0)\}.
\]

3.2.1. First-order convergence. For first-order convergence, we suppose the function \( f \) and its gradient are Lipschitz continuous in the region \( L_{\text{enl}}(x_0) \).

**Assumption 3.1.** Suppose \( x_0 \) and \( \Delta_{\text{max}} \) are given. Assume that \( f \) is continuously differentiable with Lipschitz continuous \( \nabla f \) in an open domain containing the set \( L_{\text{enl}}(x_0) \), and \( f \) is bounded from below on \( L(x_0) \).

In derivative-based trust-region methods, a quadratic model function \( m_k \) is Taylor based that formed by using information of \( f \) at current iterate \( x_k \), that is, \( m_k \) is given by

\[
m_k(x_k + s) = c_k + g_k^T s + \frac{1}{2} s^T H_k s,
\]

where \( g_k = \nabla f(x_k) \) and \( H_k = \nabla^2 f(x_k) \). Thus, the error bounds between the model \( m_k \) and the true function \( f \) are easy to estimate and control. However, in derivative-free methods, models are formed by interpolation, regression or other sample based methods. In order to ensure global convergence, the models have to satisfy Taylor-like error bounds on the function value and the gradient.

For first-order convergence, we need the model function be fully linear.

**Definition 3.3.** Assume function \( f : \mathbb{R}^n \to \mathbb{R} \) satisfy Assumption 3.1, and \( f \) is bounded from below on \( L(x_0) \). We say that a function \( m_k \in C^1 \) is a fully linear model of \( f \) on \( B(x_k, \Delta_k) \) if, for any \( s \in B(0, \Delta_k) \), we have

\[
\| \nabla f(x_k + s) - \nabla m_k(x_k + s) \| \leq \kappa_{eg} \Delta_k, \tag{9}
\]

\[
| f(x_k + s) - m_k(x_k + s) | \leq \kappa_{ef} \Delta_k^2. \tag{10}
\]
Follow Theorem 5.4 in [18], we have the following lemma to show that models based on $\Lambda$-poised sample set would be at least local fully linear.

**Lemma 3.4.** Given any $\Delta > 0, z \in \mathbb{R}^n$, we assume $Y = \{y^0, y^1, ..., y^p\} \subset B(z, \Delta)$ is $\Lambda$-poised in $B(z, \Delta)$ with $p+1 = C^2_n+2$. Let $m(\cdot) \in P^2_n$ be an interpolating polynomial of $f$ on $Y$. If $f$ is continuously differentiable and $\nabla f$ is Lipschitz continuous with Lipschitz constant $L$ in an open domain $\Omega$ containing $B(z, \Delta)$, then, for any $s \in B(0, \Delta)$, we have where $\hat{\kappa}_{eg}$ and $\hat{\kappa}_{ef}$ are positive constants depending only on $n$ and $\Lambda$.

To establish the first-order convergence, the following two assumptions are also required.

**Assumption 3.2.** There exists a constant $\kappa_H > 0$ such that for every model $m_k$

$$\|\nabla^2 m_k\| \leq \kappa_H. \quad (11)$$

**Assumption 3.3.** At all iterations $k$, we are able to compute a step $s_k$ such that

$$m_k(x_k) - m_k(x_k + s_k) \geq \kappa_{fod}[m_k(x_k) - m_k(x_k + s_k^E)], \quad (12)$$

for some constant $\kappa_{fod} \in (0, 1]$.

The second assumption guarantees that at each iteration the model can be reduced by a fraction of Cauchy decrease at least. $s_k^E$ in (12) is the Cauchy step, which is the step to the minimum of the model within the trust region along the negative model gradient $-g_k$.

With the above assumptions we have the following convergence result.

**Theorem 3.5.** Assume that Assumptions 3.1, 3.2, 3.3 hold. Then

$$\lim_{k \to +\infty} \nabla f(x_k) = 0.$$

3.2.2. **Second-order convergence.** To establish the second-order convergence, we need the Lipschitz continuity of the Hessian of $f$.

**Assumption 3.4.** Suppose $x_0$ and $\Delta_{\text{max}}$ are given. Assume that $f$ is twice continuously differentiable with Lipschitz continuous Hessian in an open domain containing the set $L_{\text{ent}}(x_0)$, and $f$ is bounded from below on $L(x_0)$.

At each iteration, the model reduction should satisfy the following assumption:

**Assumption 3.5.** At all iterations $k$, we are able to compute a step $s_k$ such that

$$m_k(x_k) - m_k(x_k + s_k) \geq \kappa_{fod}[m_k(x_k) - \min\{m_k(x_k + s_k^E), m_k(x_k + s_k^C)\}], \quad (13)$$

for some constant $\kappa_{fod} \in (0, 1]$.

$s_k^E$ in (13) is referred to as eigenstep, which is the eigenvector of $H_k$ corresponding to the most negative eigenvalue $\lambda_{\text{min}}(H_k)$ and satisfies $(s_k^E)^T g_k \leq 0$ and $\|s_k^E\| = \Delta_k$.

Then we have the following second-order global convergence result.

**Theorem 3.6.** Assume that Assumptions 3.2, 3.4, 3.5 hold. Then

$$\lim_{k \to +\infty} \sigma(x_k) = 0,$$

where $\sigma(x_k) = \max\{\|g_k\|, -\lambda_{\text{min}}(H_k)\}$ is the measure of the second-order stationarity of the model.
4. Derivative-Free Methods For Constrained Optimization Problems. Since most practical optimization problems are defined with constraints, the study of effective techniques for handling constrained optimization problems remains an active research area. In this section we briefly review some derivative-free approaches for solving constrained problems, including an algorithm we proposed recently for spherical optimization.

Most derivative-free methods for constrained problems have been developed within the framework of direct search methods. When direct search methods first appeared in 1950s, they have been adapted to solve constrained problems [8, 21, 31, 43]. However, most early proposals are not used today. Recently, some modern approaches have been proposed. In [36, 37], pattern search method is extended to solve problems with bound and general linear constraints respectively. In order to ensure the global convergence of these two extensions, at each iteration the set of search directions must be rich enough to include descent directions and the step should be long enough and keep feasible. In [38, 3], augmented Lagrangian approach and filter method are combined with pattern search method respectively to solve general constrained optimization. In [4], the mesh adaptive direct search (MADS) class of algorithms is proposed for constrained optimization which extends the generalized pattern search (GPS) class by allowing local exploration. In [41], a sequential penalty approach for nonlinear constrained optimization is extended to derivative-free context.

Some model-based methods for constrained derivative-free optimization are also proposed. In [56], a new derivative-free funnel method for equality-constrained nonlinear optimization is presented. This method is of the trust-funnel variety and is based on the use of polynomial interpolation models. In [12], a new algorithm for constrained optimization problems is proposed which handle the constraints in the trust-region subproblems. In each iteration, the objective function is approximated by a model obtained by quadratic interpolation, which is then minimized within the intersection of the feasible set with the trust region. More recently, an affine scaling cubic regularization algorithm for optimization problems with bound constraints is presented in [30].

Recently, we presented a model-based derivative-free algorithm DFOSC for the following spherically constrained optimization problem:

\[
\min_{x \in \mathbb{S}^{n-1}} f(x),
\]

where \(\mathbb{S}^{n-1} \equiv \{x \in \mathbb{R}^n : x^T x = 1\}\) is the unit sphere, \(f : \mathbb{S}^{n-1} \to \mathbb{R}\) is a smooth nonlinear function but its derivatives are unavailable. DFOSC takes trust region framework and explores the nice geometry of the spherical constraint to solve spherically constrained optimization.

In the trust-region framework, DFOSC constructs a simple model to approximate the objective function over the trust region. At iteration \(k\), a simple model is

\[
m_k(x_k + s) = c_k + g_k^T s + \frac{\gamma_k}{2} s^T s,
\]

where \(x_k\) denotes the \(k\)-th iterate, \(c_k, \gamma_k \in \mathbb{R}^1\) and \(g_k \in \mathbb{R}^n\) are parameters to be determined. The simple model \(m_k\) is established by function interpolations and it needs less interpolation points than a full quadratic model, and consequently can reduce the computational complexity.
Once $m_k$ has been constructed, a step $s$ is computed by (approximately) solving the trust-region subproblem

$$
\min_s \ m_k(x_k + s), \quad \text{subject to} \quad s^T x_k = 0 \text{ and } \|s\| \leq \Delta_k,
$$

where $\Delta_k > 0$ is the so-called trust-region radius.

To take advantage of the spherical structure of the problem, DFOSC applies the Cayley transform to preserve iterates on the sphere. Let

$$
W_k = \frac{1}{2} (x_k s_k^T - s_k x_k^T)
$$

be a skew-symmetric matrix. The Cayley transform produces an orthogonal matrix

$$
Q_k = (I + W_k)^{-1}(I - W_k),
$$

whose eigenvalues do not contain $-1$. Hence, the trial point

$$
x_k^+ = Q_k x_k
$$

satisfies the constraint $\|x_k^+\| = 1$. By applying the Cayley transform, the iterates generated by DFOSC are all feasible points. Thus, DFOSC is essentially similar to an algorithm for unconstrained optimization which do not need to use traditional methodologies for nonlinear constraints such as penalty function, augmented Lagrangian or filter methods.

Under mild conditions, we establish the global convergence of DFOSC to first-order criticality points. Preliminary numerical results show that the applicability of the DFOSC algorithm to practical problems. However, we note that, as other derivative-free methods, DFOSC can only be applied to moderate size problems. To extend existing techniques to solve derivative-free constrained optimization problems in high dimensions is a challenging task and warrants further study.

5. Applications of Derivative-Free Optimization in Practice. Derivative-free methods have been successfully applied in many areas of study. These include engineering design [7, 22, 42], molecular geometry [2, 44], geophysics [24, 68] and finance [35, 61], just to name a few. In this section, we describe two examples of these applications.

The estimation of implied volatility and the risk-free rate

An interesting application of derivative-free optimization has been explored in [61] to estimate the implied volatility of an asset and the risk-free rate.

In options trading, a call (put) option is a contract giving the buyer (seller) the right to buy (sell) a specified quantity of the underlying asset at a specific price on or before a certain date. The price of the option contract is known as the premium. Implied volatility represents the expected volatility in an option contract over its lifetime. Implied volatility is an important metric for investors to assess the validity of the premium.

The estimation of implied volatility $\sigma$ and the risk-free rate $r$ can be formulated as the following unconstrained minimization problem

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

where $c : \mathbb{R}^2 \to \mathbb{R}$ and $p : \mathbb{R}^2 \to \mathbb{R}$ denote the call and put option premiums of the volatility $\sigma$ and the risk-free rate $r$, $c^*$ and $p^*$ are the given call and put option premiums.
In [61], Takaki et al. considered two types of options: the look-back option and the Asian option. These two options are different in the way of evaluating premiums \( c(\sigma, r) \) and \( p(\sigma, r) \). Unfortunately, it is impossible to obtain closed-form solutions for the premiums of these two options. In practice, \( c(\sigma, r) \) and \( p(\sigma, r) \) of the look-back option and the Asian option are generally evaluated via Monte Carlo simulation. Therefore, (15) is a problem whose derivatives are unavailable.

In [61], Takaki et al. proposed a model-based derivative-free algorithm for this problem. By using Support Vector Regression (SVR) technique, their model-based method can construct quadratic models with pointwise accuracy and in turn reduce the total computation time of the algorithm. The numerical results show that the new algorithm is encouraging for problem (15).

**Optimization of stirrer configurations**

Stirring process is used in many industries such as food processing, chemical, biotechnological and pharmaceutical. However, the requirements for stirring process are different in various industries. It is a hard task to set various geometric parameters of the stirrers to meet different requirements. In practice, engineers generally use numerical simulation techniques to assist the design of stirrers.

In [57], Schäfer et al. proposed an approach for the numerical optimization of stirrer configurations. In this stirrer optimization problem, the objective function may correspond to the Newton number, mixing time or stresses which are all derived by simulating the complex discrete Navier-Stokes system. Thus, the local gradients of the objective functions are not directly available. In such case, Schäfer et al. [57] used the optimization package DFO to optimize the geometric parameters of the stirrer. DFO package [58] is an open-source implementation of the model-based algorithm originally developed by Conn et al. [14, 15].

In their numerical experiment, Schäfer et al. [57] took a Rushton turbine as a representative test case to optimize a practical stirrer configuration. Their numerical approach achieved a significant reduction of the Newton number with relatively low computational effort.

6. **Conclusion.** In this paper, we have concentrated on methods for solving optimization problems without using derivatives. We first discuss methods for unconstrained optimization problems. We review some classical direct search methods as well as their global convergence and then describe in detail the model-based methods with their global convergence. Next, we briefly review a number of derivative-free approaches for problems with constraints, including an algorithm we recently proposed for spherical optimization. Finally, we describe two examples of the applications of the derivative-free optimization. Derivative-free methods for optimization remains a valuable area of research since the demand from practitioners for them is still high. In addition, most practical problem present constraints and the derivative-free methods for constrained problems is a research area worthy of attention.

**Acknowledgments.** This paper is dedicated to memorize Professor Wenyu Sun, without whose help this research could not have started. We also want to express our sincere thanks to Dr. Zaikun Zhang for inviting the first author to visit the Hong Kong Polytechnic University. It is with his guidance and suggestions that this paper is finished.
REFERENCES

[1] M. A. Abramson and C. Audet, Convergence of mesh adaptive direct search to second-order stationary points, *SIAM Journal on Optimization*, 17 (2006), 606–609.

[2] P. Alberto, F. Nogueira, H. Rocha and L. N. Vicente, Pattern search methods for user-provided points: Application to molecular geometry problems, *SIAM Journal on Optimization*, 14 (2004), 1216–1236.

[3] C. Audet and J. E. Dennis, Jr., A pattern search filter method for nonlinear programming without derivatives, *SIAM Journal on Optimization*, 14 (2004), 980–1010.

[4] C. Audet and J. E. Dennis, Jr., Mesh adaptive direct search algorithms for constrained optimization, *SIAM Journal on Optimization*, 17 (2006), 188–217.

[5] C. Audet, Convergence results for generalized pattern search algorithms are tight, *Optimization and Engineering*, 5 (2004), 101–122.

[6] A. S. Bandeira, K. Scheinberg and L. N. Vicente, Convergence of trust-region methods based on probabilistic models, *Mathematical Programming*, 24 (2014), 1238–1264.

[7] A. J. Booker, J. E. Dennis, Jr., P. D. Frank, D. W. Moore and D. B. Serafini, Managing surrogate objectives to optimize a helicopter rotor design—Further experiments, in *AIAA Paper 1998–4717*, 8th AIAA/ISSMO Symposium on Multidisciplinary Analysis and Optimization, St. Louis, MO, 1998.

[8] M. J. Box, A new method of constrained optimization and a comparison with other methods, *Computer Journal*, 8 (1965), 42–52.

[9] T. D. Choi and C. T. Kelley, Superlinear convergence and implicit filtering, *SIAM Journal on Optimization*, 10 (2000), 1149–1162.

[10] B. Colson and Ph. L. Toint, Exploiting band structure in unconstrained optimization without derivatives, in *Trends in Industrial and Applied Mathematics, Vol. 72 Applied Optimization*, (eds. A.H. Siddiqi and M. Kocvara), Kluwer, (2002), 131–147.

[11] B. Colson and Ph. L. Toint, Optimizing partially separable functions without derivatives, *Optimization Methods and Software*, 20 (2005), 493–508.

[12] P. Conejo, E. Karas and P. Ledroso, A trust-region derivative-free algorithm for constrained optimization, *Optimization Methods and Software*, 30 (2015), 1126–1145.

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

[14] A. R. Conn, K. Scheinberg and Ph. L. Toint, On the convergence of derivative-free methods for unconstrained optimization, in *Approximation Theory and Optimization: Tributes to M. J. D. Powell* (eds. A. Isac and M. Buhmann), Cambridge University Press, Cambridge, (1996), 83–108.

[15] A. R. Conn, K. Scheinberg and Ph. L. Toint, Recent progress in unconstrained nonlinear optimization without derivatives, *Mathematical Programming*, 79 (1997), 397–414.

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

[17] A. R. Conn, K. Scheinberg and L. N. Vicente, Global convergence of general derivative-free trust-region algorithms to first- and second-order critical points, *SIAM Journal on Optimization*, 20 (2009), 387–415.

[18] A. R. Conn, K. Scheinberg and L. N. Vicente, *Introduction to Derivative-free Optimization*, MPS/SIAM Ser. Optim. 8, SIAM, Philadelphia, 2009.

[19] A. R. Conn and Ph. L. Toint, An algorithm using quadratic interpolation for unconstrained derivative-free optimization, in *Nonlinear Optimization and Application* (eds. G. Di. Pillo and F. Giannessi), Plenum Publishing, New York, (1996), 27–47.

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

[21] L. C. W. Dixon, ASCIM—an accelerated constrained simplex technique, *Computer-aided Design*, 5 (1973), 22–32.

[22] R. Duvigneau and M. Visonneau, Hydrodynamic design using a derivative-free method, *Structural and Multidisciplinary Optimization*, 28 (2004), 195–205.

[23] E. Fermi and N. Metropolis, Los Alamos Unclassified Report LS–1492, in *Technical Report*, Los Alamos National Laboratory, Los Alamos, NM, 1952.

[24] K. R. Fowler, J. P. Reese, C. E. Kees, J. E. Dennis, Jr., C. T. Kelley, C. T. Miller, C. Audet, A. J. Booker, G. Couyure, R. W. Darwin, M. W. Farthing, D. E. Finkel, J. M. Gablonsky, G. Gray and T. G. Kolda, A comparison of derivative-free optimization methods for groundwater...
supply and hydraulic capture community problems, Advances in Water Resources, 31 (2007), 743–757.
[25] P. Gilmore and C. T. Kelley, An implicit filtering algorithm for optimization of functions with many local minima, SIAM Journal on Optimization, 5 (1995), 269–285.
[26] G. N. Grapiglia, J. Yuan and Y. Yuan, A derivative-free trust-region algorithm for composite nonsmooth optimization, Journal of Computational and Applied Mathematics, 35 (2016), 475–499.
[27] W. L. Hare and R. A. Poliquin, Prox-regularity and stability of the proximal mapping, Journal of Convex Analysis, 14 (2007), 589–606.
[28] G. H. Golub and C. F. Van Loan, Matrix Computations, Johns Hopkins University Press, Baltimore, MD, 2013.
[29] R. Hooke and T. A. Jeeves, Direct search solution of numerical and statistical problems, Journal of the Association for Computing Machinery, 8 (1961), 212–229.
[30] X. J. Huang and D. T. Zhu, An interior affine scaling cubic regularization algorithm for derivative-free optimization subject to bound constraints, Journal of Computational and Applied Mathematics, 321 (2017), 108–127.
[31] D. L. Keeder, Simpat: Self-bounding direct search method for optimization, Industrial and Engineering Chemistry Process Design and Development, 12 (1973), 92–99.
[32] C. T. Kelley, Iterative Methods for Optimization, Frontiers Appl. Math. 18, SIAM, Philadelphia, 1999.
[33] T. G. Kolda and V. J. Torczon, On the convergence of asynchronous parallel pattern search, SIAM Journal on Optimization, 14 (2004), 939–964.
[34] J. Larson and S. C. Billups, Stochastic derivative-free optimization using a trust region framework, Computational Optimization and Applications, 64 (2016), 619–645.
[35] T. Levina, Y. Levin, J. Mcgill and M. Nediak, Dynamic pricing with online learning and strategic consumers: An application of the aggregating algorithm, Operations Research, 57 (2009), 327–341.
[36] R. M. Lewis and V. Torczon, Pattern search algorithms for bound constrained minimization, SIAM Journal on Optimization, 9 (1999), 1082–1099.
[37] R. M. Lewis and V. Torczon, Pattern search methods for linearly constrained minimization, SIAM Journal on Optimization, 10 (2000), 917–941.
[38] R. M. Lewis and V. Torczon, A globally convergent augmented Lagrangian pattern search algorithm for optimization with general constraints and simple bounds, SIAM Journal on Optimization, 12 (2002), 1075–1089.
[39] K. I. M. McKinnon, Convergence of the Nelder-Mead simplex method to a nonstationary point, SIAM Journal on Optimization, 9 (1998), 148–158.
[40] J. J. Moré and S. M. Wild, Benchmarking derivative-free optimization algorithms, SIAM Journal on Optimization, 20 (2009), 172–191.
[41] L. Nazareth and P. Tseng, Gilding the lily: A variant of the Nelder-Mead algorithm based on golden-section search, Computational Optimization and Applications, 22 (2002), 133–144.
[42] J. A. Nelder and R. Mead, A simplex method for function minimization, Computer Journal, 7 (1965), 308–313.
[43] J. Nocedal and S. J. Wright, Numerical Optimization, 2nd edition, Springer-Verlag, New York, 2006.
[44] M. J. D. Powell, A direct search optimization method that models the objective and constraint functions by linear interpolation, in Advances in Optimization and Numerical Analysis (eds. S. Gomez and J-P. Hennart), Kluwer Academic Publishers, Dordrecht, (1994), 52–67.
[51] M. J. D. Powell, A direct search optimization method that models the objective by quadratic interpolation, in Presentation at the 5th Stockholm Optimization Days, 1994.

[52] M. J. D. Powell, UOBYQA: Unconstrained optimization by quadratic approximation, Mathematical Programming, 92 (2002), 555–582.

[53] M. J. D. Powell, The NEWUOA software for unconstrained optimization without derivatives, in Large-Scale Nonlinear Optimization (eds. G. Di Pillo and M. Roma), Springer, New York, 2006.

[54] M. J. D. Powell, The BOBYQA algorithm for bound constrained optimization without derivatives, Technical Report, Department of Applied Mathematics and Theoretical Physics, University of Cambridge, 2009.

[55] C. J. Price, I. D. Coope and D. Byatt, A convergent variant of the Nelder Mead algorithm, Journal of Optimization Theory and Applications, 113 (2002), 5–19.

[56] P. R. Sampaio and Ph. L. Toint, A derivative-free trust-funnel method for equality-constrained, Computational Optimization and Applications, 61 (2015), 25–49.

[57] M. Schäfer, B. Karasözen, Y. Uludağ, K. Yapıcı and Ö. Uğur, Numerical method for optimizing stirrer configurations, Computers and Chemical Engineering, 30 (2005), 183–190.

[58] K. Scheinberg, Manual for Fortran Software Package DFO v2.0, 2003.

[59] K. Scheinberg and Ph. L. Toint, Self-correcting geometry in model-based algorithms for derivative-free unconstrained optimization, SIAM Journal on Optimization, 20 (2010), 3512–3532.

[60] W. Spendley, G. R. Hext and F. R. Himsworth, Sequential application for simplex designs in optimisation and evolutionary operation, Technometrics, 4 (1962), 441–461.

[61] J. Takaki and N. Yamashita, A derivative-free trust-region algorithm for unconstrained optimization with controlled error, Numerical Algebra, Control and Optimization, 1 (2011), 117–145.

[62] V. Torczon, Multi-directional Search: A Direct Search Algorithm For Parallel Machines, Ph.D thesis, Rice University, Houston, TX, 1989.

[63] V. Torczon, On the convergence of multidirectional search algorithms, SIAM Journal on Optimization, 1 (1991), 123–145.

[64] V. Torczon, On the convergence of pattern search algorithms, SIAM Journal on Optimization, 7 (1997), 1–25.

[65] P. Tseng, Fortified-descent simplicial search method: A general approach, SIAM Journal on Optimization, 10 (2000), 269–288.

[66] D. Winfield, Function and Functional Optimization by Interpolation in Data Table, Ph.D Thesis, Harvard University, Cambridge, MA, 1969.

[67] D. Winfield, Function minimization by interpolation in a data table, J. Inst. Math. Appl., 12 (1973), 339–347.

[68] J.-H. Yoon and C. A. Shoemaker, Comparison of optimization methods for ground-water bioremediation, Journal of Water Resources Planning and Management, 125 (1999), 54–63.

Received April 2020; 1st revision August 2020; Final revision September 2020.

E-mail address: mxi@dufs.edu.cn
E-mail address: wysun@njnu.edu.cn
E-mail address: junchennj@163.com