A WEDGE TRUST REGION METHOD WITH SELF-CORRECTING GEOMETRY FOR DERIVATIVE-FREE OPTIMIZATION

LIANG ZHANG AND WENYU SUN
School of Mathematical Sciences, Jiangsu Key Laboratory for NSLSCS
Nanjing Normal University, Nanjing 210023, China
RAIMUNDO J. B. DE Sampaio
PPGEPs, Pontifical Catholic University of Parana (PUCPR)
Curitiba, Parana, Brazil
JINYUN YUAN
Department of Mathematics
Universidade Federal do Parana (UFPR)
Curitiba, Parana, Brazil

Abstract. Recently, some methods for solving optimization problems without derivatives have been proposed. The main part of these methods is to form a suitable model function that can be minimized for obtaining a new iterative point. An important strategy is geometry-improving iteration for a good model, which needs a lot of calculations. Besides, Marazzi and Nocedal (2002) proposed a wedge trust region method for derivative free optimization. In this paper, we propose a new self-correcting geometry procedure with less computational efforts, and combine it with the wedge trust region method. The global convergence of new algorithm is established. The limited numerical experiments show that the new algorithm is efficient and competitive.

1. Introduction. In this paper, we discuss the unconstrained optimization problems
\[ \min_{x \in \mathbb{R}^n} f(x), \quad (1) \]
where \( f \) is a smooth function of several variables whose derivatives are unavailable or unreliable. This class of problems exists widely in practice and is called as the derivative-free optimization (DFO) problems. The Derivative-Free Optimization (DFO) has a history of more than a half century and is rapidly developed recently due to a growing number of applications that range from science problems [9], medical problems [18], engineering design and facility location problems [7].

The common methods for optimization involve derivatives, for example, the conjugate gradient methods need first-order derivatives; the Newton methods use the
information of the second-order derivatives. What’s more, the necessary and sufficient optimality conditions also need derivatives. Please refer Nocedal and Wright [17] (1999), Sun and Yuan [33] (2006), Dai etc. [5] (2006), and Powell [27] (2010).

If we cannot get the derivative, how to deal with it? The first idea to handle this issue may be the finite-difference approach. But there are many cases where the finite-difference does not work. First, if the value of the objective function is generated by black-box packages, we can only get the objective function values if we have the input data, but we cannot get the specific expression and the gradient information. It is the same if the code of functions is written in the way we cannot read and translate. Second, when the function evaluation is costly, it needs a prohibitive cost to provide the gradient estimation. Third, if the objective function is noisy, the gradient estimation may be completely useless, and the finite-difference approach is unreliable in the presence of noise. So the derivative-free optimization methods come into being.

In fact, DFO is not a new thing. The earliest ideas came out in the 1960s. The Hooke-Jeeves method [10] in 1961 and the Nelder-Mead method [16] in 1965 are the early literatures of direct search methods. In 1969 and 1973, Winfield [35, 36] put forward the first model-based method for derivative-free optimization. He constructed a quadratic model function to approximate the objective function by interpolation. Powell [19] in 1970 proposed the modern trust region method for unconstrained optimization in which he established the convergence theory and implementable algorithms (see [19, 20, 28]). Afterwards, several researches indicate that the modern trust region methods are efficient and robust for various optimization problems including the optimization problems without derivatives (see [2, 11, 15, 31, 32, 33, 38, 40]). In 1994 and 1998, Powell [21, 22, 25] proposed the model-based method by interpolation to handle the derivative-free optimization problems. In particular, Powell published some famous softwares, for example, “UOBYQA” [24] and “NEWUOA” [26], for solving DFO problems. In addition, Conn etc. [3] published a derivative-free optimization method by using the Newton interpolation models. Xue and Sun [37] established the convergence analysis of derivative-free trust region algorithm for constrained optimization with separable structure. Zhang etc. [39] developed a class of derivative-free algorithms for least-squares minimization problems by use of polynomial interpolation models. However, since there are several difficult issues and expensive computational efforts, so far, DFO algorithms can only handle small and medium scale optimization problems without derivatives. Now DFO algorithms are faced with big challenge and developing space.

An important issue of model-based methods in DFO is that the set of interpolation points should meet some geometry conditions under which we can get a suitable model function and global convergence. Generally, some geometry-improvement steps to assure of the suitability of the model function are used, see [3, 24, 26]. Marazzi and Nocedal [12, 13] proposed wedge trust region methods which add a wedge constraint in the trust region subproblem such that new generated step avoids approaching to the region in which the set of interpolation points is degenerate. Scheinberg and Toint [29] presented a self-correcting geometry process to achieve the goal of poisedness. Gratton, Toint, Tröltzsch [8] solved the bound-constrained optimization problems by using the technique of self-correcting geometry. However, we find that the self-correcting geometry needs expensive computation efforts, and
A WEDGE TRUST REGION METHOD

has not ideal effect when it combines with the basic trust region subproblem. Aiming at the defects, we design a new self-correcting geometry strategy which requires less computational costs. In addition, we put the strategy into the framework of the wedge trust region model, so that the new step generated by the algorithm avoids approaching to the area containing the degenerate set of sample points. Further, we give numerical experiments which show our work is interesting.

The paper is organized as follows. In Section 2 we first introduce some preliminaries about interpolation schemes, and then design the new self-correcting geometry strategy and propose our wedge trust region algorithm with self-correcting geometry. In Section 3, we prove the global convergence of our algorithm. In Section 4, we report the numerical experiments which show that our new algorithm is competitive. Finally, we make some conclusions.

2. A Wedge Trust Region Method with Self-Correcting Geometry for Optimization.

2.1. Interpolation models. There are some different methods to construct the interpolation model, for example, Lagrange interpolation [1, 23], Newton interpolation [3], radial basis function interpolation [18, 34] and so on. Here we adopt the Lagrange interpolation.

We consider the general polynomial interpolation model. Let \( P_d^n \) denote the space of polynomials of degree \( \leq d \) in \( \mathbb{R}^n \) and let \( p_1 = n + 1 \) be the dimension of this space. One has \( p_1 = n + 1 \) for \( d = 1 \) (linear model case) and \( p_1 = \frac{1}{2} (n+1)(n+2) \) for \( d = 2 \) (quadratic model case). A basis \( \Phi = \{ \phi_0(x), \phi_1(x), \ldots, \phi_p(x) \} \) of \( P_d^d \) is a set of \( p_1 \) polynomials of degree \( \leq d \) that spans \( P_d^d \). For such a basis \( \Phi \), any polynomial \( m(x) \in P_d^d \) can be written as

\[
m(x) = \sum_{j=0}^{p_1} \alpha_j \phi_j(x),
\]

where \( \alpha_0, \ldots, \alpha_p \) are real coefficients. We say that the polynomial \( m(x) \) interpolates the function \( f(x) \) at a given point \( y \) if \( m(y) = f(y) \).

Assume that we give a set \( Y = \{ y^0, y^1, \ldots, y^p \} \subset \mathbb{R}^n \) of interpolation points and let \( m(x) \) denote a polynomial of degree \( d \) in \( \mathbb{R}^n \) that interpolates a given function \( f(x) \) at the points in \( Y \). The coefficients \( \alpha_0, \alpha_1, \ldots, \alpha_p \) can be determined by solving the linear system

\[
M(\Phi, Y) \alpha_\Phi = f(Y),
\]

where

\[
M(\Phi, Y) = \begin{bmatrix}
\phi_0(y^0) & \phi_1(y^0) & \cdots & \phi_p(y^0) \\
\phi_0(y^1) & \phi_1(y^1) & \cdots & \phi_p(y^1) \\
\vdots & \vdots & \ddots & \vdots \\
\phi_0(y^p) & \phi_1(y^p) & \cdots & \phi_p(y^p)
\end{bmatrix},
\]

\[
\alpha_\Phi = \begin{bmatrix}
\alpha_0 \\
\alpha_1 \\
\vdots \\
\alpha_p
\end{bmatrix}
\]

and

\[
f(Y) = \begin{bmatrix}
f(y^0) \\
f(y^1) \\
\vdots \\
f(y^p)
\end{bmatrix}.
\]

In order to make the above system have a unique solution, the matrix \( M(\Phi, Y) \) has to be nonsingular.
Definition 2.1. [4] The set $\mathcal{Y} = \{y^0, y^1, \cdots, y^p\} \subset \mathbb{R}^n$ is poised for polynomial interpolation in $\mathbb{R}^n$ if the corresponding matrix $M(\Phi, \mathcal{Y})$ is nonsingular for some basis $\Phi$ in $\mathcal{P}_d$.

In the following, we give some basic definitions which are appeared in several literatures, for example, see [4, 21, 29, 30].

Definition 2.2. Given a set of interpolation points $\mathcal{Y} = \{y^0, y^1, \cdots, y^p\} \subset \mathbb{R}^n$, a basis of $p_1 = p + 1$ polynomials $l_j(x)(j = 0, \cdots, p) \in \mathcal{P}_d$ is called a basis of Lagrange polynomials if

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

If $\mathcal{Y}$ is poised, then the Lagrange polynomials exist, are unique and have a number of useful properties. In particular, if $m(x)$ interpolates $f(x)$ at the points of the poised set $\mathcal{Y}$, then for all $x$,

$$m(x) = \sum_{j=0}^{p} f(y^j) l_j(x), \quad (2)$$

which is unique. It can also be shown that

$$\sum_{j=0}^{p} l_j(x) = 1, \quad \forall x \in \mathbb{R}^n. \quad (3)$$

For more details and other properties of Lagrange polynomials, see, for example, Conn, Scheinberg, and Vicente [4] and Sun etc. [30].

Further, we give the following concept of $\Lambda$-poisedness and some related lemmas in [4, 29].

Definition 2.3. [4] Let $\Lambda > 0$ and a set $\mathcal{B} \in \mathbb{R}^n$ be given. Let $\phi$ be the natural basis of monomials of $\mathcal{P}_d$. A poised set $\mathcal{Y} = \{y^0, y^1, \cdots, y^p\}$ is said to be $\Lambda$-poised in $\mathcal{B}$ if and only if for the basis of Lagrange polynomials associated with $\mathcal{Y}$, one has that

$$\Lambda \geq \max_{0 \leq i \leq p} \max_{x \in \mathcal{B}} |l_i(x)|.$$

Lemma 2.4. [29] Given a closed bounded domain $\mathcal{B}$, any initial interpolation set $\mathcal{Y} \subset \mathcal{B}$, and a constant $\Lambda > 1$. Consider the following procedure: find $j \in \{0, \cdots, p\}$ and a point $x \in \mathcal{B}$ such that $|l_j(x)| \geq \Lambda$ (if such a point exists), and replace $y^j$ by $x$ to obtain a new set $\mathcal{Y}$. Then this procedure terminates after a finite number of iterations with a model which is $\Lambda$-poised in $\mathcal{B}$.

Lemma 2.5. [4, 29] Given a ball $\mathcal{B}(x, \Delta) \triangleq \{y \in \mathbb{R}^n \mid \|y - x\| \leq \Delta\}$, a poised interpolation set $\mathcal{Y} \subset \mathcal{B}(x, \Delta)$, and its associated basis of Lagrange polynomials $\{l_j(y)\}_{j=0}^{p}$, there exist constants $\kappa_{ef} > 0$ and $\kappa_{eg} > 0$ such that, for any interpolating polynomial $m(x)$ of degree one or higher of the form (2) and any given point $y \in \mathcal{B}(x, \Delta)$, we have

$$\|f(y) - m(y)\| \leq \kappa_{ef} \sum_{j=0}^{p} \|y^j - y\|^2 |l_j(y)|$$

and

$$\|\nabla f(y) - \nabla m(x)\| \leq \kappa_{eg} \Lambda \Delta,$$
where
\[ \Lambda = \max_{j=0, \ldots, p} \max_{x \in B(x, \Delta)} |l_j(x)|. \]

2.2. A new self-correcting geometry process. To guarantee the poisedness of the interpolation set and accelerate the convergence, we make several modifications to the existing self-correcting geometry process. Now we first describe our new self-correcting geometry process, and then analyze this algorithm, in particular, justify our modifications.

Algorithm 2.1. A new self-correcting geometry process.

Step 0. Initialization:
The current iterate \( x_k \), the current interpolation set \( Y_k \), the current trust region radius \( \Delta_k > 0 \), the ratio \( \rho_k \), the switch value \( \delta_k \) for updating radius and the trial point \( x_{k}^{+} \) are given. Constants \( \beta > 0 \), \( 0 < \beta_2 < 1 < \beta_1 \) and \( \eta \in [0, 1) \) are also given.

Step 1. Successful iteration:
If \( \rho_k \geq \eta \), then define \( x_{k+1} = x_k \), \( \Delta_{k+1} = \beta_1 \Delta_k \) and \( Y_{k+1} = Y_k \cup \{ x_{k}^{+} \} \setminus \{ y^n \} \), where \( y^n \in \arg \max_{y \in Y_k} || y - x_{k}^{+} || \).

Step 2. Replace the comparatively bad interpolation points:
If \( \rho_k < \eta \), the set
\[ A_k \triangleq \{ y_j \in Y_k \mid || y_j - x_k || > \beta \Delta_k \} \]
is nonempty and
\[ B_k \triangleq \{ y_j \in Y_k \mid || l_j(x_k^+) || > \Lambda \} \]
is nonempty, then set \( x_{k+1} = x_k \) and \( Y_{k+1} = Y_k \cup \{ x_{k}^{+} \} \setminus \{ y^n \} \), where \( y^n \) is an index of any point in \( A_k \cup B_k \), for instance, such that
\[ y^n = \arg \max_{y \in A_k \cup B_k} || y_j - x_k ||^2 |l_j(x_k^+)|. \]
If \( \Delta_k \geq \Delta_c \), set \( \Delta_{k+1} = \beta_2 \Delta_k \); else set \( \Delta_{k+1} = \Delta_k \).

Step 3. Replace the distant interpolation points:
If \( \rho_k < \eta \), \( A_k \neq \emptyset \) and \( B_k = \emptyset \), then set \( x_{k+1} = x_k \) and \( Y_{k+1} = Y_k \cup \{ x_{k}^{+} \} \setminus \{ y^n \} \), where \( y^n \) is an index of any point in \( A_k \), for instance, such that
\[ y^n \in \arg \max_{y \in A_k} || y_j - x_k ||. \]
If \( \Delta_k \geq \Delta_c \), set \( \Delta_{k+1} = \beta_2 \Delta_k \); else set \( \Delta_{k+1} = \Delta_k \).

Step 4. Replace the badly-poised interpolation points:
If \( \rho_k < \eta \), \( A_k = \emptyset \) and \( B_k \neq \emptyset \), then set \( x_{k+1} = x_k \) and \( Y_{k+1} = Y_k \cup \{ x_k + s_k \} \setminus \{ y^n \} \), where \( y^n \) is an index of any point in \( B_k \), for instance, such that
\[ y^n \in \arg \max_{y \in B_k} |l_j(x_k^+)|. \]
If \( \Delta_k \geq \Delta_c \), set \( \Delta_{k+1} = \beta_2 \Delta_k \); else set \( \Delta_{k+1} = \Delta_k \).

Step 5. Unchange the interpolation set:
If \( \rho_k < \eta \) and \( A_k \cup B_k = \emptyset \), then set \( x_{k+1} = x_k \) and \( \Delta_{k+1} = \beta_2 \Delta_k \).

Now, we give some interpretations of this algorithm.
There are five possible cases in Algorithm 2.1, which is similar to Algorithm 2 in [29] where there are four possible cases. The first case occurs when the trial point \( x_{k}^{+} \) makes the objective function value sufficiently decrease. Then we set \( x_{k+1} = x_{k}^{+} \).
As for updating the interpolation set, our measure is that \( x_k^+ \) replaces \( y^r \) which is the farthest point from \( x_k^+ \), instead of the farthest point from \( x_k \) which is adopted in [13]. Case 1 is described in Step 1.

The second case occurs when the iteration is unsuccessful and the interpolation set contains points that are too far from the current trust region center (\( A_k \neq \emptyset \)) and points whose corresponding Lagrange polynomials have large absolute values at the trial point (\( B_k \neq \emptyset \)). In this case, we can choose any point from \( A_k \cup B_k \) to be replaced. For practical reason, we replace one point such that we can get the best improvement in the combined measure of both distance and poisedness criteria. Case 2 is described in Step 2.

The third case occurs when \( A_k \neq \emptyset \) and \( B_k = \emptyset \), which means that although \( Y_k \) is good in the sense of poisedness criterion, but there are some points that are too far away from the current trust region center. So, we only consider the replacement in the distance sense. For practical reason, we choose the farthest point \( y^r \). Case 3 is described in Step 3.

In the fourth case, if \( A_k = \emptyset \) but \( B_k \neq \emptyset \), i.e., all points are not distant from the current iterate \( x_k \), Step 4 is executed. In this case, we only replace the points that belong to \( B_k \), ignoring the effect of the distance. For practical reason, we choose the point that violates the poisedness criterion most seriously.

Finally, if the iteration is not successful and the set \( A_k \cup B_k = \emptyset \), then we execute Step 5 and only reduce the trust region radius. Obviously, one of the five cases must be used. We show that Step 3, Step 4 and Step 5 can be used only in a finite number of times until either a successful iteration occurs or the model becomes well poised.

Algorithm 2.1 is a variant and extension of Algorithm 2 in [29] and Algorithm 1 in [13], where \( A_k \) and \( B_k \), respectively, stand for the criteria of distance and the poisedness. According to the empty property of \( A_k \) and \( B_k \), we choose the replaced point depending on the distance criterion, the poisedness criterion or the combination of two criteria. In our algorithm the distance criterion is related to \( x_k^+ \) at successful iteration and \( x_k \) in unsuccessful case, respectively, while the strategy adopted in Algorithm 2 in [29] is only based on \( x_k^+ \). Numerical experiments show that our modification is more efficient.

As for updating the radius, in Algorithm 2 in [29], it is reduced only when the interpolation set \( Y_k \) is unchanged in the unsuccessful iterations. However, according to the numerical experiments, we find this strategy is not good. To accelerate convergence, we also shrink the radius when \( Y_k \) is changed in the unsuccessful iterations, if the radius is not too small. If the radius is too small, we reduce the radius only in Step 5 of Algorithm 2.1.

If the interpolation set is changed, we update all coefficients of the Lagrange polynomial. Then a new model can be expressed in (2).

**2.3. SCGWTR algorithm.** In this subsection we describe our algorithm which is based on the new self-correcting geometry process and the modified wedge trust region method. Our algorithm is called SCGWTR, in brief.

In general, the model is quadratic in trust region framework, written as

\[
\begin{align*}
\min_s & \quad m_k(x_k + s) = m_k(x_k) + s^T g_k + \frac{1}{2} s^T G_k s, \\
\text{s.t.} & \quad \|s\| \leq \Delta_k,
\end{align*}
\]

(7)
where \( \nabla m_k(x_k + s) = G_k s + g_k, \nabla m_k(x_k) = g_k, \nabla^2 m_k(x_k) = G_k \), and \( \Delta_k \) is the trust-region radius at the \( k \)-th iteration. Note that, \( G_k \) is a symmetric approximation to \( \nabla^2 f(x_k) \). In the derivative-free case, \( G_k \neq \nabla^2 f(x_k), g_k \neq \nabla f(x_k) \).

The wedge trust region method was proposed by Marazzi and Nocedal [12, 13]. There are two versions: the linear version and the quadratic version, which are classified by the degree of the interpolation models. They follow the idea that adds a wedge constraint into the trust region subproblem.

We define a “taboo region” \( T_C \) in \( \mathbb{R}^n \), in which the interpolation points are non-poised. We also define a “wedge”, which is a set \( W \) and that is designed to avoid those points too near \( T_C \). The description of \( T_C \) and \( W \) can be referred in Section 3 and 4 in [13]. The wedge constraint is added to the trust region subproblem, and we have

\[
\begin{align*}
\min_s & \quad m_k(x_k + s), \\
\text{s.t.} & \quad \|s\| \leq \Delta_k, \\
& \quad s \notin W_k.
\end{align*}
\]

As for solving the wedge trust region subproblem (9)–(11), we usually first solve the standard trust region subproblem without the wedge constraint and get a solution \( s_k^0 \) at the \( k \)-th iteration. If \( s_k^0 \) satisfies the wedge constraint, we set \( s_k = s_k^0 \) as the trial step. Otherwise, the wedge constraint is active. By rotating \( s_k \), we find a vector satisfying the wedge constraint. Then we set the trial point \( x_k^+ = x_k + s_k \).

Now we state our new algorithm as follows.

**Algorithm 2.2. The Wedge Trust Region Method with Self-Correcting Geometry**

**Step 0. Initialization:**
Given an initial point \( x_0 \), an initial radius \( \Delta_0 \in (0, +\infty) \), an initial tolerance \( \epsilon_0 \), an initial interpolation set \( \mathcal{Y}_0 \) such that \( \mathcal{Y}_0 \) is poised and the initial interpolation model function \( m_0(x) \) corresponding to \( \mathcal{Y}_0 \). Constants \( \eta \in [0, 1], \mu \in (0, 1), \theta > 0, 0 < \beta_2 < 1 < \beta_1 \) and \( \Lambda > 1 \). Set \( k = 0 \).

**Step 1. Criticality test:**

**Step 1a.** Set \( i = 0 \) and \( \hat{m}_i = m_k \).

**Step 1b.** If \( \| \nabla \hat{m}_i(x_k) \| < \epsilon_i \), set \( \epsilon_{i+1} = \mu \| \nabla \hat{m}_i(x_k) \| \), compute a poised model \( \hat{m}_{i+1} \) in \( B(x_k, \epsilon_{i+1}) \), set \( i := i + 1 \), and return to Step 1b.

**Step 1c.** Set \( m_k = \hat{m}_i, \Delta_k+1 = \theta \| \nabla m_k(x_k) \| \), and set \( v_i = x_k \) if a new model has been computed.

**Step 2. Compute the replaced point:**
Choose the point that is the farthest from the current iterate as the replaced point \( y^r \), i.e.,

\[ y^r = \arg \max_{y \in \mathcal{Y}_k} \| y - x_k \|. \]

**Step 3. Solve the wedge trust region subproblem:**
Solve the wedge trust region subproblem (9)–(11) for getting \( s_k \), and set the trial point \( x_k^+ \equiv x_k + s_k \).

**Step 4. Update the iteration and interpolation set:**
Compute

\[ \rho_k \equiv \frac{f(x_k) - f(x_k^+)}{m_k(x_k) - m_k(x_k^+)^2}. \]

Use Algorithm 2.1 to get \( \mathcal{Y}_{k+1}, x_{k+1} \) and \( \Delta_{k+1} \).
Step 5. Update the model and Lagrange polynomials:
If $y_{k+1} \neq y_k$, recompute the interpolation model $m_{k+1}(x)$ by using the Lagrange polynomials $l_j(x)$ ($j = 1, \ldots, p$) associated with $y_{k+1}$. Set $k := k + 1$ and go to Step 1.

3. Global Convergence. In this section we prove the convergence results of the algorithm SCGWTR which say that Algorithm 2.2 produces a subsequence which converges to the stationary point. Our convergence analysis is based on the convergence theory of basic trust region methods in [2, 33] and is an extension and modification of the convergence analysis of algorithm WEDGE [13] and algorithm SCGDF [29]. We first give some assumptions.

Assumptions:
A1 The objective function $f$ is continuously differentiable in an open set $\Omega$ containing all iterates generated by the algorithm and its gradient $\nabla f$ is Lipschitz continuous in $\Omega$ with constant $L$.
A2 There exists a constant $\kappa_{\text{low}}$ such that $f(x) \geq \kappa_{\text{low}}$ for every $x \in \Omega$.
A3 There exists a constant $\kappa_H \geq L$ such that $\|G_k\| \leq \kappa_H$ for every $k \geq 0$.

Note that the above assumptions are standard assumptions for the derivative-free optimization (see [4, 13, 29]. The assumption A1 means the existence of the first derivatives of objective functions, not the possibility of their calculations. The assumption A3 implies that the Hessian of the model function has to remain bounded in norm.

Lemma 3.1. At the $k$-th iteration, the solution of the wedge trust region subproblem (9)-(11) satisfies
\[ m_k(x_k) - m_k(x_k + s_k) \geq \kappa_c \|g_k\| \min \left\{ \Delta_k, \frac{\|g_k\|}{\|G_k\|} \right\}, \]

where $\kappa_c \in (0, 1)$ is a constant.

Proof. From Lemma 6.1.3 in [33], we know that if $s_k^e$ is the exact solution of (9)-(10), then
\[ m_k(x_k) - m_k(x_k + s_k^e) \geq \frac{1}{2} \|g_k\| \min \left\{ \Delta_k, \frac{\|g_k\|}{\|G_k\|} \right\}. \]

Since, in the process of solving the subproblem of the wedge trust region methods, the model function is sufficiently descent, i.e.,
\[ m_k(x_k) - m_k(x_k^+) \geq \frac{1}{2} [m_k(x_k) - m_k(x_k + s_k^e)], \]
we can deduce that there exists a constant $\kappa_c \in (0, 1)$ such that
\[ m_k(x_k) - m_k(x_k + s_k) \geq \frac{1}{2} [m_k(x_k) - m_k(x_k + s_k^e)] \geq \kappa_c \|g_k\| \min \left\{ \Delta_k, \frac{\|g_k\|}{\|G_k\|} \right\}. \]

Now we give the self-correcting property of Algorithm 2.1.

Lemma 3.2. Suppose that assumptions A1 and A3 hold and that $m_k$ is a quadratic model. Then, for any constant $\Lambda > 1$, if the $k$-th iteration is unsuccessful,
\[ A_k = \emptyset \]

\[ \Delta_k \leq \min \left[ \frac{1}{\kappa_H}, \frac{(1 - \eta)\kappa_c}{2\kappa_c f(\beta + 1)^2(p\Lambda + 1)} \right] \left\| g_k \right\| \overset{\Delta}{=} \kappa_\Lambda \left\| g_k \right\|, \] (15)

then
\[ B_k \neq \emptyset. \]

Proof. The proof of this lemma is similar to Lemma 5.2 in [29], and we omit it. \[ \square \]

This lemma means that the self-correcting property introduced by [29] also holds in our modified algorithm, provided the trust region radius is small enough compared to the model’s gradient, and that all interpolation points are contained in the trust region. Then every unsuccessful iteration must result in an improvement of the interpolation set geometry.

Next, we will verify that, in our case, the bound \( \Delta_k \) cannot become arbitrarily small when the iterate is far away from a critical point. This lemma is an extension of Lemma 5.3 in [29].

**Lemma 3.3.** Suppose that assumptions A1 and A3 hold. Suppose also that, for some \( k_0 \geq 0 \) and all \( k \geq k_0 \), the model is quadratic and
\[ \left\| g_k \right\| \geq \kappa_g \] (16)
for some \( \kappa_g > 0 \). Then there exists a constant \( \kappa_\Delta > 0 \) such that, for all \( k \geq k_0 \),
\[ \Delta_k \geq \kappa_\Delta. \] (17)

Proof. Assume that, for some \( k \geq 0 \),
\[ \Delta_k < \min(\kappa_\Lambda \kappa_g, \mu \kappa_g, \Delta_c). \] (18)

If the \( k \)-th iteration is successful, i.e., \( \rho_k \geq \eta \), then
\[ \Delta_{k+1} \geq \Delta_k. \]
Otherwise, \( \rho_k < \eta \), and there are three cases that may occur.

The first case is when \( A_k \neq \emptyset \) and \( B_k \neq \emptyset \), Step 2 of Algorithm 2.1 is executed. Observe that (18) ensures \( \Delta_k < \Delta_c \), therefore \( \Delta_{k+1} = \Delta_k \).

The second case is when \( A_k \neq \emptyset \) and \( B_k = \emptyset \). If \( i > 0 \), then (16) and (18) ensure that
\[ \Delta_k < \mu \left\| g_{k_i} \right\| = \epsilon_i, \] (19)
where \( k_i \) is the index of the last iteration before \( k \) where a new \( \Lambda \)-poised model has been recomputed in the criticality test. Therefore, Step 3 of Algorithm 2.1 is executed. Together with \( \Delta_k < \Delta_c \), we also have \( \Delta_{k+1} = \Delta_k \).

The third case is conducted when \( A_k = \emptyset \). Under the condition (18), Lemma 3.2 can infer that \( B_k \neq \emptyset \). Since (18) and (19) hold in this case, Step 4 of Algorithm 2.1 is executed and \( \Delta_{k+1} = \Delta_k \).

As a consequence, the trust region radius can be decreased only if
\[ \Delta_k \geq \min(\kappa_\Lambda \kappa_g, \mu \kappa_g, \Delta_c), \]
and Algorithm 2.1 implies (17) with
\[ \kappa_\Delta = \min[\Delta_0, \gamma_1 \min(\kappa_\Lambda \kappa_g, \mu \kappa_g, \Delta_c)]. \]
\[ \square \]

In the following, we consider the convergence analysis in the case where the number of successful iterations is finite.
Lemma 3.4. Suppose that assumptions A1 and A3 hold, the model is quadratic for all $k$ sufficiently large and there is a finite number of successful iterations. Then
\[
\liminf_{k \to \infty} \|g_k\| = 0. \tag{20}
\]

Proof. First, since every iteration is eventually unsuccessful, then for some $x_*$ and all $k$ sufficiently large,
\[
x_k = x_*.
\]

To deduce a contradiction, we assume that for some $\kappa_g > 0$ and all $k$ sufficiently large,
\[
\|g_k\| \geq \kappa_g.
\]

We can infer from Lemma 3.3 that
\[
\Delta_k > \kappa_\Delta > 0
\]
for all $k$ sufficiently large.

Since the number of successful iterations is finite, eventually all iterations go through Step 2, Step 3, Step 4 or Step 5 in Algorithm 2.1. Then, the sequence $\{\Delta_k\}$ is nonincreasing and bounded below and therefore convergent. So, let
\[
\Delta_\infty \triangleq \lim_{k \to \infty} \Delta_k \geq \kappa_\Delta,
\]
and we have that the unsuccessful iterations of Step 5 in Algorithm 2.1 cannot happen infinitely many times because $\Delta_k$ is bounded below by $\Delta_\infty$ and $\beta_2 < 1$. While Step 2, Step 3 and Step 4 in Algorithm 2.1 cannot happen infinitely many times too for a certain lower bound $\Delta_c$ (see Algorithm 2.1). Thus $\Delta_k = \Delta_\infty$ for all $k$ sufficiently large, and all iterations eventually execute Step 5 in Algorithm 2.1.

Note that during the iteration of Step 4 in Algorithm 2.1, the new trial point replaces some interpolation points in the set $B_k$. Then, the trial point $x_k^+$ replaces a previous interpolation point $y_l$ with $|l_j(x_k^+)| \geq \Lambda$. But this is impossible from Lemma 2.4, where the exchange procedure will terminate after a finite number of iterations with a $\Lambda$-poised model. So the contradiction is obtained.

Next, we further consider the case of infinitely many successful iterations.

Lemma 3.5. Suppose that assumptions A1 - A3 hold, the model is quadratic for all $k$ sufficiently large and the number of successful iterations is infinite. Then (20) also holds.

Proof. Assume that the lemma is not true, i.e., there exists some $\kappa_g > 0$ such that (16) holds for all sufficiently large $k$. Then we have from Lemma 3.3 that (17) holds for all $k$, including all successful iterations with $k$ large enough. However, from (13), we have that
\[
f(x_k) - f(x_{k+1}) \geq \eta(m_k(x_k) - m_k(x_k^+))
\]
\[
\geq \eta \kappa_c \|g_k\| \min \left\{ \Delta_k, \frac{\|g_k\|}{\|G_k\|} \right\}
\]
\[
\geq \eta \kappa_c \kappa_g \min \left[ \frac{\kappa_g}{\kappa_H}, \kappa_\Delta \right]
\]
\[
\Delta \geq \kappa_d > 0.
\]
Since, by assumption, there are infinitely many successful iterations, we obtain that
\[
\lim_{k \to \infty} f(x_k) = f(x_0) - \sum_{i=1, i \in S}^\infty \kappa_d = -\infty,
\]
which contradicts assumption A2, where S is a set of successful iterations. Therefore the lemma holds.

**Lemma 3.6.** [2, 29] Suppose that assumptions A1 and A3 hold, then
\[
|f(x_k^+) - m_k(x_k^+)| \leq \|\nabla f(x_k) - g_k\| \Delta_k + \kappa_H \Delta_k^2.
\]

**Lemma 3.7.** [2, 29] Suppose that assumptions A1 and A3 hold, \(g_k \neq 0\),
\[
\|\nabla f(x_k) - g_k\| \leq \frac{1}{2} \kappa_c (1 - \eta) \|g_k\|,
\]
and
\[
\Delta_k \leq \frac{\kappa_c}{2 \kappa_H} (1 - \eta) \|g_k\|,
\]
then the k-th iteration is successful.

So, by use of the above lemmas and similar proof in Theorem 5.8 in [29], we can obtain the main theorem.

**Theorem 3.8.** Suppose that assumptions A1 - A3 hold and that the model is quadratic for all k sufficiently large, then
\[
\lim \inf_{k \to \infty} \|\nabla f(x_k)\| = 0.
\] (21)

4. **Numerical Experiments.** In this section we report the numerical results. We compare our algorithm with the wedge trust region method in [13] and the self-correcting geometry method in [29]. All the test problems come from Moré etc. [14]. The names of test problems are listed in Table 1. The details of numerical results for three algorithms are listed in Table 2, where WEDGE, SCGDFO and SCGWTR stand for the wedge trust region method, the self-correcting geometry method and our method, respectively. In Table 2, P, n, Iter, NF, F, and Time, respectively, stand for the problem number, the dimension of the problem, the number of iteration, the number of function calculation, the final function value and the cpu-time.

All programs are written in Matlab 7.1 with double precision where the standard trust region subproblem is solved by Moré and Sorensen (1983) [15]. The parameters are chosen as follows: \(\gamma = 0.4\), \(\epsilon = 2.2 \times 10^{-16}\), \(\Delta_0 = 1\), \(\Delta_c = 0\), \(\eta = 0\), \(\beta_1 = 2\), \(\beta_2 = 0.5\), \(\Lambda = 1.0 \times 10^{10}\). The maximum times of iterations and function calculations are set as 8000, beyond which we deem the algorithm failed, which is marked by * in the table.

The selection of the set of initial interpolation points \(Y_0 = \{y^0(= x_0), y^1, \cdots, y^p\}\) can be seen in (22), referring to UOBYQA [24], where \(p = \frac{1}{2}(n + 1)(n + 2) - 1\),

\[
\begin{align*}
y^0 &= x_0, \\
y^{2i-1} &= x_0 + \Delta_0 e_i, \\
y^{2i} &= \begin{cases} x_0 - \Delta_0 e_i, & \text{if } f(y^{2i-1}) \geq f(x_0), \\
x_0 + 2\Delta_0 e_i, & \text{otherwise}, \end{cases} i = 1, 2, \cdots, n, \\
y^{(u,v)} &= x_0 + \frac{1}{2} \Delta_0 (e_u + e_v), & 1 \leq u < v \leq n,
\end{align*}
\] (22)
where \( i(u, v) \equiv 2u + u + \frac{1}{2}(v - 1)(v - 2) \), and \( \Delta_0 > 0 \) is the initial trust region radius.

\[
\begin{array}{|c|c|}
\hline
P & Name \\
1 & Helical valley function \\
3 & Gaussian function \\
5 & Box 3-dimensional function \\
7 & Watson function \\
9 & Penalty II function \\
11 & Brown and Dennis function \\
13 & Trigonometric function \\
15 & Extended powell function \\
17 & Wood function \\
19 & Scaled cube function\((c = 100)\) \\
21 & Scaled cube function\((c = 10^6)\) \\
\hline
\end{array}
\]

From the numerical results in Table 2, we can find that, for 21 test problems and 29 cases, Algorithm WEDGE is failed for three cases, Algorithm SCGDF0 is failed for six cases, but our algorithm SCGWTR is all successful. In addition, in 29 cases, our algorithm is the best for 21 cases. So this table shows that our method is competitive with the other two methods.

According to Dolan and Moré [6], we draw the performance profiles of the three algorithms. From Figure 1, Figure 2 and Figure 3, we can also conclude that our method works more efficiently.

5. Conclusions. In this paper we propose a new self-correcting geometry process and give a self-correcting geometry wedge trust-region method for derivative-free optimization problems. We describe the motivation and details of our algorithm. The global convergence of our algorithm is established. The limited numerical experiments demonstrate that the new method is effective.

Acknowledgments. The authors are grateful to the editors and the anonymous referees for many helpful suggestions.
Figure 1. Iteration performance profiles

Figure 2. Function performance profiles

Figure 3. Time performance profiles
### Table 2. Comparison SCGWTR with WEDGE and SCGDFO

| n   | WEDGE | SCGWTR | SCGDFO |
|-----|-------|--------|--------|
|     | Iter  | NF     | F      | Time  |
|     |       |        |        |       |
| 1   | 1     | 3      | 66     | 78    |
|     | 9.4e-03 | 0.27   | 1.1e-03 | 4.2e-34 |
| 2   | 127   | 139    | 4.2e-32 | 9.4e-01 |
|     | 9.4e-01 | 0.27   | 1.1e-03 | 4.2e-34 |
| 3   | 88    | 100    | 2.2e-34 | 7.8e-02 |
|     | 9.4e-01 | 0.27   | 1.1e-03 | 4.2e-34 |
| 4   | 2      | 6      | 818    | 851   |
|     | 9.4e-01 | 0.27   | 1.1e-03 | 4.2e-34 |
| 5   | 5      | 3      | 44     | 56    |
|     | 9.4e-01 | 0.27   | 1.1e-03 | 4.2e-34 |
| 6   | 3      | 3      | 457    | 647   |
|     | 9.4e-01 | 0.27   | 1.1e-03 | 4.2e-34 |
| 7   | 6      | 6      | 981    | 1014  |
|     | 9.4e-01 | 0.27   | 1.1e-03 | 4.2e-34 |
| 8   | 6      | 6      | 239    | 272   |
|     | 9.4e-01 | 0.27   | 1.1e-03 | 4.2e-34 |
| 9   | 6      | 3      | 4919   | 4937  |
|     | 9.4e-01 | 0.27   | 1.1e-03 | 4.2e-34 |
| 10  | 2      | 2      | 55     | 73    |
|     | 9.4e-01 | 0.27   | 1.1e-03 | 4.2e-34 |
REFERENCES

[1] P. G. Ciarlet and P. A. Raviart, General Lagrange and Hermite interpolation in $R^n$ with applications to finite element methods, *Archive for Rational Mechanics and Analysis*, 46 (1972), 178–199.

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

[3] A. R. Conn, K. Scheinberg and Ph. L. Toint, A derivative free optimization algorithm in practice, *In the proceeding of the 7th AIAA/USAF/NASA/ISSMO Symposium on Multidisciplinary Analysis and Optimization*, St. Louis, MO, USA, September 1998.

[4] A. R. Conn, K. Scheinberg and L. N. Vicente, *Introduction to Derivative-Free Optimization*, MPS-SIAM Series on Optimization, SIAM, Philadelphia, PA, USA, 2008.

[5] Y. H. Dai, W. W. Hager, K. Schittkowski and H. C. Zhang, The cyclic Barzilai-Borwein method for unconstrained optimization, *IMA J. Numerical Analysis*, 26 (2006), 604–627.

[6] E. D. Dolan and J. J. Moré, Benchmarking optimization software with performance profiles, *Mathematical Programming*, 91 (2002), 201–203.

[7] K. R. Fowler, J. P. Reese, C. E. Kees, J. E. Dennis, C. T. Kelley, C. Audet, A. J. Booker, G. Couture, R. W. Darwin, M. W. Farthing, D. E. Finkel, J. M. Gablonsky, G. Gray and T. G. Kolda, Comparison of derivative-free optimization methods for groundwater supply and hydraulic capture community problems, *Advances in Water Resources*, 31 (2008), 743–757.

[8] S. Gratton, Ph. L. Toint and A. Tröltzsch, An active-set trust-region method for derivative-free nonlinear bound-constrained optimization, *Optimization Methods and Software*, 26 (2011), 873–894.

[9] G. Gray, T. Kolda, K. Sale and M. Young, Optimizing an empirical scoring function for transmembrane protein structure determination, *INFORMS Journal on Computing*, 16 (2004), 406–418.

[10] R. Hooke and T. A. Jeeves, Direct search solution of numerical and statistical problems, *Journal of the Association for Computing Machinery*, 8 (1961), 212–219.

[11] X. W. Liu and Y. Yuan, A robust algorithm for optimization with general equality and inequality constraints, *SIAM J. Scientific Computing*, 22 (2000), 517–534.

[12] M. Marazzi, *Nonlinear Optimization with and without Derivatives*, PhD thesis, Department of Industrial Engineering and Management Science, Northwestern University, Illinois, USA, 2001.

[13] M. Marazzi and J. Nocedal, Wedge trust region methods for derivative free optimization, *Mathematical Programming*, 91 (2002), 289–305.

[14] J. J. Moré, B. S. Garbow and K. E. Hillstrom, Testing unconstrained optimization software, *ACM Transactions on Mathematical Software*, 4 (1981), 136–140.

[15] J. J. Moré and D. C. Sorensen, Computing a trust region step, *SIAM Journal on Scientific and Statistical Computing*, 4 (1983), 553–572.

[16] J. A. Nelder and R. Mead, A simplex method for function minimization, *The Computer Journal*, 7 (1965), 308–313.

[17] J. Nocedal and S. J. Wright, *Numerical Optimization*, Springer, New York, (1999).

[18] R. Oeuvray, *Trust-Region Methods Based on Radial Basis Functions with Application to Biomedical Imaging*, PhD thesis, Institut de Mathématiques, École Polytechnique Fédérale de Lausanne, Lausanne, Switzerland, 2005.

[19] M. J. D. Powell, A new algorithm for unconstrained optimization, *In Nonlinear Programming* (eds. J. B. Rosen, O. L. Mangasarian and K. Ritter), Academic Press, New York, (1970), 31–65.

[20] M. J. D. Powell, On the global convergence of trust region algorithms for unconstrained optimization, *Mathematical Programming*, 29 (1984), 297–303.

[21] M. J. D. Powell, A direct search optimization method that models the objective by quadratic interpolation, *In presentation at the 5th Stockholm Optimization Days*, Stockholm, Sweden, 1994.

[22] M. J. D. Powell, A quadratic model trust region method for unconstrained minimization without derivatives, *In Proceedings of the International Conference on Nonlinear Programming and Variational Inequalities*, Hong Kong, 1998.

[23] M. J. D. Powell, *On the Lagrange functions of quadratic models that are defined by interpolation*, *Optimization Methods and Software*, 16 (2001), 289–309.
[24] M. J. D. Powell, UOBYQA: Unconstrained optimization by quadratic approximation, *Mathematical Programming*, **92** (2002), 555–582.

[25] M. J. D. Powell, Least Frobenius norm updating of quadratic models that satisfy interpolation conditions, *Mathematical Programming*, **100** (2004), 183–215.

[26] M. J. D. Powell, The NEWUOA software for unconstrained optimization without derivatives, In *Large-Scale Nonlinear Optimization* (eds. P. Pardalos, G. Pillo and M. Roma), Springer, New York, (2006), 255–297.

[27] M. J. D. Powell, On nonlinear optimization since 1959, In *The Birth of Numerical Analysis* (eds. A. Bultheel and R. Cools), World Scientific, (2010), 141–160.

[28] M. J. D. Powell and Y. Yuan, A trust region algorithm for equality constrained optimization, *Mathematical Programming*, **49** (1991), 189–211.

[29] 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.

[30] W. Sun, Q. K. Du and J. R. Chen, *Computational Methods*, Science Press, Beijing, (2007).

[31] W. Sun, J. Yuan and Y. Yuan, Conic trust region method for linearly constrained optimization, *Journal of Computational Mathematics*, **21** (2003), 295–304.

[32] W. Sun and Y. Yuan, A conic trust-region method for nonlinearly constrained optimization, *Annals of Operations Research*, **103** (2001), 175–191.

[33] W. Sun and Y. Yuan, *Optimization Theory and Methods: Nonlinear Programming*, Springer Optimization and Its Applications, Volume 1. Springer, New York, 2006.

[34] S. M. Wild, R. G. Regis and C. A. Shoemaker, ORBIT: optimization by radial basis function interpolation in trust-regions, *SIAM Journal on Scientific Computing*, **30** (2008), 3197–3219.

[35] D. Winfield, *Function and Functional Optimization by Interpolation in Data Tables*, PhD thesis, Harvard University, Cambridge, USA, 1969.

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

[37] D. Xue and W. Sun, On convergence analysis of a derivative-free trust region algorithm for constrained optimization with separable structure, *Science China Mathematics*, **57** (2014), 1287–1302.

[38] Y. Yuan, On a subproblem of trust region algorithms for constrained optimization, *Mathematical Programming*, **47** (1990), 53–63.

[39] H. C. Zhang, A. R. Conn and K. Scheinberg, A derivative-free algorithm for least-squares minimization, *SIAM J. Optimization*, **20** (2010), 3555–3576.

[40] L. Zhao and W. Sun, Nonmonotone retrospective conic trust region method for unconstrained optimization, *Numerical Algebra, Control and Optimization*, **3** (2013), 309–324.

Received December 2014; 1st revision February 2015; final revision April 2015.

E-mail address: burgers@yeah.net
E-mail address: wysun@njnu.edu.cn
E-mail address: raimundo.sampaio@pucpr.br
E-mail address: jin@ufpr.br