Manuscript submitted to
AIMS' Journals
Volume **X**, Number **0X**, XX **200X**

Website: http://AIMsciences.org

pp. **X–XX**

SURVEY OF TRUST-REGION DERIVATIVE FREE OPTIMIZATION METHODS

BÜLENT KARASÖZEN

Middle East Technical University
Department of Mathematics & Institute of Applied Mathematics
06531 Ankara, Turkey

Abstract. In this survey article we give the basic description of the interpolation based derivative free optimization methods and their variants. We review the recent contributions dealing with the maintaining the geometry of the interpolation set, the management of the trust region radius and the stopping criteria. Derivative free algorithms developed for problems with some structure like for partially separable functions are discussed. Two different versions of derivative free algorithms are applied for the optimization of the configuration of the geometry of a stirrer. Numerical results are presented to show the applicability of the algorithms to practical problems.

1. **Introduction.** Derivative free optimization(DFO) methods are designed for solving nonlinear optimization without constraints where the derivatives of the objective function are not available. We consider formally the problem

$$\min_{x \in \mathbb{R}^n} f(x)$$

where f is a smooth nonlinear objective function from $\mathbb{R}^n$ into $\mathbb{R}$ and is bounded below. We assume that the gradient $\nabla f(x)$ and the Hessian $\nabla^2 f(x)$ can not be computed for any x.

There is a high demand from practitioners for such algorithms because this kind of problems occur relatively frequently in the industry. In applications either the evaluation of the objective function $f(x)$ is very difficult or expensive, or the derivatives of f are not available. The last situation occurs when the computation of $f(x)$ at a given point x results from some physical, chemical or econometrical experiment or measurement, or is a result of large and expensive computer simulation for which the source code is either not available or unmodifiable, which can be considered as a black box. In practice the value of $f(x)$ is contaminated with noise or may be non-smooth; but we don't consider these cases in our presentation.

There are mainly four classes of derivative-free optimization methods. The first class of DFO algorithms are the direct search or pattern search methods which are based on the exploration of the variable space by using sample points from a predefined class geometric

2000 *Mathematics Subject Classification.* Primary: 90C56 ; Secondary: 65D05.

Key words and phrases. derivative-free optimization, multi-variate interpolation, trust-region methods, partially separable functions.

pattern and use either the Melder-Need simplex algorithm or parallel direct search algorithm. They do not exploit the inherit smoothness of the objective function and require therefore a very large number of function evaluations. They can be useful for non-smooth problems. A comprehensive survey of these methods can be found in [16]. The second class of DFO's are line search methods which consists of a sequence of $n+1$ one-dimensional searches introduced by Powell [21]. The combination of finite difference techniques coupled with quasi-Newton method constitutes the third class of the algorithms [22]. The last class of the methods are based on modeling the objective function by multivariate interpolation in combination with the trust-region techniques. These methods were introduced by Winfried [30], [31] and by Powell [23], [25]. In this survey we consider this class of DFO algorithms. The main idea of these methods is building a polynomial model, interpolating the objective function at all points at which its value is known. The model is then minimized over the trust region and a new point, is computed. The objective function evaluated at this new point thus possibly enlarging the interpolation set. This newly computed point is checked as to whether the objective function is improved and the whole process repeated until convergence is achieved.

The main criterion for assessment and comparison of DFO algorithms is the number of function evaluations they require for minimization. The use of all DFO algorithms mentioned above are limited to amoderate number of variables, e.g., 20 variables. In the direct search methods, the number of grid points which are required to fill the whole sample space could be very large. Similarly, for interpolation methods a total of $(n+1)(n+2)/2$ known function values would be necessary for a full quadratic model.

2. **Description of the DFO algorithm.**

DFO algorithms belong to the class of trust-region methods which are iterative and built around the current iterate as a cheaper model of the objective function which is easier to minimize than the objective function. At the kth step the quadratic model within the trust-region $\mathcal{B}_k$,

$$\mathcal{B}_k = \{x_k + s : s \in \mathbb{R}^n \text{ and } ||s|| \le \Delta_k\},$$

with the trust-region radius Δ_k may be given as

$$m_k(x_k+s) = f(x_k) + \langle g(x_k)\,,\, s\rangle + \frac{1}{2}\langle s\,,\, H(x_k)s\rangle \tag{1}$$

for some $g \in \mathbb{R}^n$ and some symmetric $n \times n$ matrix H, where $\langle \cdot\,,\,\cdot\rangle$ denotes the inner product. The vector g and the matrix H do not necessarily correspond to the first and second derivatives of the objective function f. They are determined by requiring that the model (1) interpolates the function f at a set $Y = \{y_i\}$ of points containing the current iterate x_k

$$f(y_i) = m_k(y_i) \quad \text{for all } y_i \in Y. \tag{2}$$

Here, Y denotes the set of interpolation points, which is a subset of the set of points at which the values of f is known, including the current iterate. Building the full quadratic model (1) requires the determination of $f(x_k)$, the components of the vector g_k and the entries of the matrix H_k; so that the cardinality of Y must be equal to

$$p = \frac{1}{2}(n+1)(n+2).$$

However if $n > 1$, the condition (2) is not sufficient for the existence and uniqueness of the interpolant and to guarantee the good quality of the model. Geometric conditions (known as poisedness) on the set Y are required to ensure the the existence and uniqueness of the interpolant, i.e., that the points of Y do not collapse into a lower dimensional subset or lie on a quadratic curve. If $\{\phi_i(\cdot)\}_{i=1}^p$ denotes a basis of the linear space of $n-$dimensional quadratics, the model (1) and the interpolation conditions (2) can be written as

$$m_k(x) = \sum_{i=1}^p \alpha_i\phi_i(x), \qquad \sum_{i=1}^p \alpha_i\phi_i(y) = f(y) \quad \text{for all } y \in Y$$

and some scalars $\alpha_i, i = 1, \cdots, p$. Then the set $Y = \{y_1, \cdots, y_p\}$ is poised if the matrix

$$M = \delta(Y) = \det \begin{pmatrix} \phi_1(y_1) & \cdots & \phi_1(y_p) \\ \vdots & & \vdots \\ \phi_p(y_1) & \cdots & \phi_p(y_p) \end{pmatrix} \tag{3}$$

is non-singular, i.e., $\delta(Y) = \det(\mathrm{M})$ is non-zero [2, 6, 7]. The quality of the model (1) around the current iterate x_k depends on the geometry of the interpolation set. Examples of non-poised sets for quadratic interpolation are: 6 points on a line, a circle or a quadratic curve. From practical point of view, it is important to have quality a measure for this. The interpolation set is called well-poised if $|\delta Y| \geq \varepsilon$ for some threshold $\varepsilon > 0$.

After building the model $m_k(x_k + s)$ around the current iterate x_k, the step s_k is computed by minimizing the model m_k over the trust-region. If the ratio of the achieved reduction in the objective function versus the predicted reduction in model

$$\rho_k = \frac{f(x_k) - f(x_k + s_k)}{m_k(x_k) - m_k(x_k + s_k)} \tag{4}$$

is sufficiently positive, the iteration is successful; as the next iteration point, $x_{k+1} = x_k + s_k$ will be taken and the trust-region radius Δ_k is enlarged. If ρ_k is not sufficiently positive, then the iteration was not successful; the current x_k will be kept and the trust-region radius is reduced.

The general form of the DFO algorithm can be given as follows with the given constants [8]

$$0 < \eta_0 \leq \eta_1 < 1, \quad \text{and} \quad 0 \leq \gamma_1 < 1 \leq \gamma_2.$$

- **Step 1: Initialization**
  For a given x_s and $f(x_s)$ choose an initial interpolation set Y. Determine $x_0 \in Y$ such that $f(x_0) = \min_{y_i \in Y} f(y_i)$. Choose an initial trust region radius $\Delta_0 > 0$. Set $k = 0$.
- **Step 2: Model building**
  Using the poised interpolation set Y, build the model $m_k(x_k + s)$.
- **Minimization of the model within the trust-region**
  Compute the point x_k^+ such that

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

  Compute $f(x_k^+)$ and the ratio

$$\rho_k = \frac{f(x_k) - f(x_k^+)}{m_k(x_k) - m_k(x_k^+)}.$$

- **Step 3: Update the interpolation set**
  - *Successful step:*
    If $\rho_k \geq \eta_1$, include x_k^+ in Y by dropping one of the existing interpolation points.
  - *Unsuccessful step:*
    If $\rho_k < \eta_1$ and Y is inadequate in $x \in \mathcal{B}_k$, improve the geometry.
- **Step 4:Update the trust-region radius**
  - *Successful step:*
    If $\rho_k \geq \eta_1$, then set

$$\Delta_{k+1} \in [\Delta_k, \gamma_2 \Delta_k].$$

  - *Unsuccessful step:*
    If $\rho_k < \eta_1$ and Y is adequate in $\mathcal{B}_k$, then set

$$\Delta_{k+1} \in [\gamma_0 \Delta_k, \gamma_1 \Delta_k].$$

  - Otherwise, set $\Delta_{k+1} = \Delta_k$.

- **Step 5: Update the current iterate**
  Determine $\widehat{x}_k$ such that
  $$m_k(\widehat{x}_k) = \min_{y_i \in Y, y_i \neq x_k} m_k(y_i).$$
  Then, if
  $$\widehat{\rho}_k = \frac{f(x_k) - f(\widehat{x}_k)}{m_k(x_k) - m_k(x_k^+)} \geq \eta_0,$$
  set $x_{k+1} = \widehat{x}_k$. Otherwise $x_{k+1} = x_k$. Increment k by one and go to Step 1.

There are three particular issues with the implementation of this method. The first one is the acceptance of a new iterate. In the standard trust-region method, the new point produced by the algorithm is accepted as soon as $f(x_k^+)$ is sufficiently much smaller than $f(x_k)$, the current objective value. In case of DFO algorithms this situation is more complex, because to include the x_k^+ in the set of interpolation points Y we have to remove another point from Y. Only in the early stages of the iterations, where the models are incomplete (models that are less than full quadratic or linear models) all new points x_k^+ are included. Otherwise, one has to take care of the geometry of Y; the new iterate should not deteriorate the geometry of Y and the new interpolation set remain "as poised as possible". There is no guarantee that the quality of geometry and poisedness remains acceptable after the inclusion of the new point.

The interpolation set must be maintained in each iteration either by including a new iterate into the the interpolation set or by improving the model. In order to keep the interpolation set poised at each iteration, error estimates for the function and its derivatives are needed. In [10] using Taylor type error bounds between the model $m(x)$ and the objective function $f(x)$ for multivariate interpolation, the authors derive practical measures defining poisedness. The error in the gradient can be estimated as

$$||\nabla m(x) - \nabla f(x)|| \leq \frac{1}{(d+1)!} G \Lambda_Y \sum_{i=0}^{p} ||y_i - x||^{d+1}$$

for $x \in Y$. Here, d denotes the dimension of the interpolating polynomial, G depends only on the function f and Λ_Y is a constant depending on the interpolation set Y. For the convergence of the DFO methods, the error in gradient of should go to zero when the distances between all y_i and x go to zero and Λ_Y has to remain uniformly bounded for all interpolation sets.

For Lagrange interpolation polynomials $\mathcal{L}_i(x)$, Γ_Y describes an upper bound of the form
$$\Gamma_Y = \max_i \max_x |\mathcal{L}_i(x)|,$$
where $i = 1, \ldots, p$ and x varies in the convex hull of Y. The set Y can be maintained as poised by choosing the points which ARE to leave or to enter Y so that this bound is reduced [24]. In Powell's algorithm a point is replaced by a new one having the maximum of the Lagrange polynomials.

For Newton polynomials a similar approach is followed. When Newton polynomials are chosen, from theoretical point of view, the pivots of the matrix (3), i.e., the values of the non-normalized Newton fundamental polynomials at their associated interpolation points $|N_i^u(y_i)|$, must be non-zero to ensure the poisedness. However, in practice it may be sufficient to have
$$||N_i^u(y_i)| \geq \varepsilon \quad \text{for all } y_i \in Y.$$
If this condition is satisfied, then the interpolation problem is well-poised. If this condition is not satisfied, the geometry of the interpolation set have to be improved. A point $y_- = y_i \neq x_k$ is replaced by another point y_+ such that
$$y_+ = \mathrm{argmax}_{\mathrm{y} \in \mathcal{B}_\mathrm{k}} |N(y)|,$$
provided that $|N_i(y_+)| + 1 \geq \theta$ [4].

In [10] both approaches are criticized because they do not guarantee uniform bounds on Λ_Y and proposes algorithms on how to maintain the well-poised interpolation set. In these algorithms, the set of polynomials have to be computed at each updates of Y, but not the interpolation set.

Because the condition number of the matrix (3) depends only on the choice of the basis functions $\varphi_i(x)$, it can not be used as an indicator for the well-poisedness of the interpolation set. But the condition number of a scaled version of the matrix (3) can be used with the Lagrange polynomials as shown in [10] for deriving Taylor-like bounds for the function and its derivatives. This approach is named Λ-poisedness.

The second issue concerns the management of the trust-region radius. In the classical trust-region algorithms, the radius Δ_k is decreased when no significant reduction of the objective function was achieved. It can happen that a very small trust-region radius leads to stagnation of the iteration. In case of DFO algorithms, one has to check first whether the interpolation set Y is poised or not; because an inadequate geometry may be the reason for insufficient reduction of the objective function. If Y is not poised, first its geometry should be improved before reducing the trust-region radius. It is important not to modify Δ_k in the early stages of the iteration, because the inclusion of a new point y_+ with $||y_+ - y_k \leq ||\Delta_k$ and replacing some past points $y_- \in Y\backslash\{x_k\}$ by y_+ an improvement can be obtained.

The third issue deals with the stopping criteria. At each iteration we have to apply a convergence test. In classical trust-region algorithms it is sufficient to check that the gradient of f is small enough at the current iterate x_k. In case of DFO methods the gradient $\nabla f(x)$ is not available. If the model is sufficiently accurate, one can assume that

$$||g_k|| \approx ||\nabla f(x)||.$$

Whenever the quadratic model m_k has been updated, $||g_k||$ can be computed and it can be checked if it is small enough (if $||g_k|| \leq \epsilon$ for some ϵ). If this is the case, then the model is sufficiently accurate around a ball centered at x_k with

$$||f(x) - m_k(x)|| \leq \kappa\Delta^2 \quad \text{for all } x \quad \text{ such that } ||x - x_k|| \leq \Delta$$

, where κ is a constant independent of x [4] . This condition should be verified in a finite number of steps.

3. **Derivative-free methods for partially separable functions.**

The use of DFO methods is limited in practice to moderate size problems with about 20 variables. When the number of variables grow, the DFO methods will fail. But many problems arising in industry have a structure which appears in the form of blocks or parts. Each block has its own variables which are linked with the variables of other blocks. A collection of chemical reactor tanks each with its temperature and pressure can be considered as such systems. Discretization of PDE's (partial differential equations) with domain decomposition techniques are another example of this type; each single variable is only connected to a few neighborhoods. In order to develop the DFO methods work, the underlying structure of the objective function should be exploited to minimize the number of function evaluations. The structure in many problems mentioned above can be captured by the notion of partially separability introduced in [13], [15] which may be expressed as

$$\min f(x) = \sum_{i=1}^{q} f_i(U_i x), \tag{5}$$

where $f : \mathbb{R}^n \to \mathbb{R}$ and each individual function $f_i(x)$ is depending on the internal variables $U_i x$ with U_i being an $n_i \times n$ matrix ($n_i << q$). In many cases, the matrix U_i is a row subset of the identity matrix and the internal variables is form a small subset of the variables of the original objective function. The cardinality of these q subsets is given by n_i and the n_i-dimensional subspace of $\mathbb{R}^n$ is denoted $R_i, i = 1, \cdots, q$.

The concept of partial separability in connection with optimization problems was introduced by Griewank and Toint in [12], [14] and studied in the framework of the development of the LANCELOT package [5]. It was shown in [12] that every twice differentiable continuous function with a sparse Hessian is partially separable. This is only an existence result and specifies the decomposition (7) indirectly. But for all problems with a sparse Hessian, the partially separable decomposition is explicitly known [4], [28]. A special class of partially separable functions which occur in unconstrained optimization is that with a banded and sparse structure of the objective function [2], [3].

In the trust-region algorithm for partially separable functions, the quadratic model can be build for each individual function f_i as:

$$m_{i,k}(x_k + s) = f_i(x_k) + \langle g_i(x_k)\, ,\, s\rangle + \frac{1}{2}\langle s\, ,\, H_i(x_k)s\rangle\, , \tag{6}$$

where each $g_i \in \mathbb{R}^n$ and H_i is an $n_i \times n_i$ matrix. Each model $m_{i,k}$ is an approximation of f_i's around x_k and depends on the n_i dimensional vectors of the subspace R_i. After computation of the models (6) and interpolating individual functions, the quadratic model for the global objective function f at x_k is obtained as

$$m_k(x_k + s) = \sum_{i=1}^{q} m_{i,k}(x_k + s). \tag{7}$$

This model can then be minimized within the current trust region $\mathcal{B}_k$. This model requires at most $(n_{\max} + 1)(n_{\max} + 2)/2$ function evaluations, where $n_{\max} = \max_{i=1,\cdots,q} n_i$. This indicates that the number of function evaluations is by far less than for the unstructured problem because $n_{\max} << n$ and is often independent of n. The next problem is the management of the q interpolation sets Y_i, $i = 1, \cdots, q$. There are two possibilities. The objective function f can be evaluated as a whole by computing the collection of individual function evaluations $f_i(x)$'s and the models $m_{i,k}(x)$ for a given x. This would be very expensive in terms of function evaluations. The second strategy is the grouping of the necessary function evaluations. An individual function $f_i(x)$ can be computed independently of the others. Then the geometry of the jth interpolation set Y_j is improved by yielding a vector with n_i 'useful' components, then the question arises which values of this vector should be assigned to the remaining $n - n_j$ components. This is done in [4] by using the CPR procedure in [11] for estimating sparse Jacobian matrices to the $n \times q$ occurrence matrix of variables x_j into elements f_i's.

The reduction of function evaluations can be illustrated for an objective function with a tridiagonal Hessian: the unstructured DFO requires a $(n + 1)(n + 2)/2)$, sparse version of DFO needs $6n$ and partially separable DFO only 6 function evaluations. In a series of papers, Colson and Toint [2, 3, 28] compared the unstructured algorithm UDFO with its structured versions for partially separable functions, PSDFO(I) which refers to the case where each indvidual function $f_i(x)$ is individually known and PSDFO(G) in which the individual functions can be obtained by calling a routine and therefore the complete collection of $\{f_i(x)\}_{i=1}^{q}$ must be computed for every x. The numerical experiments for medium sized ($n = 25$) and large sized ($n = 200$) test problems show the benefits of using the the partially separability property of the objective functions. For many test problems PSDFO(I) requires less than half of function evaluations required for PSDFO(G). For problems with sparse and banded Hesssian and objective functions, the model m_k can be viewed as a linear combination of $1 + n + n_H$ monomials, where n_H is the number of nonzeros in the lower triangular part of $H(x)$, with n_H as a small fraction of n. In this case the interpolation set Y is linear rather than quadratic. The numerical results for such kind of problems reported in [2] and a comparison is made with Powell's sparse algorithms UOBSQA, UBDQA and with sparse version of the structured PSDFO(G) of Colson and Toint in [2, 28].

4. **Implementation of derivative-free algorithms.** There are several implementations of interpolation based on derivative free methods which differ in detail. The oldest one is the DFO (Derivative Free Optimization) package developed by Conn and coworkers [6, 7, 8]. The newly developed CONDOR (Constrained, Non-linear, Direct, parallel Optimization using trust Region method for high-computing load function) package by Berghen [1] is based on the UOBYQA of Powell [23]. The packages UDFO and PSDFO of Colson and Toint [2], [4] where mentioned above. Although there are some extensions for constrained problems, both packages are developed mainly for unconstrained optimization problems and for easy (box) constrains.

The differences between the DFO package and CONDOR which are used for the stirrer optimization in the next Section, are summarized as:

- The DFO package uses Newton polynomials while CONDOR uses Lagrange polynomials as the basis for the space of quadratic polynomials. The choice of Newton polynomials (in DFO) causes the matrix (3) to be lower triangular with a special block diagonal structure, while the use of Lagrange polynomials (in CONDOR) results in an identity matrix.
- The minimization of the possibly quadratic model in the DFO package is done by applying a standard optimization procedure, e.g., sequential quadratic programming (SQP), the IPOPT [29]. CONDOR uses the Moré and Sorenson algorithm [20] for the computation of the trust region radius and the minimization of the quadratic model, which is very stable.
- Furthermore, when the computation of the the trust-region radius Δ is checking completely the validity (with ϵ) of the model around the point x_k in CONDOR is based on whether one of the following conditions [1]

$$\|x_j - x_k\| \leq 2\delta \quad \text{for all } j \tag{8}$$

$$\frac{1}{6} M \|x_j - x_k\|^3 \max_d \{|P_j(x_j + d)| : \|d\| \leq \Delta\} < \epsilon \quad \text{for all } j \tag{9}$$

  holds for the Lagrange interpolating basis $\{P_j\}$, where $\left|\frac{d^3}{d\alpha^3} f(x + \alpha v)\right| \leq M$ such that $\|v\| = 1$ and $\alpha \in \mathbb{R}$. Condition (8) prevents the algorithm from sampling the model at $(n+1)(n+2)/2$ new points. However, the checking of the validity of the model in DFO is mainly based on condition (9), which is not very often satisfied by the trust-region radius. Hence, in some cases more function evaluations are needed in order to rebuild the interpolation polynomial.
- Another essential difference between the DFO and the CONDOR algorithms is that the former uses the smallest Frobenius norm of the Hessian matrix to minimize the local model, which may cause numerical instability, whereas CONDOR uses the Euclidean norm which is more robust.
- The DFO package uses interpolating polynomials which oscillate between the linear and quadratic. CONDOR uses full quadratic polynomials.
- There is a parallel extension of CONDOR.

Another variant of the DFO algorithms is the so-called wedge trust region method for linear and quadratic models [18]. This method differs from the others by the minimization of the quadratic model within the trust region with the aim to improve the geometry of the interpolation set. It can be illustrated for the linear model $m_k(x_k + s) = f(x_k) + g_k^T s$ as:

$$\min_s m_k(x_k + s) = f(x_k) + g_k^T s,$$

$$\text{subject to} \qquad ||s||\Delta_k,$$
$$||b_k^T s|| \geq \gamma ||b_k|| ||s||$$

The last inequality is called the wedge constraint. This formulation is also given for the quadratic model and the method is compared for several test examples in [18].

5. **Application to stirrer optimization.** The DFO package was successfully applied as a black-box optimization routine in optimizing energy systems [17] and for the helicopter rotor blade design [9], where some practical aspects of DFO algorithm were described. Numerical tests in both papers show that DFO method works faster and more accurately than the derivative based methods such as the Quasi-Newton. CONDOR was also compared with DFO package in some respects for several test problems and its applicability for some CFD simulations has been shown in [1].

Applications of numerical optimization techniques for fluid flow problems are found mostly in the field of aerodynamics. An overview of the subject is given by Mohammadi and Pironneau [19]. For general fluid flow applications the resulting nonlinear optimization problems are typically solved by employing descent algorithms based on gradient information. However, the computation of the gradient of the objective function with respect to the design variables is usually very costly and contaminated by noise. Usually the objective function, $f(x)$, is a result of very expensive computer simulations and yet the derivatives may not be available at all.

For the computer simulation of fluid inside the stirrer by the CFD package FASTEST-3D we have a prototype of such a problem. By simulating the complex discrete Navier-Stokes system, the local gradients of the objective function $f(x)$ with respect to the design variables x, are not provided by the flow solver and are not directly available. In such case we may easily apply derivative-free optimization methods. In [26] we have used the the DFO package to optimize the geometric parameters of the stirrer, whereas in [27] the numerical results obtained with CONDOR and DFO packages are compared. In this Section we present these results in order to show the applicability of both algorithms.

We consider a Rushton turbine as a representative test case for optimizing a practical stirrer configuration. The geometric parameters, which are considered to define the standard configuration, are given in Table 1.

Parameter	Value
Tank diameter	$T = 0.15\,\mathrm{m}$
Impeller diameter	$D = T/3 = 0.05\,\mathrm{m}$
Bottom clearance	$C = H/2 = 0.075\,\mathrm{m}$
Height of the liquid	$H = T = 0.15\,\mathrm{m}$
Length of the baffles	$W = 3D/10 = 0.015\,\mathrm{m}$
Length of the blade	$\ell = D/4 = 0.0125\,\mathrm{m}$
Height of the blade	$w = D/5 = 0.01\,\mathrm{m}$
Disc thickness	$x = D/5 = 0.00175\,\mathrm{m}$
Diameter of the disk	$d = 3D/4 = 0.0375\,\mathrm{m}$

Table 1. Geometrical parameters of standard stirrer configuration.

The working Newtonian fluid is a glucose solution with density $\rho = 1330\,\mathrm{kg/m^3}$ and viscosity $\mu = 0.105\,\mathrm{Pas}$.

The numerical grid consists of 22 blocks with a total number of 238 996 control volumes. There are 17 blocks defined as rotating with the stirrer while the remaining 5 blocks are defined as stationary with the vessel. A sketch of the considered configuration and the corresponding surface grid of the stirrer are shown in Figure 1. One time step integration needs approximately 20 seconds of computing time on an eight processor Redstone cluster machine. This results in about 8 hours of computing time to reach a steady state flow in the sense of a frozen rotor computation, a criterion that we adopt for all cases.

The dimensionless Newton number, Ne, which relates the resistance force to the inertia force,

$$Ne = \frac{P}{\rho N^3 D^5} \quad \text{where} \quad P = -\int\limits_S (p u_j + \tau_{ij} u_i)\, n_j \,\mathrm{d}S\,,$$

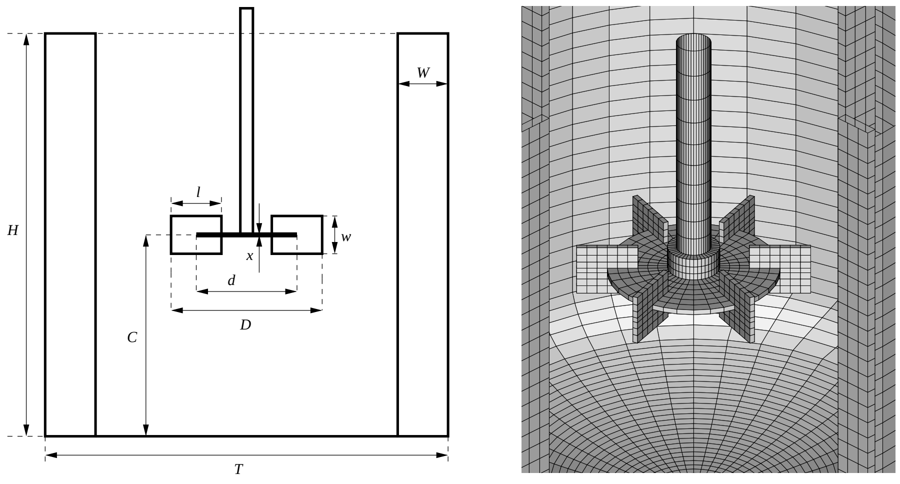

FIGURE 1. Sketch of geometrical parameters of stirrer configurration (left) and its corresponding surface grid (right).

is considered as the characteristic reference quantity to be minimized. Here, N denotes the rotational speed of the impeller and P is the power computed from the flow quantities over the surface S of the impeller (pressure p, velocity u_i, stress τ_{ij}).

The design variables are the disk thickness x, the bottom clearance C, and the baffle length W, for which the inequality constraints $0.001 \leq x \leq 0.005$, $0.02 \leq C \leq 0.075$, and $0.005 \leq W \leq 0.03$ are prescribed. All other parameters are kept fixed according to the standard configuration.

Because the gradient information is not available, for both algorithms the minimum of the trust region radius, δ_{min}, is used as a stopping criterion. In DFO, for $Re = 100$, $\delta_{min} = 0.0001$ and for $Re = 1000$, $\delta_{min} = 0.001$ were chosen. For CONDOR, we have used $\delta_{min} = 0.0001$ for both Reynolds numbers.

As a first example we consider the minimization of the Newton number for a Reynolds number of $Re = 1000$. Figure 2 shows the Newton number versus the number of cycles for the two optimization algorithms. For both algorithms the Newton number attains the same optimum. CONDOR terminates earlier then the DFO although the latter oscillates around the optimum. Figure 3 depicts the corresponding changes of the design variables during the function evaluations. The three design parameters assign almost the same optimal values. We remark that the two optimization tools differ in building the starting model: DFO starts to build the model objective function approximation (unique or not) from the very beginning. That is, its starting objective polynomial approximation is not fully quadratic. On the other hand, CONDOR starts with a fully quadratic model for the objective function. Despite its long initialization it turns out that CONDOR needs less function evaluations than DFO to reach an optimum point after completing the initial quadratic approximation. DFO, as the time passes, oscillates around the minimum although it approaches the minimum very sharply at the beginning. CONDOR waits for a full quadratic polynomial to build its model, and then gradually approaches the minimum, using a complete interpolating bases. In any case, however, both algorithms reach the same optimized Newton number which is significantly (about 37%) lower than the one for the standard tank configuration stated in Table 1.

6. **Conclusions and perspectives.**

The main drawback of DFO methods is that they can be applied to moderately sized problems, i.e. to problems with around 20 variables.

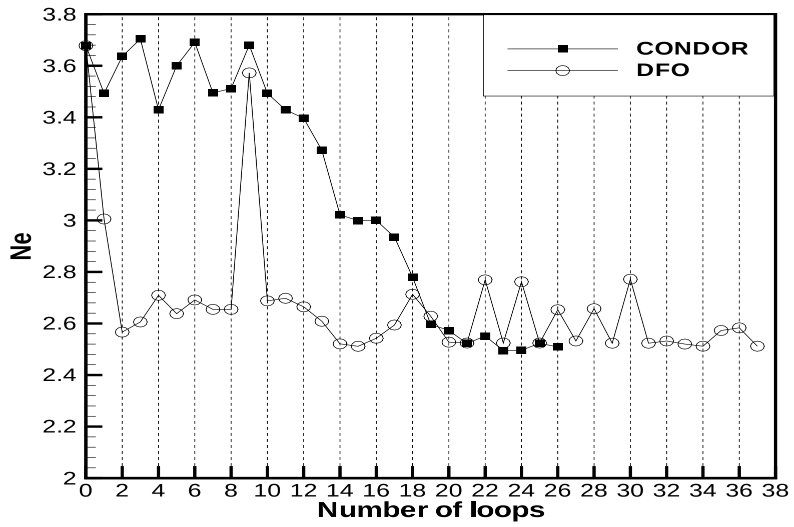

FIGURE 2. Newton number versus number of the loops.

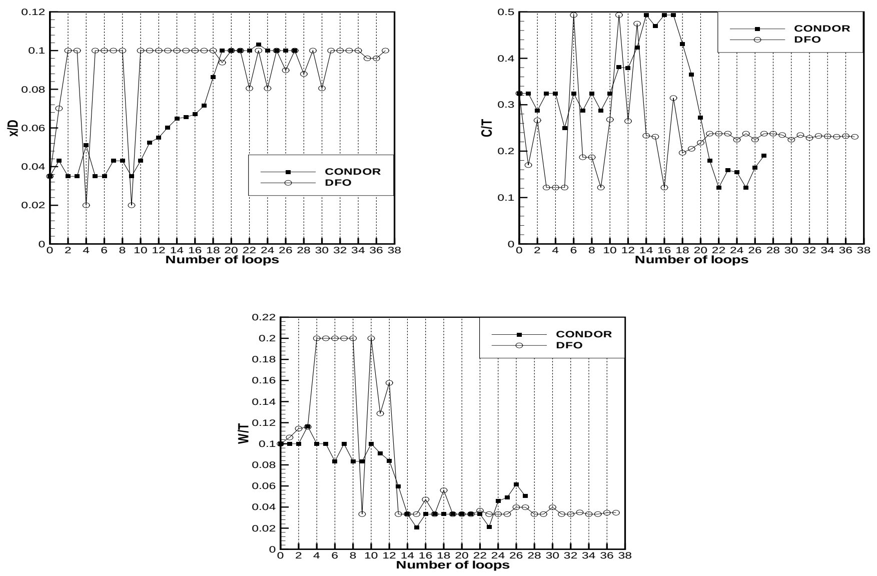

FIGURE 3. Design variables(disc thickness, bottom clearance, baffle length) versus number of function evaluations

In order to apply these methods for large-scale problems in applications, the underlying structure of the objective functions should be exploited. There is some progress for sparse problems and for partially separable functions as mentioned in this paper. Problems arising in CFD simulation and other similar large-scale optimization problems are potential application areas of DFO algorithms. Another theoretically and practically interesting issue is the maintaining of the geometry of the interpolation set at each iteration step. There are several approaches in this direction, but a practical algorithm for handling this problem is still missing. Another is issue could be the development of parallel versions of the algorithms as it was done for CONDOR.

7. **Acknowledgements.** The financial support of the work by the Volkswagen Foundation is gratefully acknowledged.

REFERENCES

[1] F. V. Berghen, H. Bersini, *CONDOR a new parallel, constrained extension of Powell's UOBYQA algorithm: Experimental results and comparison with the DFO algorithm*, Journal of Computational and Applied Mathematics, **181** (2005), 157–175

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

[3] B. Colson, Ph. L. Toint, *A derivative-free algorithm for sparse unconstrained optimization problems*, Optimization and Engineering, **2** (2001), 349-412

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

[5] A.R. Conn, N.I.M. Gould, Ph.L. Toint, *LANCELOT: a FORTRAN Package for Large-Scale Nonlinear Optimization*, Springer Verlag, 1992

[6] A. R. Conn, P.L. Toint, *An algorithm using quadratic interpolation for unconstrained derivative free optimization*, in "Nonlinear Optimization and Applications", G. Di Pillo and F. Gianessi, eds, Plenum Publishing, New York, (1996), 27–47

[7] A. R. Conn, K. Scheinberg, P.L. Toint, *On the convergence of derivative-free methods for unconstrained optimization*, in "Approximation Theory and Optimization: Tribute to M.J.D. Powell", A. Iserles, M. Buhmann, eds, Cambridge University Press, (1997), 83–108

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

[9] K. Scheinberg, *Derivative free optimization method*, IBM T.J. Watson Research Center, Newyork, Tech. Rep. Report 98/11, (2000)

[10] A.R. Conn, K. Scheinberg, L.N. Vicente, *Geometry of interpolation sets in derivative free optimization*, to appear in Mathematical Programming

[11] A.R. Curtis, M.J.D. Powell, J.K. Reid *On the estimation of sparse Jacobian matrices*, Journal of Institute of Mathematics and Apllications, **79** (1974), 117-119

[12] A. Griewank, Ph. L. Toint *Partitioned variable metric updates for large scale optimization problems*, Numerische Mathematik, **39** (1982), 119-137

[13] A. Griewank, Ph. L. Toint, *On the constrained optimization of partially separable functions*, in MJ.D. Powell, ed., Nonlinear Optimization, Academic Press, (1982), 301-312

[14] A. Griewank, Ph. L. Toint, *Local convergence analysis of partitioned quasi-Newton updates*, Numerische Mathematik, **39** (1982), 429-448

[15] A. Griewank, Ph. L. Toint, *On the existence of convex decompositions of partially separable functions*, Mathematical Programming, **28** (1984), 25-49

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

[17] P. van der Lee, T. Terlaky, T. Woudstra, *A new approach to optimizing energy systems*, Computer Methods in Applied Mechanics and Engineering, **190** (2001), 5297–5310

[18] M. Marazzi, J. Noncedal, *Wedge trust region methods for derivative free optimization*, Mathematical Programming Ser. A, **91** (2002), 289-305

[19] B. Mohammadi, O. Pironneau, *Applied Shape Optimization for Fluids*, Oxford University Press, (2001)

[20] J.J. Moré, D.C. Sorenson, *Computing a trust region step*, SIAM. J. Sci. Statsit. Comput., **4** (1983), 553-572

[21] M.J.D. Powell, *An efficient method for finding the minimum of a function of several variables without calculating derivatives*, Computer Journal, **17** (1964), 175-162

[22] M.J.D. Powell, *A method for minimizing a sum of squares of nonlinear functions without calculating derivatives*, Computer Journal, **7** (1965), 303-307

[23] 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, (1994)

[24] M. J. D. Powell, *On thne Lagrange functions of quadratic models that are defined by interpolation*, Optimization Methods Software **16** (2001), 289-309

[25] M. J. D. Powell, *UOBYQA: unconstrained optimization by quadratic approximation*, Mathematical Programming, **92** (2002), 555-582

[26] M. Schäfer, B. Karasözen, Y. Uludağ, K. Yapıcı, Ö. Uğur *Numerical Method for Optimizing Stirrer Configurations*, Computers & Chemical Engineering, **30**, (2005), 183-190

[27] M. Schäfer, B. Karasözen, K. Yapıcı,Ö. Ugur, *Derivative Free Optimization of Stirrer Configurations*, the Proceedings of the ENUMATH 2005 the 6th European Conference on Numerical Mathematics and Advanced Applications, Santiago de Compostela, Spain, July 2005, A. Bermúdez, A. Gómez, D. Quintela, P. Salgado, eds, Springer, 2006

[28] Ph. L. Toint, *Using problem structure in derivative-free optimization*, SIAG/OPT Views-and-News, **79** (2006), 11-18

[29] A. Waechter, *An Interior Point Algorithm for Large-Scale Nonlinear Optimization with Applications in Process Engineering* , PhD. thesis, Department of Chemical Engineering, Carnegie Mellon University (2002)

[30] D. Winfried, *Function and functioanl optimization by interpolation in data tables*, PhD thesis, Harvard University, Cambridge, USA (1969)

[31] D. Winfield, *Function minimization by interpolation in a data table*, Journal of the Institute of Mathematics and its Applications, **12** (1973), 339-347

E-mail address: `bulent@math.metu.edu.tr`