Derivative free optimization method

Katya Scheinberg
IBM Watson Research Center

1 Main idea

Derivative free optimization (DFO) methods are typically designed to solve optimization problems whose objective function is computed by a "black box"; hence, the gradient computation is unavailable. Each call to the "black box" is often expensive, so estimating derivatives by finite differences may be prohibitively costly. Finally, the objective function value may be computed with some noise, and the finite differences estimates may not be accurate.

All above properties, such as relatively expensive "black box" computations and presence of noise, are characteristic of the Cycle-Tempo optimization problems. However, it is the noise which creates most difficulty in applying gradient based methods to these problems.

The derivative free optimization method which we use approximates the objective function explicitly without approximating its derivatives. The theoretical analysis presented below assumes that no noise is present. However, extensive experiments and intuition support the claim that robustness of DFO does not suffer from presence of moderate level of noise.

Various other methods were developed recently to handle similar classes of problems (see [11], [12], [8], [6], [1]). In [1] a modification of a quasi-Newton method that accommodates noise in the objective function is considered. However, the analysis in [1] assumes that the level of noise reduces to zero when optimal solution is approached. This assumption is not realistic in case of Cycle-Tempo.

The DFO method that we use here compares favorably with many other existing derivative free methods (both in speed and in accuracy). It is described in [2] and [3] and is essentially a combination of a trust region framework with quadratic interpolation of the objective function.

Using polynomial interpolation models within trust region frameworks has been proposed earlier in [8], [6] and [5]. The main distinction of the method in [2] and [3] from these earlier methods is in the approach to handling the geometry of the interpolation set. In Subsection 3 we briefly describe the essence of this approach; for more detailed analysis the reader is referred to [2].

First, we describe the trust region framework.

2 Trust region framework

The trust region framework is usually used in the context when at least the gradient and sometimes Hessian of the objective function can be evaluated or estimated accurately. The main steps of a typical trust region method are the following

$\boxed{\textbf{Steps of a trust region method}}$

1. Given a current iterate build a good local approximation model (e.g., based on a second order Taylor series approximation).
2. Choose a neighborhood around the current iterate where the model is “trusted” to be accurate. Minimize the model in this neighborhood.
3. Determine if the step is successful by evaluating the true function at the new point and comparing the true reduction in value of the objective with the reduction predicted by the model.
4. If the step is successful, accept the new point as the next iterate and proceed (possibly, increasing the size of the trust region if the success is really significant).

   If the step is unsuccessful, reject the new point and reduce the size of the trust region.
5. Repeat until convergence.

For a model based on the Taylor series expansion we know that if the trust region is made small enough, then the approximation is sufficiently accurate and the algorithm will make a successful step (unless the optimum has been reached).

To use trust region framework in the derivative free case we use an alternative approximation technique, which does not use derivative estimates. Quadratic interpolation is one such technique, which can be applied successfully within a trust region method. However, we need to guarantee that the approximation model is locally good; that is, that a successful step will be made after sufficient reduction of the trust region.

The mechanism of maintaining interpolation sets, described in the next subsection, is designed to overcome this difficulty. As long as the interpolation set is, what we call, well-poised within the trust region, the interpolation model is locally good.

3 Interpolation

In this subsection we consider the problem of interpolating a given function $f(x)$, $x \in \mathbf{R}^n$ by a (quadratic) polynomial $Q(x)$ at a chosen set of interpolation points, $Y = \{y^j\}_{j=1}^p \subset \mathbf{R}^n$. In other words, we need to find $Q(x)$, for which

$$Q(y^j) = f(y^j) \quad j = 1, \ldots, p. \tag{1}$$

We will say that a set of points *can be interpolated* by a polynomial of a certain degree if for any function f there exists a polynomial m (of this degree) such that (1) hold for all points in the set.

In the case of univariate interpolation any set of distinct point can be interpolated by a polynomial of appropriate degree. For example, any 3 distinct points can be interpolated by a quadratic function. In multivariate interpolation, however, this is not the case. For example, to obtain unique quadratic interpolation of a function in two dimensions one needs six interpolation points; but a set of six points on a line cannot be interpolated by a quadratic polynomial.

Definition 3.1 *A set of points Y is called poised, with respect to a given subspace of polynomials, if it can be interpolated by polynomials from this subspace.*

For example, suppose $n = 2$ and Y is a set of six points on a unit circle. Then Y cannot be interpolated by a polynomial of the form $a_0 + a_1x_1 + a_2x_2 + a_{1,1}x_1^2 + a_{1,2}x_1x_2 + a_{2,2}x_2^2$, thus, Y is not poised with respect to the space of quadratic polynomials. On the other hand, Y *can be* interpolated by a polynomial of the form $a_0 + a_1x_1 + a_2x_2 + a_{1,1}x_1^2 + a_{1,2}x_1x_2 + a_{1,1,1}x_1^3$, thus, Y is poised in an appropriate subspace of the space of cubic polynomials.

For our purposes, we are interested in *well poised* sets. For the moment we give a general and intuitive definition.

Definition 3.2 *A set of points Y is called well-poised, if it remains poised under small perturbations.*

For example, if $n = 2$, six points *almost* on a line may make a poised set, however, since some small perturbation of the points might make them aligned, it is not *well-poised* set, the interpolation will be very ill-conditioned and is likely to provide a very bad approximation of the function.

We now fill the intuition with mathematical substance. Below we show that the concept of poisedness relates to nonsingularity of a certain matrix. Then we show how this singularity can be detected by applying an analog of a Gramm-Schmidt procedure to the matrix. We will also relate the concept of well-poisedness to that procedure and finally connect well-poisedness and the quality of an approximation.

Suppose $\{\phi_i(\cdot)\}_{i=1}^q$ is a basis in the space of quadratic polynomials, then any quadratic polynomial $Q(x) = \sum_{i=1}^q \alpha_i\phi_i(x)$ for some $\alpha = (\alpha_1, \ldots, \alpha_q)$. The interpolation condition can be written as a system of linear equations in α.

$$\sum_{i=1}^{q} \alpha_i\phi_i(y^j) = f(y^j) \quad j = 1, \ldots, p.$$

The coefficient matrix of this system is

$$\Phi(Y) = \begin{pmatrix} \phi_1(y^1) & \cdots & \phi_q(y^1) \\ \vdots & & \vdots \\ \phi_1(y^p) & \cdots & \phi_q(y^p) \end{pmatrix}.$$

For a given set of points and a set of function values an interpolation polynomial (from a given space of polynomials) exists and is unique if and only if $\Phi(Y)$ is square and nonsingular.

From here onwards we assume that $\Phi(Y)$ is square, that is that cardinality of Y and that of the basis $\{\phi_i(\cdot)\}$ is the same and equals p. For a full quadratic interpolation p equals $\frac{(n+1)(n+2)}{2}$. Notice that nonsingularity of $\Phi(Y)$ is independent of the choice of the polynomial basis, as long as the space that is spanned by the basis is fixed. With respect to a given space of polynomials (in our case it is typically the space of quadratic polynomials), a set of point Y is *poised* if $\Phi(Y)$ is nonsingular.

If the set Y is poised, then, theoretically, we can solve the linear system and find the interpolation polynomial. However, numerically the matrix $\Phi(Y)$ is often ill-conditioned even when it is nonsingular. Conditioning of $\Phi(Y)$, clearly, depends of the choice of the basis $\{\phi_i(x)\}$. For example, one can choose a basis such that $\Phi(Y)$ is an identity matrix. Such basis is called Lagrange fundamental polynomial basis.

The interpolation technique used within DFO framework is based on a so-called Newton fundamental polynomial (NFP) basis. For a given set of interpolation point Y the basis of Newton fundamental polynomial is such that $\Phi(Y)$ has the structure shown in Figure 1.

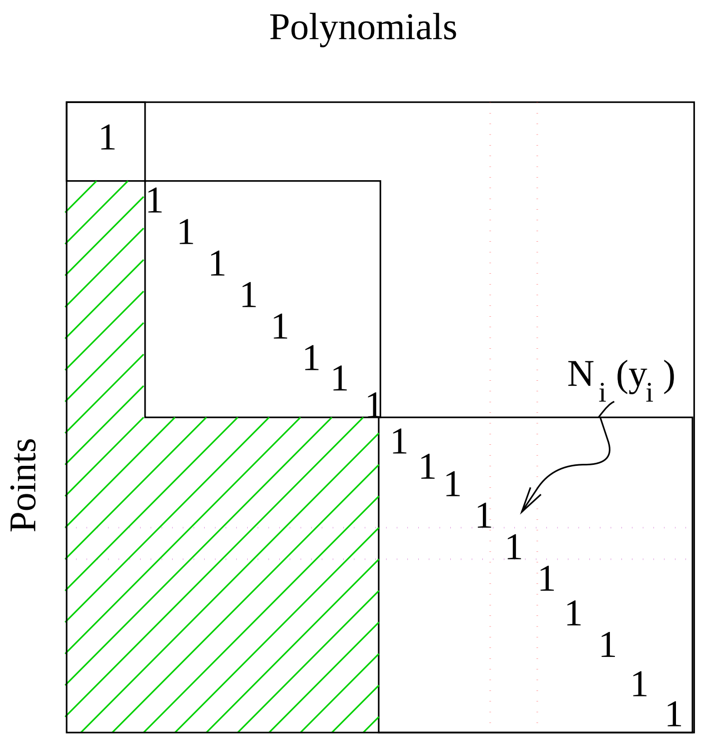

Figure 1: Newton fundamental polynomial

More precisely, the set of interpolation points is partitioned into three subsets (blocks) Y^0, Y^1 and Y^2 which correspond to constant terms, linear terms and quadratic terms of a quadratic polynomial, respectively. Hence, Y^0 has a single element, Y^1 has n elements and Y^2 has $\frac{n(n+1)}{2}$ elements. The basis of NFP $\{N_i(\cdot)\}$ is also partitioned into three blocks $\{N_i^0(\cdot)\}$, $\{N_i^1(\cdot)\}$ and $\{N_i^2(\cdot)\}$ with the appropriate number of elements in each block. Moreover, the unique element of $\{N_i^0(\cdot)\}$ is a polynomial of degree zero, each of the n elements of $\{N_i^1(\cdot)\}$ is a polynomial of degree one and, finally, each of the $\frac{n(n+1)}{2}$ elements of $\{N_i^2(\cdot)\}$ is a polynomial of degree two.

The basis elements and the interpolation points are set in one-to-one correspondence, so that points from block Y^l correspond to polynomials from block $\{N_i^l(\cdot)\}$. A Newton polynomial $N_i(\cdot)$ and a point y^i correspond if and only if the value of that polynomial at that point is *one* and its value at *any* other point in the same block or in any prior block is *zero*. In other words, if y^i corresponds to N_i, then $N_i(y^i) = 1$, and $N_i(y^j) = 0$ for all index j within first l blocks (see Figure 1).

Obtaining the NFP basis is equivalent to reducing $\Phi(Y)$ to the form in Figure 1. This can be done if and only if set Y is poised. One can use an analogue of Gramm-Schmidt orthogonalization method starting with any basis, for example the basis of monomials:

$$\{1, x_1, x_2, \ldots, x_n, x_1^2, x_1x_2, \ldots, x_{n-1}x_n, x_n^2\}$$

and applying the following pivoting procedure (see Figure 2):

Procedure 3.3 *Initialize* $\{\bar{N}_i(\cdot)\}_{i=1}^p$.

```
For $i = 1, \ldots, p$:
    **Normalize:** $\bar{N}_i(x) \leftarrow \frac{\bar{N}_i(x)}{\bar{N}_i(y^i)}$
    **Orthogonalize:** $\bar{N}_j(x) \leftarrow \bar{N}_j(x) - \bar{N}_j(y^i)\bar{N}_i(x)$
    for $j$ in the same or "later" block as $i$
Set $N_i(\cdot) = \bar{N}_i(\cdot)$, $i = 1, \ldots, p$.
```

We use notation $\bar{N}_i(\cdot)$ to indicate intermediate polynomials, obtained during Procedure 3.3, and to distinguish them from the Newton fundamental polynomials $N_i(\cdot)$, obtained at the end the procedure.

In Figure 1. the normalization step divides the i-th column of $\Phi(Y)$ by $N_i(y^i)$ and the orthogonalization step subtracts a multiple of the i-th column from all columns in the same or later blocks setting the elements in the i-th row to zero.

Consider the following example. If we consider quadratic interpolation on a plane, we require six interpolation points using three blocks

$$Y^0 = \{(0,0)\}, \quad Y^1 = \{(1,0),(0,1)\}, \text{ and } Y^2 = \{(2,0),(1,1),(0,2)\},$$

corresponding to the initial basis functions $1, x_1, x_2, x_1^2, x_1x_2$ and x_2^2 respectively. Applying Procedure 3.3 then yields

$$N_1 = 1, \quad N_2 = x_1, \quad N_3 = x_2,$$
$$N_4 = \tfrac{1}{2}(x_1^2 - x_1), \quad N_5 = x_1x_2 \text{ and } N_6 = \tfrac{1}{2}(x_2^2 - x_2).$$

Set Y is poised if and only if for every step i the pivot value $\bar{N}_i(y^i)$ is nonzero. Thus, the set Y is poised if and only if we can apply Procedure 3.3 to it without encountering zero pivots. However, for numerical purposes it is important that $|N_i(y^i)|$ is sufficiently large (not very close to zero). Small pivot values result in very large coefficients of the Newton fundamental polynomials

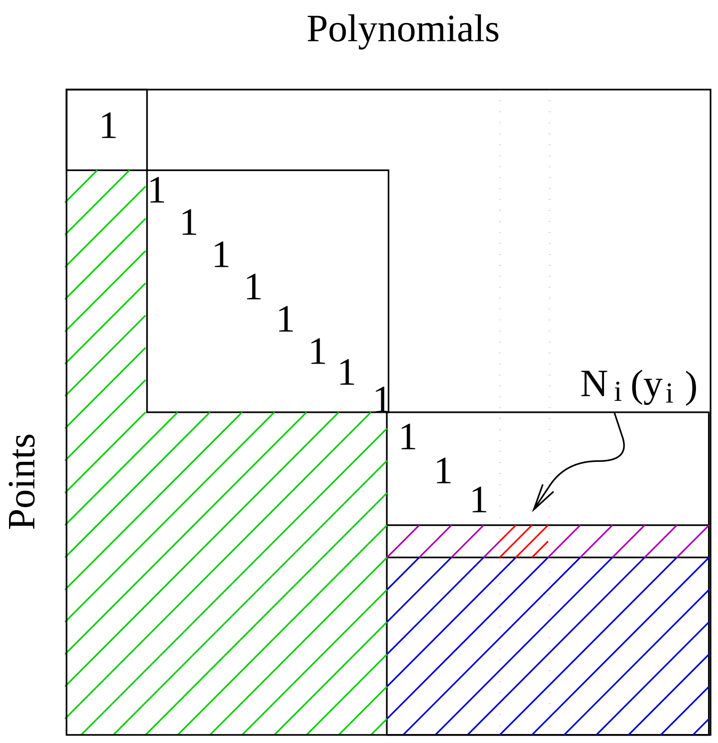

Figure 2: Pivoting procedure

(due to the normalization step) and lead to numerical instability. From now on we would mainly be interested in well-poised sets.

The concept of a well-poised set relates nicely to an error bound derived in [9] by means of NFP (for more detailed description of the results see [10] and [2]). This error bound depends on the value of Newton fundamental polynomials, which in turn depend on the coefficients of the polynomials. The error bound indicates that an interpolation is "locally good" as long as the values of Newton polynomials in the local neighborhood are not too large.

More precisely, the following lemma is a simplification of a result in [9].

Lemma 3.4 *Let $f(x)$ be a twice continuously differentiable function. Let Y be a poised set of interpolation points and $Q(x)$ be the interpolation polynomial. Given a point x, let $\pi(x) = \{y^0, y^1, y^2, y_3\}$ be a set containing three interpolation points y^0, y^1 and y^2, one from each block (recall the devision of Y into three blocks, according to the degrees of the associated polynomials), and the point $y_3 = x$; let $\Pi(x)$ be the set of all possible $\pi(x)$. The following interpolation error bound applies at point x*

$$|f(x) - Q(x)| \leq \frac{n^3 \|\nabla^2 f(x)\|_\infty}{6} \sum_{\pi(x)\in\Pi(x)} \left[\prod_{i=0}^{2} \|y^{i+1} - y^i\|_\infty \, |N_i(y^{i+1})| \right] \qquad (2)$$

The above bound is of interest locally; i.e., when all distances between y^i and y^{i+1} are small. We can tighten the error bound by iteratively constructing different interpolation sets while driving distances between interpolation points to zero as long as the other terms stay uniformly bounded. We assume that the Hessian of $f(x)$ is uniformly bounded in norm for all x. We would like to have a uniform bound on $|N_i(y^{i+1})|$.

We use the following, more specific, definition of a well-poised set. Given a neighborhood $\mathcal{B}$ and a constant K (independent of $\mathcal{B}$) we say that an interpolation set Y is *well-poised* if there exists a corresponding basis of Newton fundamental polynomials such that $|N_i(x)| \leq K$ for all $i = 1, \ldots, p$ and all x in $\mathcal{B}$.

In [2] it is shown that Y is well-poised if we can apply Procedure 3.3 with a positive threshold (lower bound) on the absolute value of the pivots. In theory, it is not a useful strategy. Theoretically, the radius of the trust region may converge to zero in the course of the optimization algorithm. Thus, applying Procedure 3.3 with a uniform pivot threshold may not be possible, since a threshold implies a uniforms lower bound on a distance between the interpolation points. This lower bound may turn out to be larger than the radius of the trust region, in which case there are no point in the trust region that are far enough apart. In [2] it is shown that a well-poised set Y can be always constructed in $\mathcal{B}$, by using a technique different from threshold pivoting.

In practice, on the other hand, the radius of the neighborhood is not driven to zero and threshold pivoting can be used in most cases for maintaining a well-poised interpolation set. One of the advantages of the threshold pivoting is that it prevents points that are very close to each other to be included in the interpolation set, but rather it has a "spreading" effect on the interpolation points. This helps to maintain the noise and it is, possibly, one of the major factors responsible for the robustness of DFO on noisy problems.

A question remains, what should be done in case a "bad" pivot is encountered during Procedure 3.3. Typically, one should consider all possible remaining pivots and choose a pivot that passes the threshold. If no such pivot exists, the Procedure 3.3 is terminated and the interpolation set Y is restricted only to the points which correspond to completed pivots. Thus, one may exit with an incomplete interpolation set and with a set of Newton polynomials which span only some subspace of the space of quadratic polynomials. In this case an interpolation can still be constructed in the subspace. As long as the interpolation is at least fully linear (spans the space of linear polynomials) a bound, similar to (2), applies. This bound shows that a well-poised interpolation is "locally good" and provides a first order approximation of $f(x)$.

Procedure 3.3 can also be applied when the initial cardinality of the interpolation set $p < \frac{(n+1)(n+2)}{2}$. The procedure simply terminates once p Newton fundamental polynomials have been constructed. Once again, the outcome of such procedure is an incomplete basis of Newton polynomials. The subspace spanned by them clearly depends on the choice of the initial basis. For example, if $n = 2$ and $p = 5 < \frac{(n+1)(n+2)}{2} = 6$, and the initial basis is $\{1, x_1, x_2, x_1^2, x_1x_2, x_2^2\}$ then the final incomplete set of Newton polynomials will span the subset of quadratic function of the form $a_0 + a_1x_1 + a_2x_2 + a_3x_1^2 + a_4x_1x_2$.

To complete an interpolation set to a full well-poised set one can continue Procedure 3.3 by picking an arbitrary interpolation point from the appropriate neighborhood, so that the value of the pivot can pass the threshold. For example, one can maximize the absolute value of the next pivot in a given neighborhood and include the maximum point as the new interpolation point (provided that the maximum value is above the threshold). Such procedure is likely to improve overall quality of the interpolation model.

4 Algorithm

We are now ready present an outline of the DFO algorithm.

DFO algorithm

Step 0: Initializations.

Let a starting point x^s and the value $f(x^s)$ be given.
Choose an initial trust region radius $\Delta_0 > 0$.
Choose at least one additional point not further than Δ_0 away from x^s to create an initial well-poised interpolation set Y and initial basis of Newton fundamental polynomials.
Determine $x^0 \in Y$ which has the best objective function value; i.e., $f(x^0) = \min_{y^i \in Y} f(y^i)$.
Set $k = 0$.
Set parameters η_0, η_1 to measure progress: $0 < \eta_0 < \eta_1 < 1$.

Step 1: Build the model.

Using the interpolation set Y and the basis of NFP, build an interpolation model $Q_k(x)$.

Step 2: Minimize the model within the trust region.

Set $\mathcal{B}_k = \{x : \, ||x - x^k|| \le \Delta_k\}$. Compute the point $\hat{x}^k$ such that

$$Q_k(\hat{x}^k) = \min_{x \in \mathcal{B}_k} Q_k(x).$$

Compute $f(\hat{x}^k)$ and the ratio

$$\rho_k \equiv \frac{f(x^k) - f(\hat{x}^k)}{Q_k(x^k) - Q_k(\hat{x}^k)}.$$

Step 3: Update the interpolation set.

- If $\rho_k \ge \eta_0$, *include* $\hat{x}^k$ in Y, dropping one of the existing interpolation points if necessary.
- If $\rho_k < \eta_0$, *include* $\hat{x}^k$ in Y, if it improves the quality of the model
- If $\rho_k < \eta_0$ and there are less that $n+1$ points in the intersection of Y and $\mathcal{B}_k$, *generate* new interpolation point in $\mathcal{B}_k$, while preserving/improving well-poisedness.
- Update the basis of the Newton Fundamental polynomials.

Step 4: Update the trust-region radius.

- If $\rho_k \ge \eta_1$, increase the trust region radius

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

- If $\rho_k < \eta_0$ and cardinality of $Y \cap \mathcal{B}_k$ was less that $n+1$ when $\hat{x}^k$ was computed, reduce the trust region

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

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

Step 5: Update the current iterate.

Determine $\bar{x}^k$ with the best objective function value

$$f(\bar{x}^k) = \min_{\substack{y^i \in Y \\ y^i \neq x^k}} f(y^i).$$

If improvement is sufficient (w.r.t. predicted improvement)

$$\bar{\rho}_k \equiv \frac{f(x^k) - f(\bar{x}^k)}{Q_k(x^k) - Q_k(\hat{x}^k)} \geq \eta_0,$$

set $x^{k+1} = \bar{x}^k$. Otherwise, set $x^{k+1} = x^k$. Increment k by one and go to Step 1.

End of algorithm

Remark 1. At each iteration up to two new interpolation points may be generated. First type of points is generated by minimizing the model and hopefully has better objective function value. Another type of points may be generated to improve the interpolation set/model. However, it is often happens, that a point of the first type improves the interpolation set and a point of the second type has the best up-to-date objective function value. Since the function evaluations may be expensive, it is important to exploit as much information as possible from all generated points.

Remark 2. Notice that the algorithm may start with interpolation set Y containing just two points and slowly build it up. It is never required for the cardinality of Y to be larger than $n + 1$, however it is encouraged to increase Y as long as it remains well-poised and contains local information. For first order convergence properties it is sufficient that the interpolation model is fully linear. On the other hand, maintaining full quadratic interpolation may be too expensive. For example, if $n = 20$, it requires 231 interpolation points to build a full quadratic model. It may be unacceptable to have to evaluate objective function 231 times before obtaining any progress in the algorithm.

For Cycle-Tempo problems, where n is around 10 and the function evaluations are not too expensive, full quadratic models are often used. This improves the robustness of the code.

Remark 3. We do not give almost any details on the implementation of step 3. There are various ways on improving an interpolation set and various criteria for adding or deleting interpolation points. We use techniques based on properties of Newton fundamental polynomials and on threshold pivoting. Some details are given in [2] and [3].

In [2] the convergence of the above algorithm is analyzed. Here we just state the final result.

Theorem 4.1 *Under some additional mild assumptions on the problem, any limit point x^* of the DFO algorithm is a stationary point, that is $\nabla f(x^*) = 0$.*

References

[1] T. D. Choi and C. T. Kelley. Superlinear convergence and implicit filtering. CRSC-TR99-14, Center for Research in Scientific Computation, North Carolina State University, Raleigh, NC. March 1999.

[2] A. R. Conn, K. Scheinberg, and Ph. L. Toint. On the convergence of derivative-free methods for unconstrained optimization. In A. Iserles and M. Buhmann, editors, *Approximation Theory and Optimization: Tributes to M. J. D. Powell*, pages 83–108, Cambridge, UK, 1997. Cambridge University Press.

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

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

[5] A. R. Conn and Ph. L. Toint. An algorithm using quadratic interpolation for unconstrained derivative free optimization. In G. Di Pillo and F. Gianessi, editors, *Nonlinear Optimization and Applications*, pages 27–47, New York, 1996. Plenum Publishing.

[6] 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, Proceedings of the Sixth Workshop on Optimization and Numerical Analysis, Oaxaca, Mexico*, volume 275, pages 51–67, Dordrecht, NL, 1994. Kluwer Academic Publishers.

[7] M. J. D. Powell. A direct search optimization method that models the objective by quadratic interpolation. Presentation at the 5th Stockholm Optimization Days, 1994.

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

[9] Th. Sauer. Notes on polynomial interpolation. Private communication, February 1996.

[10] Th. Sauer and Yuan Xu. On multivariate Lagrange interpolation. *Mathematics of Computation*, 64:1147–1170, 1995.

[11] V. Torczon. On the convergence of the multidirectional search algorithm. *SIAM Journal on Optimization*, 1(1):123–145, 1991.

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