SIAM J. OPTIM.
Vol. 20, No. 6, pp. 3555–3576

A DERIVATIVE-FREE ALGORITHM FOR LEAST-SQUARES MINIMIZATION*

HONGCHAO ZHANG†, ANDREW R. CONN‡, AND KATYA SCHEINBERG§

Abstract. We develop a framework for a class of derivative-free algorithms for the least-squares minimization problem. These algorithms are designed to take advantage of the problem structure by building polynomial interpolation models for each function in the least-squares minimization. Under suitable conditions, global convergence of the algorithm is established within a trust region framework. Promising numerical results indicate the algorithm is both efficient and robust. Numerical comparisons are made with standard derivative-free software packages that do not exploit the special structure of the least-squares problem or that use finite differences to approximate the gradients.

Key words. derivative-free optimization, least-squares, trust region, Levenberg–Marquardt method, system of nonlinear equations, global convergence

AMS subject classifications. 65K05, 90C30, 90C56

DOI. 10.1137/09075531X

1. Introduction.

In this paper, we design a class of derivative-free optimization algorithms for the following least-squares problem:

$$\min\ \Phi(\mathbf{x}) = \frac{1}{2}\sum_{i=1}^{m} f_i^2(\mathbf{x}) = \frac{1}{2}\|F(\mathbf{x})\|^2, \tag{1.1}$$

where the norm is the standard Euclidian norm, $F(\mathbf{x}) = (f_1(\mathbf{x}), f_2(\mathbf{x}), \ldots, f_m(\mathbf{x}))^{\mathsf{T}}$, and $f_i : \mathbb{R}^n \to \mathbb{R}$, $i = 1, \ldots, m$, are general nonlinear twice continuously differentiable functions, but none of their first-order or second-order derivatives is explicitly available. These least-squares optimization problems arise frequently in computational science, engineering, and industry. For example, it is the most common approach for solving a system of nonlinear equations. However, because the function values are often extracted from economic models or physical or chemical experiments or measurements or require expensive and complex computer simulations (for example, Monte Carlo simulations or simulations based on solving differential equations), in practice it is often impossible or impractical (for example, in the case of legacy codes) to obtain their derivatives. Consequently, it is often desirable to treat these functions as black boxes, necessitating that derivative-free algorithms are used to solve (1.1). The goal of these algorithms is to solve (1.1) with the least possible number of function evaluations.

In the literature, there are mainly three classes of derivative-free optimization methods. The first is a class of direct search methods that explore the variable space by sampling points from a predefined class of geometric patterns or that involve some random process [1, 2]. Some of these methods do not assume smoothness of the

*Received by the editors April 8, 2010; accepted for publication (in revised form) October 15, 2010; published electronically December 16, 2010. This material is based upon work supported by the National Science Foundation under grant 1016204.

http://www.siam.org/journals/siopt/20-6/75531.html

†Department of Mathematics, Louisiana State University, 140 Lockett Hall, Baton Rouge, LA 70803-4918 (hozhang@math.lsu.edu, http://www.math.ufl.edu/∼hozhang).

‡Department of Mathematical Sciences, IBM T.J. Watson Research Center, Route 134, P.O. Box 218, Yorktown Heights, NY 10598 (arconn@us.ibm.com).

§Department of Industrial and Systems Engineering, Lehigh University, Harold S. Mohler Laboratory, 200 West Packer Avenue, Bethlehem, PA 18015-1582 (katyascheinberg@gmail.com).

objective function and therefore can be applied to a broad class of problems. But, on the other hand, a relatively large number of function evaluations are often required. The second class of methods combines finite difference techniques with quasi-Newton methods, for instance [19]. These methods are relatively easy to implement using existing software for quasi-Newton methods. However, because finite differences are used to approximate derivatives, this class of methods is not always robust, especially in the presence of noise, which is often the case in real derivative-free applications. The third class of methods requires the sequential minimizations of models, typically quadratic or linear, based upon evaluations of the objective function at sample sets [5, 17, 22, 23]. Some recent research [20] indicates that the third class of trust region model-based methods is frequently superior compared with the first two classes of methods, even for noisy and piecewise-smooth problems. Nevertheless, the first two approaches are still widely used in the engineering community.

The algorithms described in this paper belong to the third class of methods, which is based on modeling the objective function by multivariate interpolation and/or regression, in combination with trust region techniques. However, unlike other model-based methods in the literature, we try to take full advantage of the least-squares problem structure by constructing individual models which interpolate each function contributing to the least-squares objective rather than to the objective itself. This approach is similar to the approach in [4] for constrained derivative-free optimization. In [4] quadratic interpolation models are built for individual constraints and later are combined into a penalty function. As is shown in [4], this approach is more efficient for constrained derivative-free optimization problems than combining the constraints into a penalty function first and building an interpolation model for the unconstrained problem. One reason why such an approach is efficient is the fact that, in derivative-free optimization, typically the most expensive costs are the function evaluations. While constructing and maintaining several models may require more work and storage space, it typically does not require additional function evaluations.

It is well known that in model-based derivative-free optimization, whenever the optimization suggests that a better point is unavailable in the region of interest, it is important to ensure that the model is sufficiently accurate, which in practice usually means that one needs to ensure that the local geometry of the sampling set, on which the models depend, is appropriate. We know that, in the absence of any special structure (for example, the Hessian of the objective function being diagonal), $(n+1)(n+2)/2$ sampling points are required to build a fully quadratic interpolation model. However, this computing cost is unacceptable in many real applications unless n is very small or the function evaluations are inexpensive. Consequently, our approach will leverage Powell's Frobenius norm updating strategy [23] to build at least fully linear models for each of the nonlinear functions in the sum of squares. Therefore, this approach exploits the problem structure while maintaining relatively modest computational costs. More specifically, as suggested in [23], in our numerical experiments we use only $2n+1$ (significantly less than $(n+1)(n+2)/2$, even for moderate n) sampling points for each model in the trust region subproblem. Since the same $2n+1$ interpolation points are used for each function, applying the technique of Powell [23] for updating the inverse of the matrix arising from a particular variational problem results in a computational cost for finding the updates of the coefficients of all the models that remains within $\mathcal{O}(mn)$. Hence, the total cost of building these models at each iteration is only $\mathcal{O}(mn^2)$, except for the occasional iteration which requires a shift of the model origin, whereupon extra $\mathcal{O}(mn^3)$ operations are needed.

In addition to taking advantage of the problem structure by building polynomial interpolation models for each function, in a manner similar to the general approach proposed in [5], we are able to establish the global convergence of our method for structured least-squares minimization using a trust region framework under suitable conditions. To achieve global convergence and fast local convergence, care needs to be taken in the trust region management and the choice of the sample points for the model. Powell [24] suggests maintaining separate trust regions for each purpose. However, it is possible to adaptively update a single trust region radius so as to serve both purposes simultaneously while rigorously maintaining global convergence. Our algorithm is based on a generalized Levenberg–Marquardt method combined with trust region techniques to minimize the least-squares residual of the problem. Local quadratic convergence of our algorithm can be established under some local error bound conditions [26, 27], which are considerably weaker than a nonsingularity assumption on the Jacobian [12, 14, 15, 28]. The study of local convergence is a subject of a separate paper [29]. We focus on the global convergence in this paper.

The paper is organized as follows. In section 2, we introduce our derivative-free algorithm for least-squares minimization (DFLS) along with some definitions. Some basic properties of the interpolation models are given in section 3. Global convergence of this new algorithm is established in section 4. The numerical performances of our algorithm compared with other derivative-free software packages is provided in section 5. Finally, some conclusions are made in section 6.

Notation. Unless otherwise specified, the norm $\|\cdot\|$ is the 2-norm for a vector and the induced 2-norm for a matrix. Given any set $\mathbf{S}$, $|\mathbf{S}|$ denotes the cardinality of $\mathbf{S}$. We let $\mathcal{B}$ denote a closed ball in $\mathbb{R}^n$ and $\mathcal{B}(\mathbf{z}, \Delta)$ denote the closed ball centered at z, with radius $\Delta > 0$. $\mathcal{P}_n^d$ means the space of all polynomials of degree $\leq d$ in $\mathbb{R}^n$. We let $C_{n+1}^1 = n + 1$ and $C_{n+2}^2 = (n + 1)(n + 2)/2$ throughout the paper.

There are some constants used in this paper which are denoted by κ (sometimes with acronyms for the subscripts that are meant to be helpful). We collected their definitions here for convenience. The actual meaning of the constants will become clear when each of them is introduced in the paper.

$\hat{\kappa}_{ef}$, κ_{ef}:	error in the function value
$\hat{\kappa}_{eg}$, κ_{eg}:	error in the gradient
κ_{eH}:	error in the Hessian
κ_H^1 , κ_H^2, and κ_H^3:	associated with the definition of the model Hessian
κ_m:	bound on all the separate models
κ_g:	bound on the gradients of all the separate models
κ_H:	bound on the Hessians of all the separate models
κ_H^ϕ:	bound on the (modified) Hessian of ϕ

2. The algorithm. We first state the definition of a Λ–poised set $\mathbf{Y} \subset \mathbb{R}^n$ from [7]. The definition is as follows.

DEFINITION 2.1. *Let $\Lambda > 0$ and $\mathcal{P}$ be a space of polynomials on $\mathbb{R}^n$ with a basis $\varphi = \{\varphi_0(\mathbf{x}), \varphi_1(\mathbf{x}), \ldots, \varphi_p(\mathbf{x})\}$. Then a set $\mathbf{Y} = \{\mathbf{y}^0, \mathbf{y}^1, \ldots, \mathbf{y}^p\}$ is said to be Λ–poised in $\mathcal{B}$ for $\mathcal{P}$ (in the interpolation sense) if and only if, for any $\mathbf{x} \in \mathcal{B} \subset \mathbb{R}^n$, there exists $\lambda(\mathbf{x}) \in \mathbb{R}^{p+1}$ such that*

$$\sum_{i=0}^{p} \lambda_i(\mathbf{x})\varphi(\mathbf{y}^i) \;=\; \varphi(\mathbf{x}) \qquad \textit{with} \qquad \|\lambda(\mathbf{x})\|_\infty \;\leq\; \Lambda.$$

Other equivalent definitions of a Λ–poised set are given in Chapter 3 of [7]. It is shown in [6] (see also Chapter 3 in [7]) that the above definition does not depend on the choice of the basis $\varphi(\mathbf{x})$ as long as it spans the same space $\mathcal{P}$.

Since we are mostly interested in using models which are defined on the interpolation sets whose cardinality is smaller than that required for fully determined quadratic interpolation, we will also use the definition of a Λ–poised set $\mathbf{Y} \subset \mathbb{R}^n$ for minimum Frobenius norm interpolation.

DEFINITION 2.2. *Let $\Lambda > 0$ and a set $\mathcal{B} \in \mathbb{R}^n$ be given. Let $\varphi = \{\varphi_0(\mathbf{x}), \varphi_1(\mathbf{x}), \ldots, \varphi_{C^2_{n+2}-1}(\mathbf{x})\}$ be the natural basis of monomials of $\mathcal{P}^2_n$ (ordered by degree). A poised set $Y = \{\mathbf{y}^0, \mathbf{y}^1, \ldots, \mathbf{y}^p\}$, with $C^1_{n+1} \le p+1 \le C^2_{n+2}$, is said to be Λ–poised in $\mathcal{B}$ (in the minimum Frobenius norm sense) if and only if, for any $\mathbf{x} \in \mathcal{B}$, there exists a solution $\lambda(\mathbf{x}) \in \mathbb{R}^{p+1}$ of*

$$\min \quad \sum_{j=C^1_{n+1}}^{C^2_{n+2}-1} \left(\sum_{i=0}^{p} \lambda_i(\mathbf{x}) \varphi_j(\mathbf{y}_i) - \varphi_j(\mathbf{x}) \right)^2 \tag{2.1}$$

$$\text{subject to} \quad \sum_{i=0}^{p} \lambda_i(\mathbf{x}) \varphi_j(\mathbf{y}_i) \;=\; \varphi_j(\mathbf{x}), \quad 0 \le j \le n \tag{2.2}$$

such that

$$\|\lambda(\mathbf{x})\|_\infty \;\le\; \Lambda.$$

It is shown in [8] (see also Chapter 5 in [7]) that the solution $\lambda(\mathbf{x})$ to (2.1) and (2.2) is nothing else but the Lagrange polynomial for the set Y with minimum Frobenius norms of the Hessians. The minimum Frobenius norm models are then built based on these Lagrange polynomials. We do not get into the details here, but this connection helps explain why Definition 2.2 is in the minimum Frobenius norm sense.

Note that, unfortunately, in the underdetermined case, the first definition depends on the choice of $\mathcal{P}$, and the second definition depends on the choice of the basis. In the following, we call a given set $\mathbf{Y} \subset \mathcal{B}(\mathbf{z}, \Delta)$, with $C^1_{n+1} \le |\mathbf{Y}| \le C^2_{n+2}$, Λ–poised whenever it is Λ–poised in $\mathcal{B}(\mathbf{z}, \Delta)$ for some polynomial space $\mathcal{P}$ with $\mathcal{P}^1_n \subseteq \mathcal{P} \subseteq \mathcal{P}^2_n$ in the sense of Definition 2.1. On the other hand, it can also be shown that this just described Λ-poisedness is equivalent to $\bar{\Lambda}$-poisedness given by Definition 2.2 using the natural monomial basis, just with a possible different related constant $\bar{\Lambda}$ (see Theorem 5.8 in [8]). Hence, when stating that a set $\mathbf{Y} \subset \mathcal{B}(\mathbf{z}, \Delta)$, with $C^1_{n+1} \le |\mathbf{Y}| \le C^2_{n+2}$, is Λ-poised, one can actually understand it in the sense of either of the two definitions and choose whichever is convenient. Typically, the first definition is more convenient for the purpose of constructing Λ-poised models, while the second is closely aligned to the models we employ in our algorithm.

For the convergence analysis, we will also require the following assumption.

Assumption 2.1. Assume that, given any set $\mathbf{Y} \subset \mathcal{B}(\mathbf{z}, \Delta)$ with $C^1_{n+1} \le |\mathbf{Y}| \le C^2_{n+2}$ and $\mathbf{z} \in \mathbb{R}^n$, we can apply a finite number of substitutions of the points in $\mathbf{Y}$, in fact, at most $|\mathbf{Y}| - 1$, such that the new resultant set is Λ–poised in $\mathcal{B}(\mathbf{z}, \Delta)$ for a polynomial space $\mathcal{P}$ with dimension $|\mathbf{Y}|$ and $\mathcal{P}^1_n \subseteq \mathcal{P} \subseteq \mathcal{P}^2_n$.

To see that this is not unduly restrictive and that numerical procedures can be applied to achieve this, the reader is referred to [6, 7, 8], for example.

Now we explain how the local models are constructed in our derivative-free algorithm. Suppose we have a Λ-poised set $\mathbf{Y} \subset \mathcal{B}(\mathbf{z}, \Delta)$ with $C^1_{n+1} \le |\mathbf{Y}| \le C^2_{n+2}$. For

each $i = 1, \ldots, m$, let $m_i(\mathbf{x}) \in \mathcal{P}_n^2$ be a polynomial interpolating model of $f_i(\mathbf{x})$ on $\mathbf{Y}$. Substituting $f_i(\mathbf{x})$ by $m_i(\mathbf{x})$ in the objective function, we get

$$m(\mathbf{x}) := \frac{1}{2}\sum_{i=1}^{m} m_i^2(\mathbf{x}), \tag{2.3}$$

which is normally a polynomial of degree four. Now taking the second-order Taylor expansion of $m(\mathbf{x})$ at a point $\mathbf{y}$ in terms of variable $\mathbf{s}$, we obtain

$$\frac{1}{2}\mathbf{m}(\mathbf{y})^{\mathsf{T}}\mathbf{m}(\mathbf{y}) + \left(\mathbf{m}(\mathbf{y})^{\mathsf{T}}J(\mathbf{y})\right)\mathbf{s} + \frac{1}{2}\mathbf{s}^{\mathsf{T}}\left(J(\mathbf{y})^{\mathsf{T}}J(\mathbf{y}) + \sum_{i=1}^{m} m_i(\mathbf{y})\nabla^2 m_i\right)\mathbf{s},$$

where

$$\mathbf{m}(\mathbf{y}) = (m_1(\mathbf{y}), m_2(\mathbf{y}), \ldots, m_m(\mathbf{y}))^{\mathsf{T}} \text{ and } J(\mathbf{y}) = (\nabla m_1(\mathbf{y}), \nabla m_2(\mathbf{y}), \ldots, \nabla m_m(\mathbf{y}))^{\mathsf{T}}$$

denote the Jacobian of $\mathbf{m}(\cdot)$. Using this expansion as a guide, for any $\mathbf{y} \in \mathcal{B}(\mathbf{z}, \Delta)$, we define our local quadratic model $\phi(\mathbf{y}, \mathbf{s})$ of $\Phi(\cdot)$ around $\mathbf{y}$ as

$$\phi(\mathbf{y}, \mathbf{s}) = c_\phi(\mathbf{y}) + \mathbf{g}_\phi(\mathbf{y})^{\mathsf{T}}\mathbf{s} + \frac{1}{2}\mathbf{s}^{\mathsf{T}}H_\phi(\mathbf{y})\mathbf{s}, \tag{2.4}$$

where

$$c_\phi(\mathbf{y}) = \frac{1}{2}\mathbf{m}(\mathbf{y})^{\mathsf{T}}\mathbf{m}(\mathbf{y}), \qquad \mathbf{g}_\phi(\mathbf{y}) = J(\mathbf{y})^{\mathsf{T}}\mathbf{m}(\mathbf{y}),$$

and

$$H_\phi(\mathbf{y}) = \begin{cases} J(\mathbf{y})^{\mathsf{T}}J(\mathbf{y}) & \text{if } \|\mathbf{g}_\phi(\mathbf{y})\| \geq \kappa_H^1, \\ J(\mathbf{y})^{\mathsf{T}}J(\mathbf{y}) + \kappa_H^3\|\mathbf{m}(\mathbf{y})\|I & \text{if } \|\mathbf{g}_\phi(\mathbf{y})\| < \kappa_H^1 \text{ and } c_\phi(\mathbf{y}) < \kappa_H^2\|\mathbf{g}_\phi(\mathbf{y})\|, \\ J(\mathbf{y})^{\mathsf{T}}J(\mathbf{y}) + \sum_{i=1}^{m} m_i(\mathbf{y})\,\nabla^2 m_i & \text{otherwise.} \end{cases}$$

Here I denotes the identity matrix, κ_H^1, κ_H^2, and κ_H^3 are positive constants, and $m_i(\cdot) \in \mathcal{P}_n^2$, $i = 1, \ldots, m$, are the polynomial interpolating models of $f_i(\cdot)$ on $\mathbf{Y}$. Typical values of κ_H^1, κ_H^2, and κ_H^3 are 1, 1, and 0.01, respectively. In numerical optimization algorithms, it is not unusual to "convexify" nonconvex problems by adding a multiple of a positive definite Hessian to the second-order terms to make the algorithm more stable and efficient. This is similar to what we do in the middle term of the definition of the Hessian above, and such modifications result in what are termed "regularized Hessians." This regularization serves to prevent a situation when the trust region step could be unnecessarily large along the null space of the Jacobian, which would cause the loss of local quadratic convergence for zero residual problems. In a separate paper [29], we show with this regularization that the local quadratic convergence for zero residual problems can still be maintained under certain weaker nonsingularity assuptions which include many cases even when the columns of the Jacobian are linearly dependent. On the other hand, when the columns of the Jacobian are linearly independent, in the zero residual case, asymptotically this regularization term would be negligible, thereby not affecting the local quadratic convergence in the case of the traditional Gauss–Newton step.

For further motivation for the definition of $H_\phi(\mathbf{y})$, we remark that, if $\mathbf{g}_\phi(\mathbf{y})$ is sufficiently large in magnitude, it is reasonable to assume that the first-order changes

in the least-squares objective dominate locally, and so one may take a Gauss–Newton approximation and ignore the curvature of the residual. This is true even when we do not have a relatively small residual problem locally, i.e., $\mathbf{m}(\mathbf{y})$ may not be small in magnitude. If this first-order change and the residual are relatively small, the second definition can be considered as a sensible compromise, in that some damping via the residual curvature is introduced using the identity (replacing $\pm\nabla^2 m_i$) weighted by the current overall model residual. Furthermore, it can be shown that this regularized Hessian in the middle term will eventually be chosen in the zero residual case to overcome the difficulties brought by possible singularity of the Jacobian. Note the emphasis is on relatively. The second case is triggered by both the size of the residual and the stationarity condition. Otherwise, a genuine second-order model is used. If one considers that the middle expression is approximating the terms involving the Hessian of the m_i's (as a regularization) by the identity matrices, then maybe it is more reasonable to use the l_1-norm for the regularization rather than the l_2-norm. For several reasons, this norm would be advantageous for the trust region when one includes bound constraints.

Our model can also be regarded as an adaptive variance of a standard Levenberg–Marquardt algorithm, which can be considered as a regularized Gauss–Newton method designed for small residual problems with the definition (including, if required, the constant regularization term)

$$H_\phi(\mathbf{y}) = \begin{cases} J(\mathbf{y})^{\mathsf{T}} J(\mathbf{y}) & \text{if the residual is considered small,} \\ J(\mathbf{y})^{\mathsf{T}} J(\mathbf{y}) + \kappa_H^3\, I & \text{otherwise.} \end{cases}$$

We now derive a class of DFLS algorithm. This algorithm takes into account the problem structure, but otherwise it is close in spirit to the framework presented in [6] for global convergence and the detailed algorithm applied in Powell's NEWUOA software [22] for practical efficiency. We note that the details of the steps which ensure that the interpolating set $\mathbf{Y}$ remains well-poised are omitted since they can readily be garnered from [6] and [7]. Throughout the algorithm stated below, we fix the number of points in the sampling set, i.e., $|\mathbf{Y}_k| = N_P$, for all $k \geq 0$, where $N_P \in [C_{n+1}^1, C_{n+2}^2]$ is an integer constant. We denote the resulting iterates by x_k, where k is the iteration number.

A derivative-free algorithm for least-squares minimization (DFLS)

Step 0 **(Initialization)** Choose the starting guess $\mathbf{x}_0$, $0 < \rho_0 \leq \bar{\Delta}_0 \leq \Delta_0 \leq \Delta_{max}$, and N_P, the number of sampling points, with $N_P \geq C_{n+1}^1 = n+1$. Choose an initial set of interpolation points, $\mathbf{Y}_0$, with $\mathbf{x}_0 \in \mathbf{Y}_0 \subset \mathcal{B}(\mathbf{x}_0, \bar{\Delta}_0)$. Choose $\epsilon_\beta \in (0,1)$ and $\beta > 0$. Set $k = 0$.

Step 1 **(Criticality step)** Choose a base point $\mathbf{y}_k \in \mathbf{Y}_k$, and calculate

$$\mathbf{g}_{\phi_k} = J(\mathbf{y}_k)^{\mathsf{T}} \mathbf{m}(\mathbf{y}_k) + H_\phi(\mathbf{y}_k)(\mathbf{x}_k - \mathbf{y}_k), \tag{2.5}$$

where $\mathbf{y}_k \in \mathbf{Y}_k$. If $\|\mathbf{g}_{\phi_k}\| \leq \epsilon_\beta$, let $\bar{\Delta}_k^{(0)} = \bar{\Delta}_k$, possibly modifying $\mathbf{Y}_k$ as needed to make sure $\mathbf{Y}_k$ is Λ-poised in $\mathcal{B}(\mathbf{x}_k, \bar{\Delta}_k)$, where $\bar{\Delta}_k = \min\{\bar{\Delta}_k^{(0)}, \beta\|\mathbf{g}_{\phi_k}\|\}$ and $\mathbf{g}_{\phi_k}$ is recalculated using (2.5) with the new $\mathbf{Y}_k$ if $\mathbf{Y}_k$ has changed. Determine the corresponding interpolation model $\mathbf{m}(\mathbf{y}_k)$. It is shown in Lemma 7.3 of [5] that, unless y_k is a first-order stationary point, this can be achieved in a finite number of steps.

Step 2 **(Step calculation)** Solve the following trust region subproblem

$$\begin{aligned} &\min && \phi_k(\mathbf{d}) \\ &\text{subject to} && \|\mathbf{d}\| \le \Delta_k, \end{aligned} \tag{2.6}$$

where $\phi_k(\mathbf{d}) = \phi(\mathbf{y}_k, (\mathbf{x}_k - \mathbf{y}_k) + \mathbf{d})$ with $\phi(\cdot,\cdot)$ defined by (2.4), to obtain the step[1] $\mathbf{d}_k$.

Step 3 **(Safety step)** This step applies only when $\|\mathbf{d}_k\| < \frac{1}{2}\rho_k$ and $\|\mathbf{g}_{\phi_k}\| > \epsilon_\beta$.

3.1 Let $i = 0$, $\Delta_k^{(0)} = \Delta_k$.

3.2 Choose $\bar{\Delta}_k^{(i)} \in [\rho_k, \Delta_k^{(i)}]$.

3.3 If $\bar{\Delta}_k^{(i)} > \rho_k$, then

If $\mathbf{Y}_k$ is Λ-poised in $\mathcal{B}(\mathbf{x}_k, \bar{\Delta}_k^{(i)})$, then

Let $\Delta_k^{(i+1)} = \max\{\Delta_k^{(i)}/10, \rho_k\}$. $i = i + 1$, go to 3.2.

Else

Let $\Delta_{k+1} = \bar{\Delta}_k^{(i)}, \rho_{k+1} = \rho_k$.

Endif

Else (i.e., $\bar{\Delta}_k^{(i)} = \rho_k$)

If $\mathbf{Y}_k$ is Λ-poised in $\mathcal{B}(\mathbf{x}_k, \bar{\Delta}_k^{(i)})$, then

Let $\Delta_{k+1} = \rho_k/2, \rho_{k+1} = \rho_k/10$.

Else

Let $\Delta_{k+1} = \bar{\Delta}_k^{(i)}, \rho_{k+1} = \rho_k$.

Endif

Endif

Let $\mathbf{x}_{k+1} = \mathbf{x}_k$, and choose $\bar{\Delta}_{k+1} \in [\rho_{k+1}, \Delta_{k+1}]$.

Modify $\mathbf{Y}_k$ to form $\mathbf{Y}_{k+1}$ such that $\mathbf{Y}_{k+1}$ is Λ-poised in $\mathcal{B}(\mathbf{x}_{k+1}, \bar{\Delta}_{k+1})$.

Set $k = k + 1$, go to Step 1.

Step 4 **(Acceptance of the trial step)** This step applies only when $\|\mathbf{d}_k\| \ge \frac{1}{2}\rho_k$ or $\|\mathbf{g}_{\phi_k}\| \le \epsilon_\beta$.

Compute $\Phi(\mathbf{x}_k + \mathbf{d}_k)$ and $r_k := Ared_k/Pred_k$, where the actual reduction is defined by

$$Ared_k = \Phi(\mathbf{x}_k) - \Phi(\mathbf{x}_k + \mathbf{d}_k), \tag{2.7}$$

while the predicted reduction is defined by

$$Pred_k = \phi_k(\mathbf{0}) - \phi_k(\mathbf{d}_k). \tag{2.8}$$

If $r_k > 0$, then $\mathbf{x}_{k+1} = \mathbf{x}_k + \mathbf{d}_k$; otherwise, $\mathbf{x}_{k+1} = \mathbf{x}_k$.

Step 5 **(Trust region radius update)** Set

$$\tilde{\Delta}_{k+1} = \begin{cases} \frac{1}{2}\,\|\mathbf{d}_k\| & \text{if } r_k < 0.1, \\ \max\{\frac{1}{2}\Delta_k,\ \|\mathbf{d}_k\|\} & \text{if } 0.1 \le r_k < 0.7, \\ \max\{\Delta_k,\ 2\|\mathbf{d}_k\|\} & \text{if } r_k \ge 0.7 \end{cases} \tag{2.9}$$

and

$$\Delta_{k+1} = \min\{\max\{\tilde{\Delta}_{k+1},\ \rho_k\}, \Delta_{max}\}. \tag{2.10}$$

If $r_k \ge 0.1$, then

[1] In other words, we build the model at y_k but shift our center to x_k.

Let $\rho_{k+1} = \rho_k$, and choose $\bar{\Delta}_{k+1} \in [\rho_k, \Delta_{k+1}]$.
The interpolating points set $\mathbf{Y}_k$ is updated to take into consideration the new point $\mathbf{x}_{k+1}$ to form $\mathbf{Y}_{k+1} \in \mathcal{B}(\mathbf{x}_{k+1}, \bar{\Delta}_{k+1})$.
Set $k = k + 1$, go to Step 1.
Endif

Step 6 **(Model improvement)** This step applies only when $r_k < 0.1$.
If $\mathbf{Y}_k$ is Λ-poised in $\mathcal{B}(\mathbf{x}_k, \Delta_k)$ and $\Delta_k = \rho_k$, then
Let $\Delta_{k+1} = \rho_k/2$ and $\rho_{k+1} = \rho_k/10$. Choose $\bar{\Delta}_{k+1} \in [\rho_{k+1}, \Delta_{k+1}]$.
The interpolating points set $\mathbf{Y}_k$ is updated to take into consideration the new point $\mathbf{x}_{k+1}$ to form $\mathbf{Y}_{k+1} \in \mathcal{B}(\mathbf{x}_{k+1}, \bar{\Delta}_{k+1})$.
Else
Let $\rho_{k+1} = \rho_k$. Choose $\bar{\Delta}_{k+1} \in [\rho_{k+1}, \Delta_{k+1}]$. Possibly modification of $\mathbf{Y}_k$ is needed to form $\mathbf{Y}_{k+1}$ such that $\mathbf{Y}_{k+1}$ is Λ-poised in $\mathcal{B}(\mathbf{x}_{k+1}, \bar{\Delta}_{k+1})$.
$\Delta_{k+1} = \Delta_k$
Endif
Set $k = k + 1$, go to Step 1.

Let us now discuss the steps of this algorithm. We remark that, in theory, it is a good idea to choose the base point in Step 1 to be the best point available. However, in practice, this involves shifting the bases of the models, which requires $\mathcal{O}(mn^3)$ additional work, while in most steps we are trying to keep the work of building the model to $\mathcal{O}(mn^2)$. The trust region for these methods serves two purposes: it restricts the step size to the neighborhood where the model is assumed to be good, and it also defines the neighborhood in which the points are sampled for the construction of the model. As we already mentioned, Powell [24] suggests using two different trust regions to separate these two roles, and, in some sense, Δ_k and ρ_k identify these two regions. Note that ρ_k is a lower bound of the sampling region radius $\bar{\Delta}_k$, except when a measure of stationarity is close to zero and Step 1 is applied. When the current iterate is far from stationary, it is undesirable to have all the sample points very close to the center of the trust region. However, when the measure of stationarity is close to zero (that is, the current iterate may be close to a stationary point), it is important to ensure that the sampling set is well-poised in a region with a size comparable to that measure to ensure the models will be locally accurate enough. This is exactly the purpose served by the Criticality step (Step 1).

Another important feature of using ρ_k in the algorithm is to prevent the trust region from shrinking too fast before the model becomes sufficiently accurate in the current trust region. In other words, when a "bad" trial trust region point is generated, instead of reducing the trust region radius immediately, the algorithm should first make sure the model is a "good" model, i.e., the sampling points are well-poised in the current trust region. Otherwise, the trust region radius will be reduced too quickly for the wrong reasons, consequently allowing only very small step sizes, making the algorithm, at best, very inefficient. These ideas are implemented in Steps 3, 5, and 6. Notice that, following Powell, we apply a safety step (Step 3) when the iterate is relatively far away from stationarity and the trial trust region step is too small (less than half of the lower bound of the sampling region radius). We do not calculate the true function value at this trial point since, most probably, this would be wasteful as it may neither give a lower function value nor improve the geometry of future sample sets. In addition, the updating procedure of the trust region radius in Step 3 ensures that repeatedly executing this step eventually forces the trust region radius to the

lower bound ρ_k. Hence, Step 3 will finish in a finite number of steps. When ρ_k goes to zero, it also defines a natural stopping criterion for this class of methods.

Theoretically we always have the possibility of building the trust region model at the current iterate, i.e., letting $\mathbf{y}_k = \mathbf{x}_k$ at Step 2. But, as we alluded to earlier, in practice the algorithms do not move the base point at every iteration. However, to avoid unacceptable numerical errors, it may be desirable to move the base point to the current iterate when the trust region step is sufficiently small compared with the distance between the current iterate and the base point. For example, in our implementation, we do so when $\|\mathbf{d}_k\| \leq 0.1\|\mathbf{x}_k - \mathbf{y}_k\|$. For details, the reader might refer to [22]. In general, we expect $\mathbf{y}_k$ to be different from $\mathbf{x}_k$. We also remark that in Step 4, we accept a simple decrease in efficiency, and in Step 5, the trust region management relates to the size of the actual step taken rather than just to the trust region radius. Again, as one would expect, the trust region radius is reduced only when the failure is not a consequence of the poor geometry of the interpolation points. In addition, notice that in Step 5, when a sufficient function reduction is obtained, i.e., when $r_k \geq 0.1$, the algorithm requires only that the next sampling set, Y_{k+1}, is formed by incorporating $\mathbf{x}_{k+1}$, without regard to the well-poisedness of Y_{k+1}. This is desirable since checking that the sampling set is well-poised sometimes requires a significant amount of computation. We do want to avoid extra work as long as the algorithm performs well and global convergence is guaranteed. In practice, a normal procedure in this case is just to exchange $\mathbf{x}_{k+1}$ with the sampling point in Y_k which is farthest away from the base point.

3. Model properties. In this section, we present a few lemmas which provide some basic properties of the interpolation models used in the DFLS algorithm, and these properties will also be referred to in the next section on global convergence.

The following lemma shows, under our assumptions of Λ-poisedness (by either definition) of the sample set $\mathbf{Y}$, an interpolating polynomial on $\mathbf{Y}$ would be at least a local fully linear model. This lemma follows directly from Theorem 5.4 in [7].

LEMMA 3.1. *Given any $\Delta > 0$, $\mathbf{z} \in \mathbb{R}^n$, and $\mathbf{Y} = \{\mathbf{y}^0, \mathbf{y}^1, \ldots, \mathbf{y}^p\} \subset \mathcal{B}(\mathbf{z}, \Delta)$ Λ–poised in $\mathcal{B}(\mathbf{z}, \Delta)$ with $C^1_{n+1} \leq |\mathbf{Y}| \leq C^2_{n+2}$, let $\mathbf{m}(\cdot) \in \mathcal{P}^2_n$ be a on interpolating polynomial of f on $\mathbf{Y}$, i.e.,*

$$\mathbf{m}(\mathbf{y}^i) = f(\mathbf{y}^i),\ i = 1, \ldots, |\mathbf{Y}|.$$

If $f : \mathbb{R}^n \to \mathbb{R}$ is continuously differentiable and ∇f is Lipschitz continuous with Lipschitz constant L in an open set containing $\mathcal{B}(\mathbf{z}, \Delta)$, then, for any $\mathbf{s} \in \mathcal{B}(\mathbf{0}, \Delta)$, we have

$$\|\nabla f(\mathbf{z}+\mathbf{s}) - \nabla \mathbf{m}(\mathbf{z}+\mathbf{s})\| \leq \hat{\kappa}_{eg}(n, \Lambda)\ (\|\nabla^2 \mathbf{m}\| + L)\ \Delta,$$
$$|f(\mathbf{z}+\mathbf{s}) - \mathbf{m}(\mathbf{z}+\mathbf{s})| \leq \hat{\kappa}_{ef}(n, \Lambda)\ (\|\nabla^2 \mathbf{m}\| + L)\ \Delta^2,$$

where $\hat{\kappa}_{eg}$ and $\hat{\kappa}_{ef}$ are positive constants depending only on n and Λ.

Proof. The proof can be found in Theorem 5.4 in [7], and similar proofs can be found in [6]. ∎

Given a Λ-poised set $\mathbf{Y} \subset \mathcal{B}(\mathbf{z}, \Delta)$ with $C^1_{n+1} \leq |\mathbf{Y}| \leq C^2_{n+2}$ for each $i = 1, \ldots, m$, suppose $m_i(\mathbf{x}) \in \mathcal{P}^2_n$ is a polynomial interpolating model of $f_i(\mathbf{x})$ on $\mathbf{Y}$. Then, based on the above lemma, there exist positive constants κ_{eg} and κ_{ef} such that, for any $\mathbf{s} \in \mathcal{B}(\mathbf{0}, \Delta)$,

$$\|\nabla f_i(\mathbf{z}+\mathbf{s}) - \nabla m_i(\mathbf{z}+\mathbf{s})\| \leq \kappa_{eg}\Delta, \tag{3.1}$$
$$|f_i(\mathbf{z}+\mathbf{s}) - m_i(\mathbf{z}+\mathbf{s})| \leq \kappa_{ef}\Delta^2 \tag{3.2}$$

for all $i = 1, \ldots, m$, where κ_{eg} and κ_{ef} are positive constants depending only on n, Λ, F, and $\max\{\|\nabla^2 m_i\|, i = 1, \ldots, m\}$. Hence, if $\max\{\|\nabla^2 m_i\|, i = 1, \ldots, m\} \leq \kappa_H$, a fixed constant,[2] then κ_{eg} and κ_{ef} will be fixed constants, too. In particular, κ_{eg} and κ_{ef} depend neither on $\mathbf{z}$ nor Δ.

Now define $conv(\mathcal{L}_{enl}(\mathbf{x}_0))$ to be the convex hull of $\mathcal{L}_{enl}(\mathbf{x}_0)$ with

$$\mathcal{L}_{enl}(\mathbf{x}_0) = \bigcup_{\mathbf{x} \in \mathcal{L}(\mathbf{x}_0)} \mathcal{B}(\mathbf{x}, \Delta_{max}) \quad \text{and} \quad \mathcal{L}(\mathbf{x}_0) = \{\mathbf{x} \in \mathbb{R}^n : \Phi(\mathbf{x}) \leq \Phi(\mathbf{x}_0)\},$$

where Φ is defined in (1.1). In this paper, we also need the following two assumptions.

Assumption 3.1. Suppose F is twice continuously differentiable and the level set $\mathcal{L}(\mathbf{x}_0) = \{\mathbf{x} \in \mathbb{R}^n : \Phi(\mathbf{x}) \leq \Phi(\mathbf{x}_0)\}$ is bounded.

Assumption 3.2. There exists a constant κ_H, independent of the iteration number k in the DFLS algorithm, such that if m_i, $i = 1, \ldots, m$, is the polynomial interpolating model of f_i on a Λ-poised sampling set $\mathbf{Y}_k$ constructed as in the DFLS algorithm, then

$$\|\nabla^2 m_i\| \leq \kappa_H \tag{3.3}$$

for all $i = 1, \ldots, m$.

Recall that F is assumed twice continuously differentiable, although none of their first-order or second-order derivatives is explicitly available. Based on Assumption 3.1, the following lemmas holds.

LEMMA 3.2. *Under Assumption* 3.1*, there exist positive constants* L_0, L_1*, and* L_2 *such that*

$$\|F(\mathbf{x})\| \leq L_0, \quad \|F(\mathbf{x}) - F(\mathbf{y})\| \leq L_1 \|\mathbf{x} - \mathbf{y}\|, \quad \textit{and} \quad \|\nabla F(\mathbf{x})\| \leq L_1,$$
$$\|\nabla F(\mathbf{x}) - \nabla F(\mathbf{y})\| \leq L_2 \|\mathbf{x} - \mathbf{y}\| \quad \textit{and} \quad \|\nabla^2 f_i(\mathbf{x})\| \leq L_2, \quad i = 1, \ldots, m$$

for any $\mathbf{x}, \mathbf{y} \in conv(\mathcal{L}_{enl}(\mathbf{x}_0))$.

Proof. Since $\mathcal{L}(\mathbf{x}_0)$ is bounded by Assumption 3.1 and $\Delta_{max} > 0$ is a constant, we know $\mathcal{L}_{enl}(\mathbf{x}_0)$ is bounded. Hence, its convex hull $conv(\mathcal{L}_{enl}(\mathbf{x}_0))$ is also bounded. Then the results follow from $f_i : \mathbb{R}^n \to \mathbb{R}, i = 1, \ldots, m$, being twice continuously differentiable. ∎

Now, with Assumptions 3.1 and 3.2, we show the models used in the DFLS algorithm would satisfy the following bounds.

LEMMA 3.3. *Under Assumptions* 3.1 *and* 3.2*, there exist constants* κ_m *and* κ_g*, independent of* k*, such that if* m_i*,* $i = 1, \ldots, m$*, is the polynomial interpolating model of* f_i *on a* Λ*-poised sampling set* $\mathbf{Y}_k$ *constructed as in the DFLS algorithm, then, for any* $\mathbf{y} \in \mathbf{Y}_k$*,*

$$|m_i(\mathbf{y})| \leq \kappa_m \quad \textit{and} \quad \|\nabla m_i(\mathbf{y})\| \leq \kappa_g \tag{3.4}$$

for all $i = 1, \ldots, m$*. Hence there exists a constant* κ_H^ϕ *such that*

$$\|H_\phi(\mathbf{y})\| \leq \kappa_H^\phi \tag{3.5}$$

for any $\mathbf{y} \in \mathbf{Y}_k$.

[2]By "fixed constant", we mean to emphasize that these constants will stay the same throughout a particular instance of the algorithm, i.e., for all iterations.

Proof. Fix any index $i \in [1, m]$. From the DFLS algorithm, we have

$$\mathbf{Y}_k \subseteq \mathcal{L}_{enl}(\mathbf{x}_0) \tag{3.6}$$

for all $k \geq 0$. Since $f_i(\mathbf{y}) = m_i(\mathbf{y})$ for any $\mathbf{y} \in \mathbf{Y}_k$, it follows from Lemma 3.2 that

$$|m_i(\mathbf{y})| \leq \kappa_m := L_0$$

for any $\mathbf{y} \in \mathbf{Y}_k$. Now, by Assumption 3.2, $\bar{\mathbf{Y}}_k$ being Λ-poised in $\mathcal{B}(\mathbf{x}_k, \bar{\Delta}_k)$, $\mathbf{y} \in \mathbf{Y_k} \subset \mathcal{B}(\mathbf{x}_k, \bar{\Delta}_k)$, and (3.1), we have

$$\|\nabla f_i(\mathbf{y}) - \nabla m_i(\mathbf{y})\| \leq \kappa_{eg} \bar{\Delta}_k$$

with κ_{eg} a fixed constant. Therefore, by (3.6), we know

$$\|\nabla m_i(\mathbf{y})\| \leq \max_{\mathbf{x} \in \mathcal{L}_{enl}(\mathbf{x}_0)} \|\nabla f_i(\mathbf{y})\| + \kappa_{eg} \bar{\Delta}_k.$$

Then it follows from Lemma 3.2 and $\bar{\Delta}_k \leq \Delta_{max}$ that

$$\|\nabla m_i(\mathbf{y})\| \leq \kappa_g := L_1 + \kappa_{eg} \Delta_{max}.$$

Hence, by the definition of $H_\phi(\cdot)$, Assumption 3.2, (3.4), and (3.5) hold. ∎

In our numerical experiments, the models are updated by taking the minimum Frobenius norm change of the Hessian since our software is based upon Powell's approach in [23], and Assumption 3.2 is guaranteed in practice by a safeguard condition in the code. For example, in practice when the Frobenius norm of the current accumulated Hessian is too large, one could restart building the Hessian with the minimum Frobenius norm. However, the following lemma indicates that, for the interpolation models with minimum norm of the Hessian, Assumption 3.2 would hold automatically without any safeguard condition needed in the code, which in turn implies that choosing the Hessian with minimum norm, i.e., choosing the "flattest" interpolating polynomial, to emulate a Gauss–Newton method, might be even more preferable in our case for a least-squares problem.

LEMMA 3.4. *Suppose m_i, $i = 1, \ldots, m$, is the polynomial interpolating model of f_i on a Λ-poised sampling set $\mathbf{Y}_k$ constructed as in the k-th iteration of the DFLS algorithm satisfying*

$$\begin{aligned} m_i(\mathbf{x}) = \operatorname{Arg\,min}_q \quad & \|\nabla^2 q(\mathbf{x})\|_F \\ \text{subject to} \quad & q(\mathbf{y}) = f_i(\mathbf{y}) \quad \textit{for all } \mathbf{y} \in \mathbf{Y}_k. \end{aligned} \tag{3.7}$$

Then there exists a constant κ_H, independent of k, such that (3.3) *holds.*

Proof. The lemma is a direct consequence of Theorem 5.7 in [7]. ∎

Notice that Lemma 3.4 proves that the Hessian of minimum Frobenius norm models remains bounded. In our implementation, following Powell's ideas from [23], we use the models with the smallest Frobenius norm of the update. To show rigorously that the Hessian remains bounded for these models, one needs to impose certain assumptions on the old model (whose Hessian is being updated). We omit this discussion here since, for the global convergence results, it is sufficient just to bound the norm of the Hessian. Hence, if a model based on the minimum Frobenius norm of the update fails to have a small norm, theoretically we can always replace it with the minimum Frobenius norm model.

4. Global convergence. In this section we establish the global convergence of the DFLS algorithm. We will need the following lemma, which can be obtained directly from the well-known result (due to Powell [25]; see also p. 125 of [3]). We omit its proof here.

LEMMA 4.1. *Let* $\mathbf{d}_k$ *be the solution of the trust region subproblem* (2.6) *Then*

$$\phi_k(\mathbf{0}) - \phi_k(\mathbf{d}_k) \geq \frac{1}{2}\|\mathbf{g}_{\phi_k}\| \min\left\{\Delta_k, \frac{\|\mathbf{g}_{\phi_k}\|}{\|H_\phi(\mathbf{y}_k)\|}\right\}, \tag{4.1}$$

where $\mathbf{g}_{\phi_k}$ *is defined by* (2.5).

With regard to Step 1 (Criticality step) in the DFLS algorithm, we have the following lemma. It is proved in Lemma 5.1 of [5].

LEMMA 4.2. *If* $\|\nabla\Phi(\mathbf{x}_k)\| \neq 0$, *Step* 1 *(Criticality step) in the DFLS algorithm terminates in a finite number of improvement steps.*

For the function value reduction ratio $r_k = Ared_k/Pred_k$ defined in Step 4 of the DFLS algorithm, we have the following lemma which corresponds to a relatively standard result[3] (for example, Theorem 6.4.2 in [3]) modified for our context.

LEMMA 4.3. *Under Assumptions* 3.1 *and* 3.2, *suppose* $\mathbf{Y}_k \subset \mathcal{B}(\mathbf{x}_k, \Delta_k)$ *is* Λ*-poised. Then, for any* $\epsilon > 0$, *there exists a positive constant* c_ϵ, *independent of* k, *such that, if* $\Delta_k \leq c_\epsilon\|\mathbf{g}_{\phi_k}\|$, *then*

$$|r_k - 1| \leq \epsilon. \tag{4.2}$$

Proof. First we prove that the model $\phi(\mathbf{y}_k, \mathbf{s} - \mathbf{y_k})$ which is minimized on the trust region step is a sufficiently accurate approximation of the least-squares function $\Phi(\mathbf{s})$. After we achieve this, the rest of the proof follows the standard technique of trust region method convergence theory. For any $\mathbf{y}_k \in \mathbf{Y}_k$ and $\mathbf{s} \in \mathbb{R}^n$, it follows from (1.1), (2.4), (3.1), Lemmas 3.2 and 3.3, and $f_i(\mathbf{y}_k) = m_i(\mathbf{y}_k)$, for $i = 1, \ldots, m$, that

(4.3)

$$\begin{aligned}
&\Phi(\mathbf{s}) - \phi(\mathbf{y}_k, \mathbf{s} - \mathbf{y_k}) \\
&\quad = \Phi(\mathbf{s}) - \left[c_\phi(\mathbf{y}_k) + \mathbf{g}_\phi(\mathbf{y}_k)^{\mathsf{T}}(\mathbf{s} - \mathbf{y}_k) + \frac{1}{2}(\mathbf{s} - \mathbf{y}_k)^{\mathsf{T}} H_\phi(\mathbf{y}_k)(\mathbf{s} - \mathbf{y}_k)\right] \\
&\quad \leq \frac{1}{2}\|F(\mathbf{s})\|^2 - \left[\frac{1}{2}\|\mathbf{m}(\mathbf{y}_k)\|^2 + \left(J(\mathbf{y}_k)^{\mathsf{T}}\mathbf{m}(\mathbf{y}_k)\right)^{\mathsf{T}}(\mathbf{s} - \mathbf{y}_k)\right] + \kappa_H^\phi\|\mathbf{s} - \mathbf{y}_k\|^2 \\
&\quad = \frac{1}{2}\|F(\mathbf{s})\|^2 - \left[\frac{1}{2}\|F(\mathbf{y}_k)\|^2 + \left(J(\mathbf{y}_k)^{\mathsf{T}}F(\mathbf{y}_k)\right)^{\mathsf{T}}(\mathbf{s} - \mathbf{y}_k)\right] + \kappa_H^\phi\|\mathbf{s} - \mathbf{y}_k\|^2 \\
&\quad \leq F(\mathbf{y}_k)^{\mathsf{T}}\left[\nabla F(\mathbf{y}_k) - J(\mathbf{y}_k)\right](\mathbf{s} - \mathbf{y}_k) + c_1\|\mathbf{s} - \mathbf{y}_k\|^2 \\
&\quad \leq c_2\Delta_k\|\mathbf{s} - \mathbf{y}_k\| + c_1\|\mathbf{s} - \mathbf{y}_k\|^2,
\end{aligned}$$

where $c_1 > 0$ and $c_2 > 0$ are appropriately chosen constants. Note that for the above last inequality, we used the fact that (3.1) holds with κ_{eg} being a fixed constant. Now substituting $\mathbf{s}$ in (4.3) by $\mathbf{x}_k$ and $\mathbf{x}_k + \mathbf{d}_k$, respectively, since $\mathbf{y}_k \in \mathbf{Y}_k \subset \mathcal{B}(\mathbf{x}_k, \bar{\Delta}_k) \subseteq \mathcal{B}(\mathbf{x}_k, \Delta_k)$ and $\|\mathbf{d}_k\| \leq \Delta_k$, we have

$$\begin{aligned}
\Phi(\mathbf{x}_k) - \phi_k(\mathbf{0}) &= \Phi(\mathbf{x}_k) - \phi(\mathbf{y}_k, \mathbf{x}_k - \mathbf{y_k}) \\
&\leq c_2\Delta_k\|\mathbf{x}_k - \mathbf{y}_k\| + c_1\|\mathbf{x}_k - \mathbf{y}_k\|^2 \\
&\leq c\Delta_k^2
\end{aligned}$$

[3]This result states that, supposing the sampling set has good geometry, when Δ_k becomes small compared with the criticality criteria, the overall model is extremely good relatively.

and

$$\begin{aligned}\Phi(\mathbf{x}_k+\mathbf{d}_k)-\phi_k(\mathbf{d}_k) &= \Phi(\mathbf{x}_k+\mathbf{d}_k)-\phi(\mathbf{y}_k,\mathbf{x}_k-\mathbf{y_k}+\mathbf{d}_k)\\ &\le c_2\Delta_k\|\mathbf{x}_k-\mathbf{y}_k+\mathbf{d}_k\|+c_1\|\mathbf{x}_k-\mathbf{y}_k+\mathbf{d}_k\|^2\\ &\le c\Delta_k^2,\end{aligned}$$

where again $c_1>0$ and $c_2>0$ are appropriately chosen constants and $c=c_1+c_2$. The above inequalities together with (4.1) imply

$$\begin{aligned}|r_k-1| &= \left|\frac{\Phi(\mathbf{x}_k)-\Phi(\mathbf{x}_k+\mathbf{d}_k)}{\phi_k(\mathbf{0})-\phi_k(\mathbf{d}_k)}-1\right|\\ &\le 4c\frac{\Delta_k^2}{\|\mathbf{g}_{\phi_k}\|\min\left\{\Delta_k,\frac{\|\mathbf{g}_{\phi_k}\|}{\|H_\phi(\mathbf{y}_k)\|}\right\}}.\end{aligned}$$

Since $\|H_\phi(\mathbf{y}_k)\|\le\kappa_H^\phi$ by (3.5), (4.2) holds with $c_\epsilon=\min\{\epsilon/4c,1/\kappa_H^\phi\}$. ∎

The following lemma plays the role of Theorem 6.4.3 in [3]. It states that if one supposes that the criticality measure is bounded away from zero, then so is the trust region radius.

LEMMA 4.4. *Under Assumptions* 3.1 *and* 3.2*, suppose there exists a constant* $\delta_1>0$ *with* $\|\mathbf{g}_{\phi_k}\|\ge\delta_1$ *for all* k*. Then there exists a constant* $\delta_2>0$ *such that, for all* k*,*

$$\Delta_k\ge\delta_2.$$

Proof. The proof again follows the standard technique for the analysis of trust region methods, with an additional complication introduced by the interconnected updates to the trust region lower bounds ρ_i. We prove this lemma by way of contradiction. Suppose there exists an infinite subsequence $\{k_i\}$ such that

$$\lim_{i\to\infty}\Delta_{k_i}=0.$$

By the DFLS algorithm, we know $\rho_k\le\Delta_k$ for all k. Hence,

$$\lim_{i\to\infty}\rho_{k_i}=0. \tag{4.4}$$

Without loss of generality, we can choose a subsequence such that $\rho_{k_i}<\rho_{k_i-1}$. So, by Step 3 or Step 6 of the DFLS algorithm, we know that

$$\rho_{k_i}=\rho_{k_i-1}/10,\quad \Delta_{k_i}=\rho_{k_i-1}/2<\Delta_{k_i-1},\quad \|\mathbf{d}_{k_i-1}\|\le\rho_{k_i-1}, \tag{4.5}$$

and $\mathbf{Y}_{k_i-1}\subset\mathcal{B}(\mathbf{x}_{k_i-1},\rho_{k_i-1})\subseteq\mathcal{B}(\mathbf{x}_{k_i-1},\Delta_{k_i-1})$ is Λ-poised. It also follows from the DFLS algorithm that $\{\rho_k\}$ is a nonincreasing sequence. Hence, by (4.4) and (4.5),

$$\lim_{i\to\infty}\rho_{k_i-1}=\lim_{i\to\infty}\|\mathbf{d}_{k_i-1}\|=0. \tag{4.6}$$

Since $\mathbf{d}_{k_i-1}$ is a solution of the trust region subproblem (2.6), by Lemma 4.1,

$$\begin{aligned}\|\mathbf{d}_{k_i-1}\| &\ge \min\{\|\mathbf{g}_{\phi_{k_i-1}}\|/\|H_\phi(\mathbf{y}_{k_i-1})\|,\ \Delta_{k_i-1}\}\\ &\ge \min\{\delta_1/\kappa_H^\phi,\Delta_{k_i-1}\},\end{aligned} \tag{4.7}$$

where $\kappa_H^\phi > 0$ is the constant in (3.5) and we used the fact that $\mathbf{d}_{k_i-1}$ is a solution of the trust region subproblem (2.6) with, by definition,

$$\begin{aligned}\phi_{k_i-1}(\mathbf{d}) &= \phi(\mathbf{y}_{k_i-1}, (\mathbf{x}_{k_i-1} - \mathbf{y}_{k_i-1}) + \mathbf{d}) \\ &= c_\phi(\mathbf{y}_{k_i-1}) + \mathbf{g}_\phi(\mathbf{y}_{k_i-1})^\mathsf{T}((\mathbf{x}_{k_i-1} - \mathbf{y}_{k_i-1}) + \mathbf{d}) \\ &\quad + \frac{1}{2}((\mathbf{x}_{k_i-1} - \mathbf{y}_{k_i-1}) + \mathbf{d})^\mathsf{T} H_\phi(\mathbf{y}_{k_i-1})((\mathbf{x}_{k_i-1} - \mathbf{y}_{k_i-1}) + \mathbf{d}) \\ &= \phi_{k_i-1}(\mathbf{0}) + (\mathbf{g}_\phi(\mathbf{y}_{k_i-1}) + H_\phi(\mathbf{y}_{k_i-1})(\mathbf{x}_{k_i-1} - \mathbf{y}_{k_i-1}))^\mathsf{T}\mathbf{d} + \frac{1}{2}\mathbf{d}^\mathsf{T} H_\phi(\mathbf{y}_{k_i-1})\mathbf{d} \\ &= \phi_{k_i-1}(\mathbf{0}) + \mathbf{g}_{\phi_{k_i-1}}^\mathsf{T}\mathbf{d} + \frac{1}{2}\mathbf{d}^\mathsf{T} H_\phi(\mathbf{y}_{k_i-1})\mathbf{d},\end{aligned}$$

where, by definition of (2.5),

$$\mathbf{g}_{\phi_{k_i-1}} = \mathbf{g}_\phi(\mathbf{y}_{k_i-1}) + H_\phi(\mathbf{y}_{k_i-1})(\mathbf{x}_{k_i-1} - \mathbf{y}_{k_i-1}).$$

So $\mathbf{d}_{k_i-1}$ is a solution of the following problem:

$$\begin{aligned} \min \quad & \mathbf{g}_{\phi_{k_i-1}}^\mathsf{T}\mathbf{d} + \frac{1}{2}\mathbf{d}^\mathsf{T} H_\phi(\mathbf{y}_{k_i-1})\mathbf{d} \\ \text{subject to} \quad & \|\mathbf{d}\| \le \Delta_{k_i-1}. \end{aligned}$$

Now suppose $\mathbf{d}_{k_i-1} < \Delta_{k_i-1}$, and we have $H_\phi(\mathbf{y}_{k_i-1})\mathbf{d}_{k_i-1} + \mathbf{g}_{\phi_{k_i-1}} = \mathbf{0}$. Hence

$$\|\mathbf{g}_{\phi_{k_i-1}}\| = \|H_\phi(\mathbf{y}_{k_i-1})\mathbf{d}_{k_i-1}\| \le \|H_\phi(\mathbf{y}_{k_i-1})\|\|\mathbf{d}_{k_i-1}\|$$

So $\|\mathbf{d}_{k_i-1}\| \ge \|\mathbf{g}_{\phi_{k_i-1}}\|/\|H_\phi(\mathbf{y}_{k_i-1})\|$ in the case of $||\mathbf{d}_{k_i-1}|| < \Delta_{k_i-1}$. Hence, in general,

$$\|\mathbf{d}_{k_i-1}\| \ge \min\{\|\mathbf{g}_{\phi_{k_i-1}}\|/\|H_\phi(\mathbf{y}_{k_i-1})\|, \Delta_{k_i-1}\}.$$

Now it follows from (4.6), (4.7), and $\|\mathbf{d}_{k_i-1}\| \le \Delta_{k_i-1}$ that

$$\|\mathbf{d}_{k_i-1}\| = \Delta_{k_i-1} \tag{4.8}$$

for all large i. Hence

$$\lim_{i\to\infty} \Delta_{k_i-1} = 0. \tag{4.9}$$

Since $\mathbf{Y}_{k_i-1} \subset \mathcal{B}(\mathbf{x}_{k_i-1}, \Delta_{k_i-1})$ is Λ-poised, it follows from $\|\mathbf{g}_{\phi_{k_i-1}}\| \ge \delta_1 > 0$ and Lemma 4.3 that there exists a constant ϵ_{δ_1} such that if $\Delta_{k_i-1} \le \epsilon_{\delta_1}$, then $|r_{k_i-1} - 1| \le 0.3$. Hence, by (4.9), we have $r_{k_i-1} \ge 0.7$ for all large i. This, together with $\|\mathbf{d}_{k_i-1}\| = \Delta_{k_i-1} > \frac{1}{2}\rho_{k_i-1}$ (using (4.5)), imply that, for large i, Steps 3 and 6 will not be involved at the $(k_i - 1)$-th iteration. Therefore, from the DFLS algorithm, $r_{k_i-1} \ge 0.7$ and (4.8), we have

$$\Delta_{k_i} = \min\{\max\{\tilde{\Delta}_{k_i},\ \rho_{k_i-1}\}, \Delta_{max}\} \ge \max\{2\|\mathbf{d}_{k_i-1}\|,\ \rho_{k_i-1}\} = 2\Delta_{k_i-1} > \Delta_{k_i-1}$$

for all large i. This contradicts the middle expression in (4.5). ∎

The following lemma establishes a relationship between $\|\mathbf{g}_{\phi_k}\|$ and $\|\nabla\phi(\mathbf{x}_k)\|$.

LEMMA 4.5. *Under Assumptions* 3.1 *and* 3.2, *for any subsequence* $\{\mathbf{x}_{k_i}\}$ *generated by the DFLS algorithm, if*

$$\lim_{i\to\infty} \|\mathbf{g}_{\phi_{k_i}}\| = 0, \tag{4.10}$$

then

$$\lim_{i\to\infty} \|\nabla\Phi(\mathbf{x}_{k_i})\| = 0. \tag{4.11}$$

Proof. By (4.10) and Step 1 (Criticality step) of the DFLS algorithm, we know $\mathbf{Y}_{k_i} \subset \mathcal{B}(\mathbf{x}_{k_i}, \bar{\Delta}_{k_i}) \subseteq \mathcal{B}(\mathbf{x}_{k_i}, \beta\|\mathbf{g}_{\phi_{k_i}}\|)$ is Λ-poised for all large i. So, by Lemmas 3.2 and 3.3, (2.5), and (3.1), we have

$$\begin{aligned}
\|\nabla\Phi(\mathbf{x}_{k_i}) - \mathbf{g}_{\phi_{k_i}}\| &= \left\|\nabla\Phi(\mathbf{x}_{k_i}) - J(\mathbf{y}_{k_i})^{\mathsf{T}}\mathbf{m}(\mathbf{y}_{k_i}) - H_\phi(\mathbf{y}_{k_i})(\mathbf{x}_{k_i} - \mathbf{y}_{k_i})\right\| \\
&= \left\|\nabla F(\mathbf{x}_{k_i})^{\mathsf{T}}F(\mathbf{x}_{k_i}) - J(\mathbf{y}_{k_i})^{\mathsf{T}}F(\mathbf{y}_{k_i}) - H_\phi(\mathbf{y}_{k_i})(\mathbf{x}_{k_i} - \mathbf{y}_{k_i})\right\| \\
&\le \left\|\nabla F(\mathbf{x}_{k_i})^{\mathsf{T}}F(\mathbf{x}_{k_i}) - \nabla F(\mathbf{y}_{k_i})^{\mathsf{T}}F(\mathbf{y}_{k_i})\right\| \\
&\quad + \left\|\nabla F(\mathbf{y}_{k_i})^{\mathsf{T}}F(\mathbf{y}_{k_i}) - J(\mathbf{y}_{k_i})^{\mathsf{T}}F(\mathbf{y}_{k_i})\right\| + \kappa_H^\phi\|\mathbf{x}_{k_i} - \mathbf{y}_{k_i}\| \\
&\le L_0\|\nabla F(\mathbf{y}_{k_i}) - J(\mathbf{y}_{k_i})\| + c_1\|\mathbf{x}_{k_i} - \mathbf{y}_{k_i}\| \\
&\le c_2\|\mathbf{x}_{k_i} - \mathbf{y}_{k_i}\| \le c_2\beta\|\mathbf{g}_{\phi_{k_i}}\|
\end{aligned}$$

with c_1 and c_2 properly chosen constants. Hence (4.11) holds. ∎

Now we prove the following global convergence theorem.

THEOREM 4.6. *Under Assumptions* 3.1 *and* 3.2*, the sequence* $\{\mathbf{x}_k\}$ *generated by the DFLS algorithm satisfies*

$$\liminf_{k\to\infty} \|\nabla\Phi(\mathbf{x}_k)\| = \liminf_{k\to\infty} \|\nabla F(\mathbf{x}_k)^{\mathsf{T}}F(\mathbf{x}_k)\| = 0. \tag{4.12}$$

Proof. First we prove that there exists a subsequence $\{\mathbf{x}_{k_i}\}$ generated by the DFLS algorithm such that (4.10) holds. Now suppose (4.10) does not hold, i.e., there exists a $\delta_1 > 0$ such that

$$\|\mathbf{g}_{\phi_k}\| > \delta_1 \tag{4.13}$$

for all large k. Then, by Lemma 4.4, there exists a constant $\delta_2 > 0$, independent of k, such that

$$\Delta_k \ge \delta_2. \tag{4.14}$$

Now define

$$\mathcal{I} = \{k : r_k \ge 0.1\}.$$

Then, since $\Phi(\mathbf{x})$ is bounded below by zero, it follows from the DFLS algorithm, Lemmas 3.3 and 4.1, and (4.13) that

$$\begin{aligned}
+\infty &> \sum_{k=0}^{\infty}[\Phi(\mathbf{x}_k) - \Phi(\mathbf{x}_{k+1})] \\
&\ge \sum_{k\in\mathcal{I}}[\Phi(\mathbf{x}_k) - \Phi(\mathbf{x}_{k+1})] \\
&\ge \sum_{k\in\mathcal{I}} 0.1\ Pred_k \ge \sum_{k\in\mathcal{I}} \frac{0.1}{2}\,\delta_1 \min\{\Delta_k, \delta_1/\kappa_H^\phi\}.
\end{aligned}$$

Hence, if $|\mathcal{I}| = \infty$,

$$\lim_{k\to\infty, k\in\mathcal{I}} \Delta_k = 0. \tag{4.15}$$

On the other hand, if $|\mathcal{I}| < \infty$, i.e., $r_k < 0.1$ for all sufficiently large k, by the mechanisms of Steps 3, 5, and 6, we know that $\mathbf{Y}_{k+1}$ will be Λ-poised whenever Step 3 or Step 6 is invoked, Δ_k is not increased in Step 5, and Δ_k is reduced in Steps 3 and 6 whenever $\mathbf{Y}_k$ is Λ-poised. Hence Δ_k cannot be increased for sufficiently large k, and we can conclude that

$$\lim_{k\to\infty} \Delta_k = 0. \tag{4.16}$$

Now either (4.15) or (4.16) contradicts (4.14). Hence, (4.10) holds. Then (4.12) follows from Lemma 4.5. ∎

5. Numerical experiments. In this section, we would like to compare the practical performance of the DFLS algorithm with the following two codes:

- LMDIF [19]: A version, developed in 1980 by Garbow, Hillstrom, and Moré, of the Levenberg–Marquardt algorithm that uses finite (forward) differences for approximating the gradients of the sum of the squares objective function.[4]
- NEWUOA [22]: Software for unconstrained optimization without derivatives, developed by Powell in 2004.

LMDIF and NEWUOA belong to two fundamentally different classes of algorithms that do not use explicit derivatives. One class of methods uses finite difference to approximate the derivative, whereas the other uses polynomial interpolations to build approximate models of the problem.

NEWUOA is derivative-free software that was initially developed for general unconstrained optimization. An extension to accommodate simple bounds, BOBYQA, has been developed by Powell [21]. Although NEWUOA is applicable to minimizing least-squares objectives, it does not make use of the problem structure. The purpose of, nevertheless, using it as a comparison code is to indicate that, perhaps not surprisingly, the performances of this type of algorithm in the case of least-squares can be greatly improved by taking advantage of the problem's structure. On the other hand, LMDIF is a base code that serves to indicate the superiority of the approach in [22] for building local quadratic models for each function in the the least-squares problem instead of using finite difference to approximate the gradient.

Our implementation of the DFLS algorithm is based on the same framework as that provided by NEWUOA. This is partly because the details in NEWUOA are carefully and intelligently chosen and partly because it makes the comparison between it and DFLS more meaningful. In our experiments with DFLS, we chose $N_P = 2n+1$ sampling points. We applied the same polynomial interpolation techniques as those implemented in NEWUOA but on each of the nonlinear functions f_i, $i = 1, \ldots, m$. The model m_i interpolates f_i for all $i = 1, \ldots, m$ at the $2n+1$ sampling points. Hence, the degree of m_i, $i = 1, \ldots, m$, should be at least 2. The remaining freedom in defining m_i is taken up by minimizing the Frobenius norm of the change to the Hessian of m_i as in [23]. Since the sampling points for all the functions f_i, $i = 1, \ldots, m$, are the same,

[4] As we already mentioned, the Levenberg–Marquardt algorithm is essentially a modified Gauss–Newton algorithm, where the modification can be thought of as a regularization of $J^\top J$ with a parameter scaled identity matrix. Although the motivation is rather different, if one associates the parameter with a trust region radius, it is essentially a trust region Gauss–Newton method.

the amount of work for calculating the updates of the coefficients of all the models can be kept within $\mathcal{O}(m(N_P + n)) \approx \mathcal{O}(mn)$, and therefore, discounting the occasional origin shifts, the work of building the models per iteration can be kept within $\mathcal{O}(mn^2)$ (see [23] for details). On the other hand, DFLS requires more memory compared with NEWUOA. In fact, DFLS needs to build and store the individual models $m_i(\mathbf{x})$ for $i = 1, \ldots, m$. This leads to $(m-1)(1+n+n(n+1)/2) = \mathcal{O}(mn^2)$ more storage and arithmetic operations compared with NEWUOA. This is the essential cost of exploiting the structure. In addition, $\mathcal{O}(mn)$ auxiliary memory may be needed for the implementation. In NEWUOA, Powell developed an efficient scheme for selecting the sampling points and for updating the models accurately and efficiently. Many numerical rounding error and stability issues, whose control are crucial for the success of any software, are addressed in [19, 23] and the references therein, and the reader is encouraged to obtain more details there.

All three codes were written in Fortran and were compiled with F77 on an IBM ThinkPad T40 laptop computer. In LMDIF, it is admissible for the variables to be scaled internally. All the parameters are set to the default values except the step length parameter for the forward-difference approximation, which is specified later in the section. In NEWUOA, we set the number of interpolating points to the default number recommended by the code, namely, $2n + 1$. In both NEWUOA and DFLS, the initial trust region radius is set to 1.0. All algorithms stop when the required stopping conditions are satisfied or the algorithms stop internally for numerical reasons. We agree with the point of view taken by Moré and Wild [20] that, in the real application of derivative-free optimization, there are often relatively limited computational budgets available. Hence users are often interested to know the ratio of the obtained reduction in the function value as a function of the number of function evaluations. Consequently, in this section, we apply the same procedure as that used in [20] to evaluate the behavior of the different solvers. All three codes were tested on a collection of benchmark problems, including both problems with noise and problems without noise. The test problem set $\mathcal{P}$ used is proposed in the paper [20], and the source code for these problems can be downloaded from Moré's Web page [18]. This benchmark test problem set comprises 22 types of nonlinear least-squares problems from the CUTEr [13] problem library. Because of the different combinations of starting points and the number of variables for each problem, the benchmark set $\mathcal{P}$ has a total of 53 problems, and each of the 22 types of nonlinear least-squares problems is fairly represented. For more details on how this problem set was formulated, the reader can refer to [20]. As in [20], we also use the following convergence condition:

$$\Phi(\mathbf{x}_0) - \Phi(\mathbf{x}_k) \geq (1-\tau)(\Phi(\mathbf{x}_0) - \Phi_L), \tag{5.1}$$

where $0 \leq \tau < 1$ is a tolerance and Φ_L is the smallest function value obtained by any solver within the same maximum computational budget. In our numerical experiments, we set the maximum computational budget to be 50 (simplex) gradients and investigate the behavior of different solvers within this computational budget. That is, for each problem with dimension n_p, the maximum number of function evaluations allowed for each solver is $50(n_p + 1)$, which is very reasonable in real applications. Then, given a test problem set $\mathcal{P}$, the performances of the different solvers can be measured by the percentage of problems that can be solved (for a given a tolerance τ) within a certain number of (simplex) gradient evaluations. Now, by defining $t_{p,s}$ to be the number of function evaluations needed for solver s to satisfy (5.1) for a problem $p \in \mathcal{P}$ with dimension n_p, the percentage of problems solved as a function of

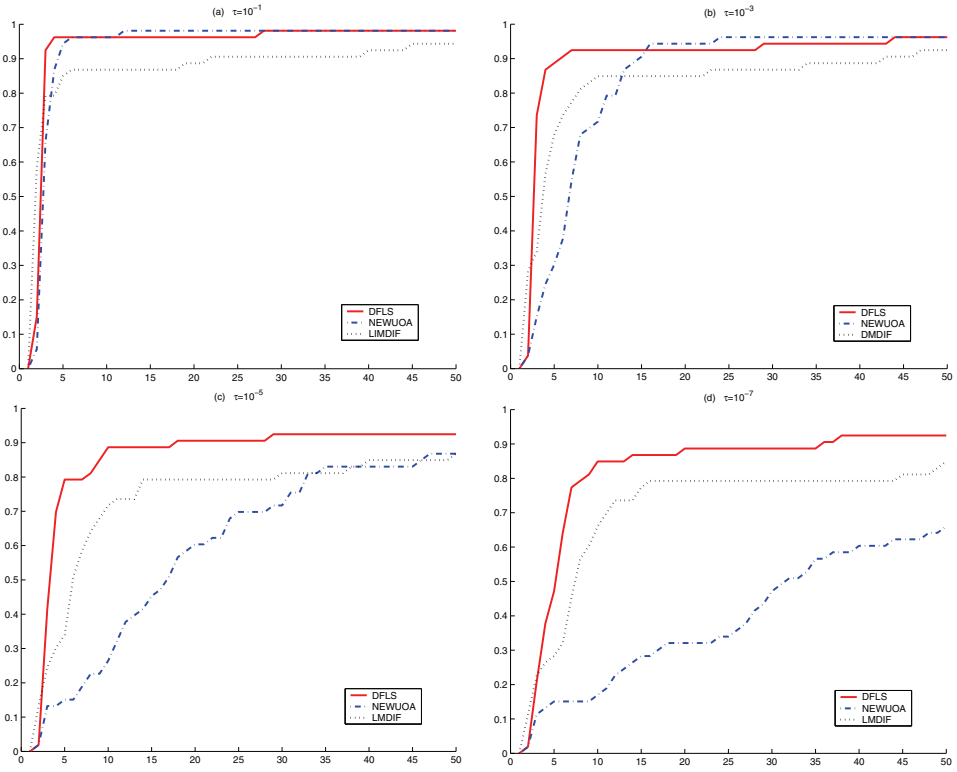

FIG. 5.1. *Data profiles of function $d_s(\alpha)$ for smooth problems:* (a) $\tau = 10^{-1}$, (b) $\tau = 10^{-3}$, (c) $\tau = 10^{-5}$, (d) $\tau = 10^{-7}$.

a computational budget of (simplex) gradients can be expressed as a function

$$d_s(\alpha) = \frac{|p \in \mathcal{P} : t_{p,s} \leq \alpha(n_p + 1)|}{|\mathcal{P}|},$$

where α is a positive integer and $|\cdot|$ means the cardinality of the set (see [20]). If the convergence test (5.1) cannot be satisfied after the prespecified computational budget, we set $t_{p,s} = \infty$.

The data profiles in Figure 5.1 show the performances of the different solvers for the original 53 problems in the test set $\mathcal{P}$ for various sizes of the computational budget up to 50 (simplex) gradients and different levels of accuracy, $\tau = 10^{-1}, 10^{-3}, 10^{-5}$, and 10^{-7}. In this case, the final trust region radius of NEWUOA and DFLS is set to 10^{-8}, and the finite difference parameter of LIMDIF is 10^{-7}. When the tolerance τ is relatively large, the performances of these solvers is more alike. In this case, although there are some differences when the computational budget is very small, say, within the budget of 25 (simplex) gradients, all solvers seem to solve more than 90% of the problems within 50 (simplex) gradients (see Figures 5.1(a) and (b)). However, when the tolerance becomes small, say, $\tau = 10^{-5}$, DFLS and LMDIF outperform NEWUOA within the budget of 25 (simplex) gradients, but NEWUOA is able to catch up with LMDIF by 50 (simplex) gradients (see Figure 5.1(c)). When the tolerance is even tighter, say, $\tau = 10^{-7}$, it seems DFLS ranks first, LMDIF ranks second, and NEWUOA ranks third (see Figure 5.1(d)). This result is perhaps not surprising.

Because NEWUOA assumes no knowledge of the problem structure, it probably is unable to build a sufficiently accurate model within a relatively small computational budget, while both DFLS and LMDIF take advantage of the structure of the least-squares problem and are thereby capable of constructing a more adequate model. In all cases, DFLS seems to perform the best of these three solvers. In addition, we can see from Figure 5.1 that DFLS seems to be able to obtain the function value to the level $(1-10^{-7})\Phi_L$ for about 88% of the smooth problems in $\mathcal{P}$ within only about 22 (simplex) gradients, where Φ_L is the smallest function value obtained by any solver within 50 (simplex) gradients.

As suggested in the paper [20], we also want to see the behavior of different solvers on the second class of problems proposed in [20], which mimic simulations defined by some iterative process, for example, solving differential equations with parameters. For convenience, we repeat the following definition of this class of problems, denoted by $\mathcal{P}_\mathcal{N}$, in terms of our notation. Let

$$\begin{aligned}\Phi_\mathcal{N}(\mathbf{x}) &= (1+\epsilon_\mathcal{N}\pi(\mathbf{x}))\Phi(\mathbf{x}) \\ &= \frac{1}{2}(1+\epsilon_\mathcal{N}\pi(\mathbf{x}))\sum_{i=1}^{m} f_i^2(\mathbf{x}),\end{aligned} \tag{5.2}$$

where $0 < \epsilon_\mathcal{N} < 1$ is the relative noise level, $\Phi(\mathbf{x})$ is a problem in $\mathcal{P}$, and the nonstochastic noise function $\pi : \mathbb{R}^n \to [-1,1]$ is defined as

$$\pi(\mathbf{x}) = T_3(\pi_0(\mathbf{x})), \quad \text{where} \quad T_3(\alpha) = \alpha(4\alpha^2 - 3)$$

is the cubic Chebyshev polynomial and

$$\pi_0(\mathbf{x}) = 0.9\sin(100\|\mathbf{x}\|_1)\cos(100\|\mathbf{x}\|_\infty) + 0.1\cos(\|\mathbf{x}\|_2).$$

One advantage of using a nonstochastic noise function is that the computational results are reproducible. In total, there are 53 problems in $\mathcal{P}_\mathcal{N}$. From the definition (5.2), the problem $\Phi_\mathcal{N}(\mathbf{x}) \in \mathcal{P}_\mathcal{N}$ can also be easily written in our least-squares format

$$\Phi_\mathcal{N}(\mathbf{x}) = \frac{1}{2}\sum_{i=1}^{m} \hat{f}_i^2(\mathbf{x}),$$

where $\hat{f}_i(\mathbf{x}) = \sqrt{1+\epsilon_\mathcal{N}\pi(\mathbf{x})}f_i(\mathbf{x})$ for $i = 1,\ldots,m$.

As in [20], we set the noise level, $\epsilon_\mathcal{N} = 10^{-3}$, in our numerical experiments. The final trust region radius of NEWUOA and DFLS and the finite difference parameter of LIMDIF are all set to $\max\{\sqrt{10^{-3}\Phi^*}, 10^{-7}\}$ to be comparable to the noise level, where Φ^* is the optimal function value of the problem without noise. The performances of different solvers with computational budget 50 (simplex) gradients for this second problem set, $\mathcal{P}_\mathcal{N}$, are shown in Figure 5.2. Unsurprisingly, for each level of accuracy, the performances of all of the solvers declines for this nonstochastic noisy problem set. However, DFLS always ranks first and is least affected by the noise among the three codes when compared with their performances for the problem set $\mathcal{P}$ without noise. For very low accuracy solutions, NEWUOA ranks second, and LMDIF ranks third (see Figure 5.2(a)). As we require more accuracy, LMDIF starts to surpass NEWUOA if we have a small computational budget (see Figures 5.2(b) and (c)) and finally completely outperforms NEWUOA (see Figure 5.2(d)). We can also see by comparing the individual performances of each code in Figures 5.1 and 5.2 that the polynomial

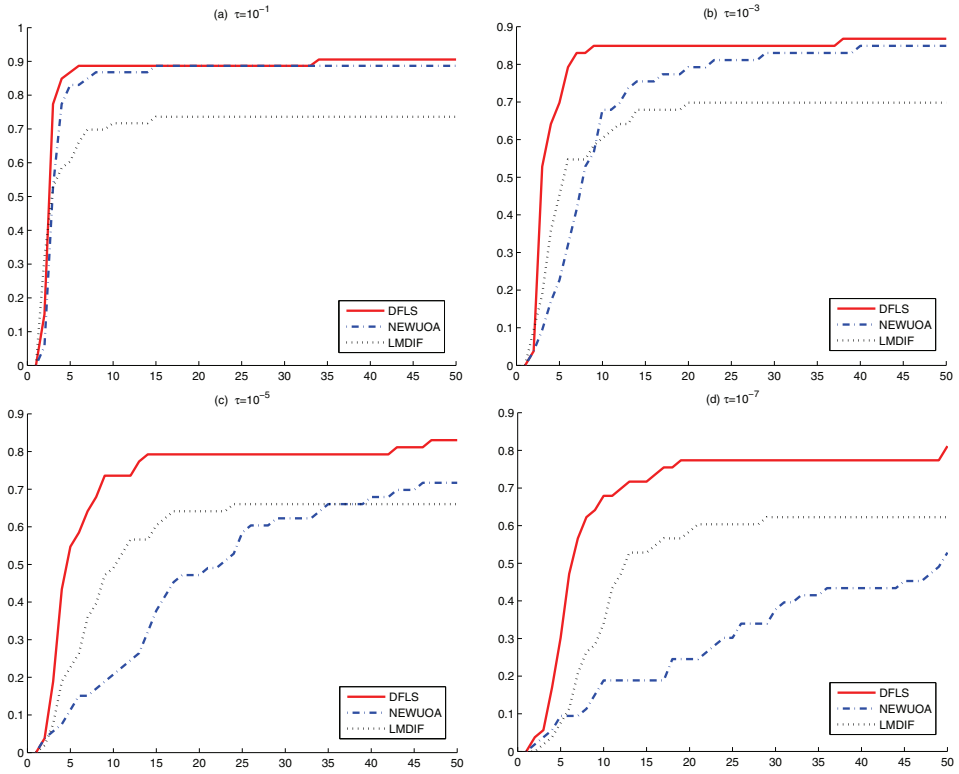

FIG. 5.2. *Data profiles of function* $d_s(\alpha)$ *for nonstochastic noisy problems:* (a) $\tau = 10^{-1}$, (b) $\tau = 10^{-3}$, (c) $\tau = 10^{-5}$, (d) $\tau = 10^{-7}$.

interpolation-based methods DFLS and NEWUOA are relatively less affected than the finite difference-based method LMDIF. This may be because DFLS and NEWUOA gradually build quadratic models based on the accumulated information from previous iterations, while the gradient is only approximated in LMDIF by taking the finite difference at the current iteration. So the noise in DFLS and NEWUOA could be, at least to some extent, filtered or canceled. However, again, since NEWUOA does not use the structure of the least-squares problem, its performance becomes worst compared with DFLS and LMDIF when we require more and more accurate solutions. But DFLS consistently performs very well because of exploiting the problem structure. In addition, it can be noticed that compared with Figure 5.1, the relative performances of DFLS compared with the other two codes in Figure 5.2 are even better for the noisy problems.

6. Conclusion. In this paper, we present a framework for a class of DFLS algorithms for minimizing least-squares problems. The global convergence of these methods has been established in a trust region framework. The local convergence of the methods will be reported in a separate paper [29]. Preliminary numerical experiments have been done to compare the performances of DFLS with the other two benchmark algorithms, NEWUOA and LMDIF. It is worth noting that numerical results can vary considerably, depending upon the presence or absence of noise and upon the approximate or more accurate solution one requires. This is the case even to the extent that the algorithm of choice might change. For example, in the present

paper there are circumstances where one is preferable to the other among LMDIF and NEWUOA, although overall it is clear that DFLS is preferable to both. We have also observed that the polynomial interpolation-based methods DFLS are more reliable (for both problems with noise and problems without noise) than the finite difference-based method LMDIF. However, for the least-squares minimization, without making use of the problem structure, the polynomial interpolation-based method NEWUOA may not build accurate models in a relatively limited computing budget and hence cannot ultimately perform as efficiently as LMDIF, which does make full use of the problem structure. Therefore, for a method to be both efficient and reliable for least-squares minimization, the polynomial interpolation-based methods which also exploit the problem structure are recommended. DFLS is not only based on polynomial interpolation models but also takes full advantage of the least-squares problem structure. Hence, it is reasonable to believe DFLS will perform more efficiently and will be more robust compared with methods that implement only one of the two features in the above discussion. Our numerical results support this conclusion. Finally, we would like to remark that DFLS can naturally be extended to handle the least-squares minimization with bound constraints, and this generalization of DFLS, called DFBOLS, has been developed recently and again shows excellent numerical performance.

REFERENCES

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

[2] G. A. Gray and T. G. Kolda, *Algorithm 856: APPSPACK 4.0: Asynchronous parallel pattern search for derivative-free optimization*, ACM Tran. Math. Software, 32 (2006), pp. 485–507.

[3] A. R. Conn, N. I. M. Gould, and Ph. L. Toint, *Trust-Region Methods*, MPS/SIAM Ser. Optim. 1, SIAM, Philadelphia, 2000.

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

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

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

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

[8] A. R. Conn, K. Scheinberg, and L. N. Vicente, *Geometry of sample sets in derivative free optimization: Polynomial regression and underdetermined interpolation*, IMA J. Numer. Anal., 28 (2008), pp. 721–748.

[9] H. Dan, N. Yamashita, and M. Fukushima, *Convergence properties of the inexact Levenberg–Marquardt method under local error bound*, Optim. Methods Softw., 17 (2002), pp. 605–626.

[10] J. E. Dennis and J. J. Moré, *A characterization of superlinear convergence and its application to quasi-Newton methods*, Math. Comp., 28 (1974), pp. 549–560.

[11] J. E. Dennis and J. J. Moré, *Quasi-Newton methods, motivation and theory*, SIAM Rev., 19 (1977), pp. 46–89.

[12] J. Fan and Y. Yuan, *On the quadratic convergence of the Levenberg–Marquardt method without nonsingularity assumption*, Computing, 74 (2005), pp. 23–39.

[13] N. I. M. Gould, D. Orban, and Ph. L. Toint, *CUTEr and SifDec: A constrained and unconstrained testing environment, revisited*, ACM Trans. Math. Software, 29 (2003), pp. 373–394.

[14] W. W. Hager and H. Zhang, *Self-adaptive inexact proximal point methods*, Comput. Optim. Appl., 39 (2008), pp. 161–181.

[15] W. W. Hager and H. Zhang, *Asymptotic convergence analysis of a new class of proximal point methods*, SIAM J. Control and Optim., 46 (2007), pp. 1683–1704.
[16] W. Hock and K. Schittkowski, *Test Examples for Nonlinear Programming Codes*, Lecture Notes in Econom. and Math. Systems 187, Springer, Berlin, 1981.
[17] M. Marazzi and J. Nocedal, *Wedge trust region methods for derivative-free optimization*, Math. Program., 91 (2002), pp. 289–305.
[18] J. J. Moré, *Optimization: Algorithms, Software and Environments*, http://www.mcs.anl.gov/∼more.
[19] J. J. Moré, *The Levenberg–Marquardt algorithm, implementation and theory*, Numerical Analysis, G. A. Watson, ed., Lecture Notes in Math. 630, Springer, New York, 1977.
[20] J. J. Moré and S. M. Wild, *Benchmarking derivative-free optimization algorithms*, SIAM J. Optim., 20 (2009), pp. 172–191.
[21] M. J. D. Powell, *Developments of NEWUOA for unconstrained minimization without derivatives*, IMA J. Numer. Anal., 28 (2008), pp. 649–664.
[22] M. J. D. Powell, *The NEWUOA software for unconstrained optimization without derivatives*, in Large-Scale Nonlinear Optimization, Nonconvex Optim. Appl. 83, Springer, New York, 2006.
[23] M. J. D. Powell, *Least Frobenius norm updating of quadratic models that satisfy interpolation conditions*, Math. Program. Ser. B, 100 (2004), pp. 183–215.
[24] M. J. D. Powell, *On trust region methods for unconstrained minimization without derivatives*, Math. Program., 97 (2003), pp. 605–623.
[25] M. J. D. Powell, *A new algorithm for unconstrained optimization*, in Nonlinear Programming, J. B. Rosen, O. L. Mangasarian, and K. Ritter, eds., Academic Press, New York, 1971, pp. 31–36.
[26] P. Tseng, *Error bounds and superlinear convergence analysis of some Newton-type methods in optimization*, in Nonlinear Optimization and Related Topics, G. Di Pillo and F. Giannessi, eds., Kluwer, Academic Publishers, Norwell, MA, 2000, pp. 445–462.
[27] N. Yamashita and M. Fukushima, *The proximal point algorithm with genuine superlinear convergence for the monotone complementarity problem*, SIAM J. Optim., 11 (2000), pp. 364–379.
[28] N. Yamashita and M. Fukushima, *On the rate of convergence of the Levenberg–Marquardt method*, Comput., 15 (2001), pp. 237–249.
[29] H. Zhang and A. R. Conn, *On the local convergence of a derivative-free algorithm for least-squares minimization*, Comput. Optim. Appl., accepted, 2010.