Optimization by moving ridge functions: derivative-free optimization for computationally intensive functions

James C. Gross and Geoffrey T. Parks

Department of Engineering, University of Cambridge, Cambridge, UK

ABSTRACT
A novel derivative-free algorithm, called optimization by moving ridge functions (OMoRF), for unconstrained and bound-constrained optimization is presented. This algorithm couples trust region methodologies with output-based dimension reduction to accelerate convergence of model-based optimization strategies. The dimension-reducing subspace is updated as the trust region moves through the function domain, allowing OMoRF to be applied to functions with no known global low-dimensional structure. Furthermore, its low computational requirement allows it to make rapid progress when optimizing high-dimensional functions. Its performance is examined on a set of test problems of moderate to high dimension and a high-dimensional design optimization problem. The results show that OMoRF compares favourably with other common derivative-free optimization methods, even for functions in which no underlying global low-dimensional structure is known.

ARTICLE HISTORY
Received 8 July 2020
Accepted 1 February 2021

KEYWORDS
Derivative-free optimization; nonlinear optimization; trust region methods; dimension reduction; ridge functions

1. Introduction
Derivative-free optimization (DFO) methods seek to solve optimization problems using only function evaluations—that is, without the use of derivative information. These methods are particularly suited for cases where the objective function is a ‘black box’ or computationally intensive (Conn, Scheinberg, and Vicente 2009). In these cases, computing gradients analytically or through algorithmic differentiation may be infeasible and approximating gradients using finite differences may be intractable. Common applications of DFO methods include engineering design optimization (Kipouros et al. 2008), hyper-parameter optimization in machine learning (Ghanbari and Scheinberg 2017) and more (Levina et al. 2009). Derivative-free trust region (DFTR) methods are an important class of DFO that iteratively create and optimize a local surrogate model of the objective in a small region of the function domain, called the trust region. Unlike standard trust region methods, DFTR methods use interpolation or regression to construct a surrogate model, thereby avoiding the use of derivative information.

However, acquiring enough samples for surrogate model construction may be computationally prohibitive for problems of moderate to high dimension. This issue is magnified when considering computationally intensive functions, such as computational fluid dynamics (CFD) or finite element method (FEM) simulations, where a single function evaluation may require minutes, hours or even days (Gu 2001). For these functions, the cost of optimization is dominated by the cost of function
evaluation rather than by the optimization algorithm itself. Algorithms that can achieve an acceptable level of convergence in relatively few function evaluations are highly desirable in such cases.

Fortunately, it has been shown that many functions of interest vary primarily along low-dimensional linear subspaces of the inputs. For example, the efficiency and pressure ratio of turbo-machinery models (Seshadri et al. 2018), merit functions of hyper-parameters of neural networks (Bergstra and Bengio 2012), and drag and lift coefficients of aerospace vehicles (Łukaczyk et al. 2014) have all been shown to have low-dimensional structure. Functions that have this structure are known as ridge functions (Pinkus 2015), and may be written

\[ f(x) \approx m(U^T x), \]

where \( f : \mathcal{D} \subseteq \mathbb{R}^n \rightarrow \mathbb{R} \), \( m : \text{proj}_U(D) \subseteq \mathbb{R}^d \rightarrow \mathbb{R} \), \( U \in \mathbb{R}^{n \times d} \), \( \text{proj}_U(D) \) denotes the \( d \)-dimensional projection of domain \( D \) onto the subspace \( U \), and \( d < n \). If \( d \ll n \), then exploiting low-dimensional structure may lead to significant reductions in computational requirement.

In this article, a novel DFTR method for unconstrained and bound-constrained nonlinear optimization of computationally intensive functions is presented. This algorithm is called optimization by moving ridge functions (OMoRF), as it leverages local ridge function approximations that move through the function domain. Although other optimization algorithms have used subspaces to reduce the problem dimension (Wang et al. 2016; Zhao, Alimo, and Bewley 2018; Kozak et al. 2019), OMoRF differs from these in some key aspects. First, it is completely derivative-free. This differs from the variance reduced stochastic subspace descent (VRSSD) algorithm presented by Kozak et al. (2019). Although the VRSSD algorithm does not require full gradient calculations, it still requires directional derivatives to be computed. Similarly, the Delaunay-based derivative-free optimization via global surrogates with active subspace method (Δ-DOGS with ASM) proposed by Zhao, Alimo, and Bewley (2018) requires an initial sample of gradient evaluations to determine the dimension-reducing subspace. Second, the subspaces computed by OMoRF correspond to the directions of strongest variability of the objective function. In contrast, the random embeddings approach used by Wang et al. (2016) and Cartis and Otemissov (2020) randomly generates a dimension-reducing subspace which, in general, does not correspond to directions of high variability of the function. Finally, OMoRF does not assume a global dimension-reducing subspace. Many similar algorithms that use ridge functions for optimization purposes (Łukaczyk et al. 2014; Zhao, Alimo, and Bewley 2018; Gross, Seshadri, and Parks 2020) assume that the function \( f \) varies primarily along a constant linear subspace \( U \) throughout the entire domain \( D \). This assumption may limit the application of ridge functions to a few special cases. Although adaptive sampling approaches have been applied previously for stochastic optimization with active subspaces (Choromanski et al. 2019), to the best of the authors’ knowledge, OMoRF is the first model-based optimization algorithm to propose the use of local ridge functions.

This article has three main contributions. First, a novel strategy for dynamically updating the subspace for ridge functions during model-based optimization is proposed. Moreover, using theoretical results from interpolation and ridge approximation theory, the benefits of this approach are demonstrated. Second, the OMoRF algorithm is presented, and its performance is tested on a variety of test problems, including a high-dimensional aerodynamic design optimization problem. Additionally, an open source Python implementation of this algorithm has been made available for public use. Third, a novel sampling method for ridge function models that maintains two separate interpolation sets, one for ensuring accurate local subspaces and one for ensuring accurate quadratic models over those subspaces, is presented.

The rest of this article is organized as follows. A brief introduction to trust region methods is provided in Section 2. In Section 3, algorithms for constructing ridge function approximations are explored. This section also includes a discussion on the suitability of ridge function models in DFTR methods, with this discussion motivating the use of moving ridge function models. The OMoRF algorithm and some key features of the algorithm are presented in Section 4. In Section 5, the
algorithm is tested against other common DFO methods on a variety of test problems. Finally, a few concluding remarks are provided in Section 6.

2. Trust region methods

Trust region methods replace the unconstrained optimization problem

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

with a sequence of trust region subproblems

\[ \min_{s} m_k(x_k + s) \]

subject to \( \|s\| \leq \Delta_k \),

where \( x_k \) is the current iterate, \( \Delta_k \) is the trust region radius, and \( m_k \) is a simple model that approximates \( f \) in the trust region

\[ B(x_k, \Delta_k) := \{ x \in \mathbb{R}^n | \|x - x_k\| \leq \Delta_k \}. \]

The solution to the trust region subproblem (3) gives a step \( s_k \), with \( x_k + s_k \) a candidate for the next iterate \( x_{k+1} \). The ratio

\[ r_k = \frac{\text{actual reduction}}{\text{predicted reduction}} := \frac{f(x_k) - f(x_k + s_k)}{m_k(x_k) - m_k(x_k + s_k)} \]

is used to determine if the candidate is accepted or rejected and the trust region radius increased or reduced.

2.1. Derivative-free trust region methods

For standard trust region methods, a common choice of model is the Taylor series expansion centred around \( x_k \)

\[ m_k(x_k + s) = f(x_k) + \nabla f(x_k)^T s + \frac{1}{2} s^T B_k s, \]

where \( B_k \) is a symmetric matrix approximating the Hessian \( \nabla^2 f(x_k) \). Constructing this Taylor quadratic clearly requires knowledge of the function derivatives. Alternatively, DFTR methods may use interpolation or regression to construct \( m_k \). That is, using a set of \( p \) samples \( X = \{ x^1, x^2, \ldots, x^p \} \) and \( q \) basis functions \( \phi(x) = \{ \phi_1(x), \ldots, \phi_q(x) \} \), the model is defined as

\[ m_k(x) = \sum_{j=1}^{q} \alpha_j \phi_j(x), \]

where \( \alpha_j \) for \( j = 1, \ldots, q \) are the coefficients of the model. In the case of fully-determined interpolation, the number of samples is equal to the number of coefficients, i.e. \( p = q \), so the coefficients may be found by solving the linear system

\[ M(\phi, X) \alpha = f, \]
where

\[
\mathbf{M}(\phi, X) = \begin{bmatrix}
\phi_1(x^1) & \phi_2(x^1) & \cdots & \phi_p(x^1) \\
\phi_1(x^2) & \phi_2(x^2) & \cdots & \phi_p(x^2) \\
\vdots & \vdots & \ddots & \vdots \\
\phi_1(x^p) & \phi_2(x^p) & \cdots & \phi_p(x^p)
\end{bmatrix}, \quad \alpha = \begin{bmatrix}
\alpha_1 \\
\alpha_2 \\
\vdots \\
\alpha_p
\end{bmatrix}, \quad \text{and} \quad f = \begin{bmatrix}
f(x^1) \\
f(x^2) \\
\vdots \\
f(x^p)
\end{bmatrix}.
\] (9)

Global convergence of trust region methods which use the Taylor quadratic (6) has been proven (Conn, Gould, and Toint 2000). These convergence properties rely heavily on the well-understood global convergence of trust region methods which use the Taylor-like error bounds for Taylor series. For these same guarantees to hold for DFTR methods, one must ensure the surrogate models satisfy Taylor-like error bounds

\[
|f(x) - m_k(x)| \leq \kappa_1 \Delta_k^2
\]

\[
\|\nabla f(x) - \nabla m_k(x)\| \leq \kappa_2 \Delta_k
\] (10)

for all \(x \in B(x_k, \Delta_k)\), where \(\kappa_1, \kappa_2 > 0\) are independent of \(x_k\) and \(\Delta_k\). Models which satisfy these conditions are known as fully linear. It has been shown that the full linearity of a model may be ensured by satisfying certain geometric conditions on the sample set \(X\) (Conn, Scheinberg, and Vicente 2008). Moreover, numerous geometry-improving algorithms for ensuring fully linear polynomial models are presented in Conn, Scheinberg, and Vicente (2009, Chapter 6).

### 2.2. Limitations of derivative-free trust region methods

Fully-determined polynomial interpolation of degree \(r\) in \(n\) dimensions requires \(p = \binom{n + r}{r}\) function evaluations. In the case of quadratic interpolation, this would require \(p = \frac{1}{2}(n + 1)(n + 2)\) sample points. When \(n\) is small, this requirement may be met easily; however, as \(n\) increases, this requirement may become prohibitive, particularly in the case of functions that are expensive to evaluate. Many DFTR methods account for this computational burden by avoiding the use of fully-determined quadratic models. For instance, the optimization algorithm COBYLA (Powell 1994) uses linear models, requiring only \(n + 1\) samples. Although this greatly reduces the effect of increased dimensionality, linear models generally do not capture the curvature of the true function, so convergence may be slow (Wendor, Botero, and Alonso 2016).

Some algorithms reduce the required number of samples by constructing under-determined quadratic interpolation models. This requires more samples than are necessary for linear models, but fewer than fully-determined quadratic models. For example, the algorithms NEWUOA (Powell 2006) and BOBYQA (Powell 2009) use minimum Frobenius norm interpolating quadratics, which require a constant number of more than \(n + 1\) points, but less than \(\frac{1}{2}(n + 1)(n + 2)\) points. Note that, typically, a default of \(2n + 1\) points is used. Although this approach has generally proven to be quite effective in practice, this initial requirement may limit the efficacy of these algorithms when the objective is both high-dimensional and computationally intensive.

Alternatively, other DFTR algorithms seek to reduce the initial start-up cost by using very few points at first, but increasing the number of points as more become available. For example, the DFO-TR algorithm proposed by Bandeira, Scheinberg, and Vicente (2012) builds quadratic models using significantly fewer than \(\frac{1}{2}(n + 1)(n + 2)\) points, possibly as few as \(n + 1\) points. Although this was done using minimum Frobenius norm models, as in NEWUOA and BOBYQA, they also showed that, by assuming approximate Hessian sparsity, one could also use sparsity recovery techniques, such as compressed sensing (Eldar and Kutyniok 2012). Furthermore, it was proven that such models are probabilistically fully quadratic. That is, these models satisfy second-order Taylor-like error bounds with a probability bounded below by a term dependent on the number of sample points used. Moreover, the convergence of DFTR methods that employ probabilistically fully linear/quadratic models was proved by Bandeira, Scheinberg, and Vicente (2014), provided the models satisfied the Taylor-like error bounds with probability greater than or equal to 1/2.
3. Ridge function approximations

Ridge function approximations allow one to reduce the effective dimensionality of a function by determining a low-dimensional representation that is a function of a few linear combinations of the high-dimensional input. These approximations can be determined using a number of methods (Constantine 2015; Diez, Campana, and Stern 2015; Hokanson and Constantine 2017). This article will focus on two approaches: (1) derivative-free active subspaces; and (2) polynomial ridge approximation.

3.1. Derivative-free active subspaces

The active subspace of a given function $f(x)$ has been defined by Constantine, Dow, and Wang (2014) as the $d$-dimensional subspace $U \in \mathbb{R}^{n \times d}$ of the inputs $x \in \mathbb{R}^n$ that corresponds to the directions of strongest variability of $f$. To see how one may discover $U$, consider a probability density function $\rho(x)$ that is strictly positive on the domain of interest and assume that

$$\int x \rho(x) \, dx = 0 \quad \text{and} \quad \int x x^T \rho(x) \, dx = I,$$

where $I$ is the $n \times n$ identity matrix. Provided that $\int x x^T \rho(x) \, dx$ is full rank, these assumptions are easily satisfied by a change of variables (Constantine and Doostan 2017). Typically, $\rho(x)$ is taken to be Gaussian for unbounded inputs $x$, with each coordinate scaled and shifted to be of mean zero and standard deviation one. When $x$ is bounded below and above, $\rho(x)$ is generally taken to be the uniform distribution with $x$ scaled and shifted to lie within $[-1, 1]^n$.

Given $f$ and its partial derivatives are square integrable with respect to $\rho(x)$, the active subspace of $f$ can be found using the covariance matrix

$$C = \int (\nabla f(x))(\nabla f(x))^T \rho(x) \, dx.$$  

(12)

In practice, this covariance matrix is approximated by

$$C \approx \frac{1}{M} \sum_{i=1}^{M} (\nabla f(x_i))(\nabla f(x_i))^T,$$

(13)

where $x_i$ are drawn randomly from $\rho(x)$. This matrix is symmetric, positive semidefinite, so its real eigendecomposition is given by

$$C = W \Lambda W^T,$$

(14)

where $\Lambda = \text{diag}(\lambda_1, \ldots, \lambda_d, \ldots, \lambda_n)$ and $\lambda_1 \geq \cdots \geq \lambda_d \geq \cdots \geq \lambda_n \geq 0$. Partitioning $W$ and $\Lambda$ as

$$W = [U \quad V], \quad \Lambda = \begin{bmatrix} \Lambda_1 & 0 \\ 0 & \Lambda_2 \end{bmatrix}$$

(15)

results in the active subspace $U \in \mathbb{R}^{n \times d}$ and the inactive subspace $V \in \mathbb{R}^{n \times (n-d)}$. The reduced coordinates $y = U^T x$ and $z = V^T x$ are known as the active and inactive variables, respectively. The following lemma quantifies the variation of $f$ along these coordinates.

Lemma 3.1: The mean-squared gradients of $f$ with respect to the coordinates $y$ and $z$ satisfy

$$\mathbb{E} \left[ (\nabla_y f)^T (\nabla_y f) \right] = \lambda_1 + \cdots + \lambda_d,$$

$$\mathbb{E} \left[ (\nabla_z f)^T (\nabla_z f) \right] = \lambda_{d+1} + \cdots + \lambda_n.$$  

(16)
**Proof:** See the proof of Constantine (2015, Lemma 2.2).

From Lemma 3.1 it is clear that on average $f$ shows greater variability along $y$ than $z$. Moreover, the sum of the partitioned eigenvalues $\Lambda_1$ and $\Lambda_2$ quantifies this variation. This motivates the well-known heuristic of choosing the reduced dimension $d$ as the index with the greatest log decay of eigenvalues (Constantine 2015).

In the derivative-free context, the approximate covariance matrix

$$\hat{C} = \int (\nabla \hat{f}(x))(\nabla \hat{f}(x))^T \rho(x) \, dx,$$

(17)

where $\hat{f}$ is a surrogate model of $f$, may be used as a surrogate for $C$. The efficacy of this approach is clearly dependent on the accuracy of the inferred gradients. To provide a theoretical guarantee of this statement, assume that

$$\| \nabla \hat{f}(x) - \nabla f(x) \| \leq \omega_h$$

(18)

for all $x \in B$ for some domain $B$ with $\omega_h$ independent of $x$ and

$$\lim_{h \to 0} \omega_h = 0,$$

where $\hat{f}$ is a surrogate model for $f$ and $h$ is some controllable parameter. Note that, if $\hat{f}$ is fully linear, then by definition

$$\| \nabla f(x) - \nabla \hat{f}(x) \| \leq \kappa_2 \Delta_k,$$

implying that fully linear models inherently satisfy this assumption as $\Delta_k \to 0$, i.e. as the trust region radius shrinks. Given this assumption, the following lemma—modified from Constantine (2015, Lemma 3.11)—provides an error bound between the approximate covariance matrix $\hat{C}$ (17) and the true covariance matrix $C$ (12).

**Lemma 3.2:** Assume $\nabla f(x)$ is Lipschitz continuous with Lipschitz constant $\gamma_f$. The norm of the difference between $C$ and $\hat{C}$ is bounded by

$$\| C - \hat{C} \| \leq (\omega_h + 2\gamma_f) \omega_h.$$  

(19)

**Proof:** See Section 1 of the online supplemental data, which can be accessed at https://doi.org/10.1080/0305215X.2021.1886286.

Provided $\nabla \hat{f}(x)$ is easily computed, one may be able to compute an analytic form for $\hat{C}$. Two model-based heuristics for approximating active subspaces using $\hat{C}$, one with $\hat{f}$ a quadratic model and one a linear model, were proposed by Constantine and Doostan (2017, Algorithms 1 and 2). In the case of a linear model

$$\hat{f}(x) = c + b^T x,$$

(20)

the approximate covariance matrix (17) becomes

$$\hat{C} = bb^T,$$

(21)

so the active subspace may be approximated by the one-dimensional vector

$$U \approx b / \| b \|.$$  

(22)

Note that, when using a linear model, this method may only be used for approximating a one-dimensional active subspace. In the case where a higher dimension is desired, an alternative approach is required.
3.2. Polynomial ridge approximation

Unlike the active subspaces approach, ridge function recovery allows one to find the subspace $U$ and the coefficients $\alpha$ (7) of a ridge function $m(U^T x)$ simultaneously. Hokanson and Constantine (2017) developed a method of doing this for the case in which $m$ is a polynomial of dimension $d$ and degree $r$. Their approach was to use variable projection to solve the minimization problem

$$
\min_{m \in P^r(\mathbb{R}^d), \ U \in \mathcal{G}(d, \mathbb{R}^n)} \sum_{i=1}^{M} \left[ f(x^i) - m(U^T x^i) \right]^2,
$$

where $P^r(\mathbb{R}^d)$ denotes the set of polynomials on $\mathbb{R}^d$ of degree $r$, $\mathcal{G}(d, \mathbb{R}^n)$ denotes the Grassmann manifold of $d$-dimensional subspaces of $\mathbb{R}^n$, and $\{x^i\}$ is a set of $M$ samples. Writing $m(U^T x^i) = M(\phi, \mathcal{Y}) \alpha$ (as seen in Equation (8)), where $\mathcal{Y} = \{U^T x^i \mid i = 1, \ldots, M\}$, allows one to formulate (23) as a nonlinear least squares problem in terms of the coefficients $\alpha$ and the subspace $U$

$$
\min_{\alpha \in \mathbb{R}^q, \ U \in \mathcal{G}(d, \mathbb{R}^n)} \| f - M(\phi, \mathcal{Y}) \alpha \|_2^2,
$$

with $f \in \mathbb{R}^M$ such that $f_i = f(x^i)$ and $q = \binom{d+r}{r}$. Using the fact that $\alpha$ may be easily discovered using the Moore–Penrose pseudoinverse, one may write (24) as the Grassmann manifold optimization problem

$$
\min_{\ U \in \mathcal{G}(d, \mathbb{R}^n)} \| f - M(\phi, \mathcal{Y}) M(\phi, \mathcal{Y})^\dagger f \|_2^2,
$$

over strictly $U$. To solve this problem, Hokanson and Constantine (2017) developed a novel Grassmann–Gauss–Newton method for solving (25) iteratively.

3.3. Fully linear ridge function models

Global convergence of DFTR methods relies on models that are fully linear, i.e. models that satisfy the bounds (10). Demonstrating full linearity of ridge function models $m(U^T x)$ requires

$$
|f(x) - m(U^T x)| \leq \kappa_1 \Delta^2
$$

$$
\|\nabla f(x) - \nabla m(U^T x)\| \leq \kappa_2 \Delta
$$

for all $x \in B(x_\ell, \Delta)$ (4), where $\kappa_1, \kappa_2$ are constants that are independent of $x$ and $\Delta$. For standard polynomial interpolation models, one may use well poisedness of the interpolation set to prove full linearity—as shown by Conn, Scheinberg, and Vicente (2008). However, in the case of polynomial ridge functions, there is an extra level of complexity involved as the models are constructed over a projected input space. Unless $f$ has an exact ridge function representation with known effective dimension, projection of its domain onto the subspace $U$ will have some inherent information loss associated with it. This is because, for each value of the reduced coordinate $y$, there exist many (possibly infinitely many) coordinates in the full space that map to it. Variations in the function values associated with each full space coordinate may show up as ‘noise’ in the $d$-dimensional projection of the function domain. This observation leads to two forms of error: (1) information loss from dimension reduction; and (2) response surface error that arises from polynomial interpolation with samples that are corrupted by noise.

To formalize these two forms of error, consider the conditional expectation of $f$ given $y = U^T x$. 
Theorem 8.3).  

Proof: See the proof of Constantine (2015, Proposition 2.3).

Definition 3.3: Let $f(x)$ be square-integrable with respect to a probability density function $\rho(x)$, $U \in \mathbb{R}^{n \times d}$ be a subspace with orthogonal columns and $V \in \mathbb{R}^{n \times (n-d)}$ be an orthogonal basis for the complement of the span of the columns of $U$. The conditional expectation of $f$ given $y = U^T x$ is defined as

$$g(y) = \mathbb{E}[f | y] = \int_{z} f(Uy + Vz) \pi(z | y) \, dz,$$

where $z = V^T x$ and $\pi(z | y)$ is the conditional density

$$\pi(z | y) = \frac{\rho(Uy + Vz)}{\int_{z} \rho(Uy + Vz) \, dz}.$$

That is, the conditional expectation of $f$ given $y$ is the average value of $f(x)$ for all possible values $x$ for a given reduced coordinate $y = U^T x$. The function $g(y)$ is the unique, optimal ridge function approximation in the $L^2(\rho)$ norm for a given subspace $U$—see the proof of Pinkus (2015, Theorem 8.3).

The error functions

$$e^f(x) = f(x) - g(U^T x), \quad e^g(x) = \nabla f(x) - \nabla g(U^T x)$$

represent the information loss from reducing onto the subspace $U$. If one can appropriately bound $|e^f(x)|$ and $\|e^g(x)\|$ for all $x \in B(\mathbf{x}_k, \Delta)$, one can show full linearity of the conditional expectation $g$ (27). Although the mean-squared error of $g$ with respect to $f$ is known to be bounded in expectation—see Constantine, Dow, and Wang (2014, Theorem 3.1)—there exists no formal error analysis for bounding the error functions (28) for any $f$ and $U$. However, in the special case where $f$ has no dependence on $z$, these error bounds are satisfied. Such functions are known as $z$-invariant. Using the following proposition proposed by Constantine (2015), full linearity of $g$ is trivially shown in the case where $f$ is $z$-invariant.

Proposition 3.4: Let $f$ be $z$-invariant. Then, for any two points $\mathbf{x}^1, \mathbf{x}^2$ that lie in the domain of $f$ and satisfy $y = U^T \mathbf{x}^1 = U^T \mathbf{x}^2$,

$$f(x^1) = f(x^2) \quad \text{and} \quad \nabla f(x^1) = \nabla f(x^2).$$

Proof: See the proof of Constantine (2015, Proposition 2.3).

Unfortunately, using $g(y)$ as a surrogate model is not practical, as it requires high-dimensional integration along the $z$-coordinate. Instead, a ridge function approximation $m(y)$ that acts as a surrogate to $g$ is used. This introduces the error functions

$$e^m(y) = g(y) - m(y), \quad e^m(y) = \nabla g(y) - \nabla m(y),$$

which represent the response surface error. Although $g$ may be smooth and differentiable with respect to reduced $y$-coordinates, the $d$-dimensional samples used to construct $m$ will be noisy as they will be obtained from the $n$-dimensional function $f$. Kannan and Wild (2012, Theorem 2.2) provide theoretical guarantees for quadratic models constructed from noisy functions. Using similar logic, the following theorem provides error bounds for $|e^m(y)|$ and $\|e^m(y)\|$.

Theorem 3.5: Suppose $g$ is continuously differentiable, $\nabla g$ is Lipschitz continuous with Lipschitz constant $\gamma_g$ in the trust region $B$ (where $B$ denotes $B(\mathbf{x}_k, \Delta)$, and that $X = \{\mathbf{x}_k, \ldots, \mathbf{x}^d\} \subset B$ contains at
least $d + 1$ points (including the current iterate $x_k$) which when projected onto the subspace $U$ results in 

$$Y = \{ y^i = U^T x^i \mid i = 2, \ldots, d + 1 \} \bigcup y_k$$

(where $y_k = U^T x_k$) of affinely independent points such that the matrix

$$Y = \frac{1}{\Delta} \begin{bmatrix} y^2 - y_k & \cdots & y^{d+1} - y_k \end{bmatrix}$$

is invertible. Then, if the quadratic ridge function

$$m(y) = c + g^T y + \frac{1}{2} y^T H y$$

interpolates $f$ at all points in $X$ such that, for any $x^i \in X$,

$$m(U^T x^i) = f(x^i),$$

the following inequalities hold for any $y = U^T x$ with $x \in B$:

$$|e^m(y)| \leq \kappa_3 \Delta^2 + \left( 2\sqrt{d} \| Y^{-1} \| \| U^T \| + 1 \right) \max_{x \in B} |e^g(x)|$$

$$\| e^m(x) \| \leq \kappa_4 \Delta + \frac{2\sqrt{d} \| Y^{-1} \| \| U^T \| \| U^T \| + 1}{\Delta} \max_{x \in B} |e^g(x)|$$

with

$$\kappa_3 = \| U^T \|^2 (\gamma_g + \| H \|_F) \frac{5\sqrt{d} \| Y^{-1} \| \| U^T \| + 1}{2}$$

$$\kappa_4 = (\gamma_g + \| H \|_F) \frac{5\sqrt{d} \| Y^{-1} \| \| U^T \| ^2}{2}.$$ (31)

**Proof:** See Section 2 of the online supplemental data. ■

### 3.4. Motivating moving ridge functions

The majority of research into ridge function approximations has assumed that the function of interest varies along a global subspace $U$ (Glaws et al. 2017; Wong et al. 2019; Gross, Seshadri, and Parks 2020). Unless the function is an exact ridge function, i.e. $f$ is $z$-invariant, this assumption may lead to a significant amount of information loss when projecting onto the subspace. However, using local subspaces $\{U_k\}$, with each $U_k$ corresponding to a small region of interest in the function domain, may allow $f$ to be modelled accurately as a $z$-invariant function.

To motivate this approach, six ten-dimensional functions from the CUTEst (Gould, Orban, and Toint 2015) problem set have been considered (ARGLINA, MCCORMCK, NCVXBQP1, PENALTY1, SCHMVETT, VARDIM). For each of these functions, active subspaces have been approximated using the Monte Carlo gradient sampling method (13) with 100,000 samples taken at uniformly distributed random locations. The active subspaces for six regions of interest defined by hypercubes of variable radius $\Delta$, all centred at the same randomly chosen location in the function domain, were calculated. The eigenvalues for each of these functions are shown in Figure 1.

From inspection of Figure 1, it is apparent that, as $\Delta$ decreases, the gap between the first eigenvalue and the remaining eigenvalues generally increases for these functions. This suggests that, as the region of interest becomes smaller, these functions become inherently one-dimensional. Moreover, for many of these functions, the remaining eigenvalues seem to tend to zero as $\Delta$ decreases. By Lemma 3.1, this
implies very little to no average variability along the inactive variables $z$, meaning these functions can be treated as nearly $z$-invariant for small $\Delta$. Note the observation that, in small regions of interest, the fact that multivariate functions can be approximated by one-dimensional ridge functions is not surprising. In particular, the first-order Taylor expansion

$$m(x) = f(a) + \nabla f(a)^T(x-a)$$

(32)

can be considered a one-dimensional ridge function with the subspace $U = \nabla f(a)$. However, this observation clearly breaks down when considering subspaces of higher dimension. Nevertheless, using subspaces of higher dimension may still be advantageous for some problems, as will be seen later.

In order to use ridge functions in a DFTR algorithm, it is hypothesized that the subspaces should be periodically updated as one moves through the function domain. To motivate this hypothesis, local active subspaces for six regions have been defined by hypercubes of radius $\Delta = 1.0 \times 10^{-3}$ for each of these six functions. The centroids of the hypercubes were chosen using Latin hypercube sampling such that each local region was sufficiently distant from the others. It was again found that each of these local regions had very strong one-dimensional structure, suggesting that the function behaviour in each of these regions could be accurately described using a one-dimensional subspace. The weights from each of these one-dimensional subspaces are shown in Figure 2, with the size of the markers indicating the relative size of the weights.

Inspecting this figure, it is clear that the weight vectors generally vary significantly between each region of interest. Clearly, it would be impossible accurately to describe these functions with global subspaces, as there are regions that have a weight vector that is linearly independent of the weight vectors associated with other regions of the function domain. Interestingly, some functions seem to have multiple regions of the function domain that can be defined using a single subspace. For instance, it appears possible to define the SCHMVETT function’s six regions of interest by three low-dimensional subspaces, one using strictly the 9th parameter, another using the 10th parameter,
Figure 2. Weights for one-dimensional active subspaces for functions: (a) ARGLINA; (b) MCCORMCK; (c) NCVXBQP1; (d) PENALTY1; (e) SCHMVETT; and (f) VARDIM for domains of radius $\Delta_1 = 1 \times 10^{-3}$ at variable locations.

and one with a mixture of the two. This observation may motivate the use of subspace clustering techniques (Parsons, Haque, and Liu 2004) in further studies.

4. OMoRF algorithm

The OMoRF algorithm is detailed in Algorithm 1. At each iteration, a local subspace $U_k$ is determined and a quadratic ridge function $m_k(U_k^T x)$ is constructed. To ensure the accuracy of this model, two separate interpolation sets are maintained: the set $X_{\text{sub}}^k$ is used to construct the local subspace $U_k$, while $X_{\text{int}}^k$ is used to determine the coefficients of the interpolation model $m_k$. Next, the trust region subproblem

$$\min_s m_k(U_k^T (x_k + s))$$

subject to $\|s\| \leq \Delta_k$

is solved to obtain a candidate solution $x_k + s_k$. The ratio

$$r_k = \frac{f(x_k) - f(x_k + s_k)}{m_k(U_k^T x_k) - m_k(U_k^T (x_k + s_k))}$$

is used to determine whether or not this candidate solution is accepted and if the trust region radius is decreased. Before decreasing the trust region, checks on the quality of $X_{\text{sub}}^k$ and $X_{\text{int}}^k$ are performed and, if necessary, a geometry-improving algorithm is used.

Remark 4.1: An open source Python implementation of OMoRF is available for public use from the Effective Quadratures package (Seshadri and Parks 2017).
Algorithm 1 Optimization by moving ridge functions

1. Let starting point $x_0 \in \mathbb{R}^n$ and initial trust region radius $\Delta_0 > 0$ be given.
2. Set values of algorithmic parameters $\rho_0 = \Delta_0, 0 < \gamma_1 < 1 \leq \gamma_2 \leq \gamma_3, 0 < \eta_1 < \eta_2 < 1, \gamma_s > 0$,
   \[0 < \omega_s < 1\text{ and } 1 \leq d < n.\]
3. Build an initial set $\mathcal{X}^\text{sub}_0$ of $n+1$ samples.
4. Construct $U_0$ with points $\mathcal{X}^\text{sub}_0$ using (22) if $d = 1$ or (25) if $d > 1$.
5. Build an initial set $\mathcal{X}^\text{int}_0$ of $\frac{1}{2}(d+1)(d+2)$ samples.
6. for $k = 0, 1, \ldots$ do
   7. Construct $d$-dimensional quadratic $m_k$ using $\mathcal{J}^\text{int}_k = \{ U_k^T x' \mid x' \in \mathcal{X}^\text{int}_k \}$.
   8. Solve (33) to get $s_k$.
   9. if $\|s_k\| \leq \gamma_s \rho_k$ then
      Set $\Delta_{k+1} = \max(\omega_s \Delta_k, \rho_k)$.
      11. Invoke Algorithm 2 to get $\mathcal{X}^\text{int}_{k+1}, \mathcal{X}^\text{sub}_{k+1}, U_{k+1}, \rho_{k+1}$ and $\Delta_{k+1}$ (evaluating $f$ for any new samples).
   10. end if
   12. go to line 6
   13. end if
   14. Evaluate $f(x_k + s_k)$ and calculate the ratio $r_k$ (34).
   15. Accept/reject step and update trust region radius:
      \[x_{k+1} = \begin{cases} x_k + s_k, & r_k \geq \eta_1, \\ x_k, & r_k < \eta_1, \end{cases}\]
      \[\Delta_{k+1} = \begin{cases} \max(\gamma_2 \Delta_k, \gamma_s \|s_k\|), & r_k \geq \eta_2, \\ \max(\gamma_1 \Delta_k, \|s_k\|, \rho_k), & \eta_1 \leq r_k < \eta_2, \\ \max(\min(\gamma_1 \Delta_k, \|s_k\|), \rho_k), & r_k < \eta_1. \end{cases}\]
   16. Append $x_k + s_k$ to $\mathcal{X}^\text{int}_k$ and $\mathcal{X}^\text{sub}_k$.
   17. if $r_k \geq \eta_1$ then
      18. Remove a point from both $\mathcal{X}^\text{int}_k$ and $\mathcal{X}^\text{sub}_k$ to get $\mathcal{X}^\text{int}_{k+1}$ and $\mathcal{X}^\text{sub}_{k+1}$.
      19. Set $U_{k+1} = U_k$ and $\rho_{k+1} = \rho_k$.
   20. else
      21. Invoke Algorithm 2 to get $\mathcal{X}^\text{int}_{k+1}, \mathcal{X}^\text{sub}_{k+1}, U_{k+1}, \rho_{k+1}$ and $\Delta_{k+1}$ (evaluating $f$ for any new samples).
   22. end if
   23. end for

Remark 4.2: In line 18, a point from each $\mathcal{X}^\text{int}_k$ and $\mathcal{X}^\text{sub}_k$ is removed to get $\mathcal{X}^\text{int}_{k+1}$ and $\mathcal{X}^\text{sub}_{k+1}$. This may be done using any of the geometry-improving algorithms detailed by Conn, Scheinberg, and Vicente (2009). In practice, a modified pivoting algorithm that prioritizes points in the trust region is used (see Section 3 of the online supplemental data).

Remark 4.3: It is assumed in Algorithm 1 that the solution to the trust region subproblem (33) results in a step that satisfies the sufficient decrease condition
\[m_k(U_k^T x_k) - m_k(U_k^T (x_k + s_k)) \geq c_1 \|g_k\| \min \left\{ \Delta_k, \frac{\|g_k\|}{\|H_k\|} \right\}, \tag{35}\]
where $c_1 \in (0, 1/2]$ is a constant and $g_k, H_k$ are the gradient and Hessian of $m_k$ at $x_k$, respectively.

Remark 4.4: Just as in UOBYQA, NEWUOA and BOBYQA, two trust region radii $\Delta_k$ and $\rho_k$ are maintained. However, in this case $\rho_k$ is not explicitly used to detach control of the sampling region from $\Delta_k$. Rather, $\rho_k$ acts as a lower bound for $\Delta_k$, preventing it from shrinking too quickly before the model is sufficiently ‘good’ (Cartis et al. 2019).
Remark 4.5: The convergence of many DFTR algorithms is generally dependent on a so-called criticality step (Conn, Scheinberg, and Vicente 2009). During this step, the accuracy of the model $m_k$ is ensured whenever its gradient is sufficiently small. In Algorithm 1, this has been replaced by a safety step (lines 9–11). This check can be seen as an analogue of the criticality step—see the discussion in Conn, Scheinberg, and Vicente (2009, Section 11.3).

Remark 4.6: The infinity norm $\| \cdot \|_\infty$ is used in this implementation of OMoRF. This allows it to be easily extended to bound-constrained optimization problems by swapping the trust region constraint in subproblem (33) with bound constraints.

4.1. Interpolation set management

In Section 3.3, it was shown that the accuracy of a ridge function model $m_k$ is dependent on two sources of error: information loss by projecting onto a subspace $U_k$ and the response surface error of $m_k$. Ideally, a single interpolation set could be improved to reduce both sources of error. Unfortunately, such an approach would require a-priori knowledge of the subspace $U_k$. Therefore, OMoRF maintains two separate interpolation sets: $X_{\text{sub}}^k$ of $n + 1$ samples for calculating $U_k$, and $X_{\text{int}}^k$ of $\frac{1}{2}(d + 1)(d + 2)$ samples for calculating the coefficients of $m_k$.

The subspace $U_k$ is calculated using either a derivative-free active subspace approach or through polynomial ridge approximation. In either case, the first step is to build a fully linear $n$-dimensional linear interpolator $f$ (20) using the set $X_{\text{sub}}^k$. In the case of derivative-free active subspaces, $U_k$ is simply the one-dimensional active subspace (22). If a greater dimensionality is required, $X_{\text{sub}}^k$ is used to solve the Grassmann manifold optimization problem (25). Note that solving (25) requires an initial guess for $U_k$. In OMoRF, the one-dimensional subspace (22) is appended with its orthogonal complement and used as an initial guess. Also note that, although the solution to problem (25) could also be used in the case of $d = 1$, it was found that the one-dimensional active subspace (22) generally gave superior algorithmic performance. Therefore, this method is only employed in the case where higher dimensions are desired, e.g. when it is believed that a one-dimensional subspace insufficiently describes the underlying problem dimension.

Once the subspace $U_k$ is known, one may be tempted to use the points $Y_{\text{sub}}^k = \{U_k^T x^i \mid x^i \in X_{\text{sub}}^k\}$ to calculate the coefficients of $m_k$. However, these projected samples generally insufficiently span the $d$-dimensional projected space, leading to poor surrogate models. Provided $d \ll n$, determining a more suitable set of $\frac{1}{2}(d + 1)(d + 2)$ samples $X_{\text{int}}^k$ is not only relatively cheap, but can also dramatically improve the quality of the ridge function surrogate. Figure 3 provides an example of the 10-dimensional Styblinski–Tang function

$$f(x) = \sum_{i=1}^{10} 0.5 \left( x_i^4 - 16x_i^2 + 5x_i \right)$$

projected onto a two-dimensional subspace.

From this figure it is clear that, although the set $X_{\text{sub}}^k$ may be well-suited for linear interpolation in 10 dimensions, the projected set $Y_{\text{sub}}^k$ does not span the two-dimensional space very well. In contrast, the projected set $Y_{\text{int}}^k = \{U_k^T \tilde{x}^i \mid \tilde{x}^i \in X_{\text{int}}^k\}$ spans this space effectively, which in turn gives a much more accurate ridge function model. To demonstrate this increase in accuracy, $N = 100,000$ samples $\tilde{x}_i$ were drawn at random from a uniform distribution bounded by the trust region domain. From these samples, the coefficient of determination

$$R^2 = 1 - \frac{\text{SSR}}{\text{SST}},$$

(37)
where
\[
    SSR = \sum_{i=1}^{N} (f(\hat{x}_i) - m(U_k^T \hat{x}_i))^2, \quad SST = \sum_{i=1}^{N} (f(\hat{x}_i) - \bar{f})
\]
and \( \bar{f} = \frac{1}{N} \sum_{i=1}^{N} f(\hat{x}_i) \), was calculated for both of these models; these \( R^2 \) values are given in Figure 3.

### 4.2. Interpolation set updates

Algorithm 1 always includes new sample points as they become available by appending \( x_k + s_k \) to the interpolation sets \( \mathcal{X}_k^{\text{sub}} \) and \( \mathcal{X}_k^{\text{int}} \). When the iterate is successful, i.e. \( r_k \geq \eta_1 \), a point from each \( \mathcal{X}_k^{\text{int}} \) and \( \mathcal{X}_k^{\text{sub}} \) is removed and the algorithm proceeds to the next iteration. When the iterate is not successful, it is necessary to ensure the accuracy of the model before reducing the trust region radius \( \Delta_k \). In this case, a new geometry-improving point may be determined and subsequently added. The full details of this process are provided in Algorithm 2.

There are a few points to note about Algorithm 2. First, the maximum distance of the samples to the current iterate is used to determine whether or not the sample sets need to be improved. If this distance is too large, it indicates the interpolation set has not been updated recently, so it may need improvement. Although other conditions may be used as a measure of the quality of an interpolation set, the maximum distance of the samples to the current iterate gives a quick and simple means of determining whether or not to improve the interpolation set. Similar approaches have been successfully applied in other DFTR methods (Fasano, Morales, and Nocedal 2009; Bandeira, Scheinberg, and Vicente 2012; Cartis et al. 2019). Second, although any geometry-improving algorithm from Conn, Scheinberg, and Vicente (2009) may be used to improve the quality of the sample sets, in practice a modified pivoting algorithm is used. This algorithm can be seen as a modification of Conn, Scheinberg, and Vicente (2009, Algorithm 6.6). In this modified algorithm, priority is given to points that lie within the trust region. The details of this algorithm may be found in Section 3 of the online supplemental data. Third, improvements to \( \mathcal{X}_k^{\text{int}} \) are prioritized over \( \mathcal{X}_k^{\text{sub}} \). This is because \( \mathcal{X}_k^{\text{int}} \) generally has significantly fewer samples than \( \mathcal{X}_k^{\text{sub}} \), so \( \mathcal{X}_k^{\text{int}} \) can be updated more rapidly than \( \mathcal{X}_k^{\text{sub}} \). If all of the points in \( \mathcal{X}_k^{\text{int}} \) are sufficiently close to the current iterate \( x_k \), this indicates that \( \mathcal{X}_k^{\text{int}} \) has been recently improved. In these cases, if the model \( m_k \) needs improving, it may be because the subspace \( U_k \) needs to be updated. Finally, a new subspace \( U_{k+1} \) is calculated whenever the geometry of \( \mathcal{X}_k^{\text{sub}} \) is improved.
Algorithm 2 Interpolation set update for OMoRF

1: Let \( x_k \) be the current iterate, \( \mathcal{X}_k^{\text{sub}} \) be a set of at least \( n + 1 \) samples, \( \mathcal{X}_k^{\text{int}} \) be a set of at least \( \frac{1}{2} (d + 1) (d + 2) \) samples, \( U_k \) be the current subspace and both \( \Delta_k \) and \( \rho_k \) be given.
2: Set values of algorithmic parameters \( 0 < \alpha_1 < \alpha_2 < 1, \epsilon_k > 0 \)
3: if \( \max ||x^i - x_k|| > \epsilon_k \) for \( x^i \in \mathcal{X}_k^{\text{int}} \) then
4: Invoke a geometry-improving algorithm to improve \( \mathcal{X}_k^{\text{int}} \) by finding a new sample point and set \( \mathcal{X}_{k+1}^{\text{int}} = \mathcal{X}_k^{\text{int}} \).
5: Set \( U_{k+1} = U_k, \rho_{k+1} = \rho_k \) and \( \Delta_{k+1} = \Delta_k \).
6: else if \( \max ||x^i - x_k|| > \epsilon_k \) for \( x^i \in \mathcal{X}_k^{\text{sub}} \) then
7: Invoke a geometry-improving algorithm to improve \( \mathcal{X}_k^{\text{sub}} \) by finding a new sample point and set \( \mathcal{X}_{k+1}^{\text{sub}} = \mathcal{X}_k^{\text{sub}} \).
8: Construct \( U_{k+1} \) with points \( \mathcal{X}_k^{\text{sub}} \) using (22) if \( d = 1 \) or (25) if \( d > 1 \), set \( \rho_{k+1} = \rho_k \) and \( \Delta_{k+1} = \Delta_k \).
9: else
10: Set \( \mathcal{X}_{k+1}^{\text{sub}} = \mathcal{X}_k^{\text{sub}} \) and \( \mathcal{X}_{k+1}^{\text{int}} = \mathcal{X}_k^{\text{int}} \).
11: Set \( U_{k+1} = U_k \), and if \( \Delta_{k+1} = \rho_k \), set \( \rho_{k+1} = \alpha_1 \rho_k \) and \( \Delta_{k+1} = \alpha_2 \Delta_k \), otherwise set \( \rho_{k+1} = \rho_k \) and \( \Delta_{k+1} = \Delta_k \).
12: end if
13: return \( \mathcal{X}_{k+1}^{\text{int}}, \mathcal{X}_{k+1}^{\text{sub}}, U_{k+1}, \rho_{k+1}, \Delta_{k+1} \)

This is because improving the geometry of \( \mathcal{X}_k^{\text{sub}} \) improves the quality of the linear interpolator \( \hat{f} \) (20), which, by Lemma 3.2, leads to a more accurate covariance matrix (17). Therefore, improving \( \mathcal{X}_k^{\text{sub}} \) before calculating \( U_{k+1} \) potentially allows the algorithm to find a more suitable dimension-reducing subspace.

5. Numerical results

The performance of OMoRF has been tested against three well-known DFO algorithms: COBYLA, BOBYQA and Nelder–Mead (Nelder and Mead 1965). The Effective Quadratures (Seshadri and Parks 2017) implementation was used for OMoRF, SciPy (Virtanen et al. 2020) was used for COBYLA, Py-BOBYQA (Cartis et al. 2019) was used for BOBYQA and NLopt (Johnson 2018) was used for Nelder–Mead. For BOBYQA, two variants, one with the minimum of \( n + 2 \) interpolation points and another with the default of \( 2n + 1 \), were tested. All of the tested algorithms were provided the same initial starting point \( x_0 \) and arbitrarily chosen characteristic length \( \Delta_0 \). In the case of unconstrained problems, a value of \( \Delta_0 = 0.1 \max(||x_0||_\infty, 1) \) was used, while \( \Delta_0 = 0.1 \min(\max(||x_0||_\infty, 1), ||b - a||_\infty) \) was used for bound-constrained problems. To force the solvers to use all of the available computational budget, the convergence criterion was set to a value of \( 10^{-16} \) such that it was generally not reached. For OMoRF, the following parameter values were used: \( \gamma_1 = 0.5, \gamma_2 = 2.0, \gamma_3 = 2.5, \eta_1 = 0.1, \eta_2 = 0.7, \alpha_1 = 0.1, \alpha_2 = 0.5, \epsilon_k = \max(2\Delta_k, 10\rho_k) \), \( \gamma_s = 0.5 \) and \( \omega_s = 0.5 \).

5.1. Testing methodology

Performance and data profiles (Moré and Wild 2009) have been used for comparing these algorithms on many of the following test problems. These profiles are defined in terms of three characteristics: the set of test problems \( P \), the set of algorithms tested \( S \) and a convergence test \( T \). Given the convergence test, a problem \( p \in P \) and a solver \( s \in S \), the number of function evaluations necessary to pass the
convergence test \( T \) was used as a performance metric \( t_{p,s} \). Moreover, the convergence test

\[
f(x) \leq f_L + \tau (f(x_0) - f_L),
\]

where \( \tau > 0 \) is some tolerance, \( x_0 \) is the starting point and \( f_L \) is the minimum attained value of \( f \) for all solvers \( S \) within a given computational budget for problem \( p \), was used.

The performance profile is defined as

\[
\rho_s(\alpha) = \frac{1}{|P|} \text{size} \left\{ p \in P : \min \{ t_{p,s} : s \in S \} \leq \alpha \right\}.
\]

In other words, \( \rho_s(\alpha) \) is the proportion of problems in \( P \) in which solver \( s \in S \) attains a performance ratio of at most \( \alpha \). In particular, \( \rho_s(1) \) is the proportion of problems for which the solver performs the best for that particular convergence criterion \( t_{p,s} \) and, as \( \alpha \to \infty \), \( \rho_s(\alpha) \) represents the proportion of problems that can be solved within the computational budget. Data profiles are defined as

\[
d_s(\alpha) = \frac{1}{|P|} \text{size} \left\{ p \in P : \frac{t_{p,s}}{n_p + 1} \leq \kappa \right\},
\]

where \( n_p \) is the dimension of problem \( p \in P \). This represents the proportion of problems that can be solved—measured by convergence criterion \( t_{p,s} \)—by a solver \( s \) within \( \kappa (n_p + 1) \) function evaluations (or \( \kappa \) simplex gradients).

### 5.2. CUTEst problems

The CUTEst (Gould, Orban, and Toint 2015) test problem set was used to examine solver performance. From the set of unconstrained and bound-constrained optimization problems, two subsets were defined: (1) 40 problems of moderate dimension \( (10 \leq n < 50) \); and (2) 40 problems of high dimension \( (50 \leq n \leq 100) \). A full list of these problems may be found in Section 4 of the online supplemental data. In order to simulate an environment where a strict computational budget may limit the number of function evaluations available to a solver, a computational budget of 20 simplex gradients—i.e. 20\((n+1)\) function evaluations—was specified. These solvers were tested with a low accuracy requirement of \( \tau = 10^{-1} \) and a high accuracy requirement of \( \tau = 10^{-5} \). In these studies, two comparisons were made. Initially, four variants of OMoRF, with \( d = 1, 2, 3, 4 \), were compared. From this comparison, the best of these solvers was chosen for comparison with the other solvers. In this second comparison, the best solution per problem from the first comparison was retained, even if the solver had been eliminated. This approach was taken in order to avoid performance profile crowding—see Gould and Scott (2016). Note that, to further reduce the crowding effect, the results from BOBYQA with \( n + 2 \) points have been omitted from the plots below. This is because this solver was generally significantly inferior to BOBYQA with \( 2n + 1 \) points.

#### 5.2.1. Moderate dimension problems

The data and performance profiles for all tested variants of OMoRF for the test set of moderate dimension problems are shown in Figure 4. It is clear that OMoRF \( (d = 1) \) significantly outperformed the other solvers. In fact, for the low accuracy requirement cases, this variant of OMoRF was the best solver for more than 90% of the test problems. Although this dropped to 80% for the high accuracy requirement, this was still significantly better than the other solvers. The solver with the next best performance, OMoRF \( (d = 2) \), was the quickest solver to reach convergence for only around 10% of the problems for both the low and high accuracy requirements. Furthermore, it is clear that, for problems of moderate dimension, increasing \( d \) can have a negative effect on the algorithmic performance. This is because, given a subspace \( U_k \), the number of points required to construct a quadratic ridge function is \( O(d^2) \). If \( d \) is not significantly less than \( n \), this requirement can be prohibitive.
Figure 4. Data and performance profiles for OMoRF solvers for problems of moderate dimension from the CUTEst test set at $\tau = 10^{-1}$ and $\tau = 10^{-5}$. (a) Data profile: $\tau = 10^{-1}$. (b) Performance profile: $\tau = 10^{-1}$. (c) Data profile: $\tau = 10^{-5}$ and (d) Performance profile: $\tau = 10^{-5}$.

Owing to the clear advantages in performance, OMoRF ($d = 1$) has been used for comparison with the other solvers. The data and performance profiles for OMoRF ($d = 1$), COBYLA, BOBYQA ($2n + 1$) and Nelder–Mead for the test set of moderate dimension problems are shown in Figure 5. As previously mentioned, the data and performance profiles shown in Figure 5 used the minimum attained value $f_L$ from all solvers, including the other three variants of OMoRF. This explains the relative decrease in performance for OMoRF ($d = 1$) in Figure 5. Nevertheless, for the problems in this test set, OMoRF was generally the quickest solver to achieve convergence at both the low and high accuracy requirements. From the performance profile, one can see that it was the first solver to converge for over 80% of the problems for the low accuracy requirement and nearly 40% for the high accuracy requirement. Additionally, OMoRF was able to make much quicker initial progress than the other methods, as demonstrated in the data profiles. In the case of the low accuracy requirement, nearly 80% of the problems could be solved to convergence within two simplex gradients, compared to 5% for COBYLA and 0% for BOBYQA. This quick convergence is probably due to its ability to model functions with low-dimensional quadratics, allowing it to capture function curvature with significantly fewer samples. However, one point to note is that, as the number of function evaluations increased, BOBYQA was able to solve a larger proportion of the problems at the low accuracy requirement. This suggests that BOBYQA may be a slightly superior general-purpose solver when seeking low accuracy solutions.

5.2.2. High-dimension problems

The data and performance profiles for all tested variants of OMoRF for the test set of high-dimension problems are shown in Figure 6. Just as in the case of problems of moderate dimension, OMoRF
Figure 5. Data and performance profiles for problems of moderate dimension from the CUTEst test set at $\tau = 10^{-1}$ and $\tau = 10^{-5}$.
(a) Data profile: $\tau = 10^{-1}$. (b) Performance profile: $\tau = 10^{-1}$. (c) Data profile: $\tau = 10^{-5}$ and (d) Performance profile: $\tau = 10^{-5}$.

$(d = 1)$ was generally the superior solver for this test set. In particular, it was the fastest solver to reach convergence at both the low and high accuracy requirements. Additionally, OMoRF ($d = 1$) achieved convergence to the high accuracy requirement more than any other solver tested, solving about 70% of the problems using the full computational budget. Interestingly, the other variants of OMoRF were more competitive for the high-dimensional problems than the problems of moderate dimension, with all the solvers achieving convergence to the low accuracy requirement for more than 90% of the problems. In particular, OMoRF ($d = 2$) was able to achieve convergence to the low accuracy requirement for approximately the same proportion of problems as OMoRF ($d = 1$). This relative performance increase is probably due to the fact that, as the dimension of the problem increases, the required $O(d^2)$ samples needed to construct a quadratic ridge function becomes less restrictive.

Although the other variants of OMoRF were more competitive, OMoRF ($d = 1$) was still generally the best performing solver, so this variant was again used for comparison with the other solvers. The data and performance profiles for OMoRF ($d = 1$), COBYLA, BOBYQA ($2n + 1$) and Nelder–Mead for the test set of high-dimension problems are shown in Figure 7. For this set, the superiority of the OMoRF solver over the other algorithms is even more apparent, with its being the fastest solver to achieve convergence for around 90% of the problems at the low accuracy requirement and approximately 45% for the high accuracy requirement. Although BOBYQA still achieved convergence for a slightly larger proportion of the problems at the low accuracy requirement, OMoRF achieved convergence at the high accuracy requirement for a greater proportion of these problems than any other solver. In particular, at the high accuracy requirement, OMoRF was able to achieve convergence for approximately 60% of these problems. COBYLA, the next best solver, managed to achieve convergence for only approximately 50% of these problems.
To demonstrate the efficacy of OMoRF when optimizing computationally intensive high-dimensional functions, design optimization of the ONERA-M6 transonic wing, parameterized by 100 free-form deformation (FFD) points, has been used as a test problem. The objective is to minimize inviscid drag subject to bound constraints on the FFD parameters. This problem has been adapted from an open source tutorial (Palacios and Kline 2017). Furthermore, it has been used for testing design optimization algorithms and approaches in multiple studies (Lukaczyk et al. 2014; Qiu et al. 2018). In this study, this problem has been formulated as

$$\min_{x \in \mathbb{R}^{100}} \ C_D(x)$$

subject to $x \in [-0.1, 0.1]^{100}$

with $x$ denoting the FFD parameters and $C_D(x)$ the drag coefficient. The flight conditions are of steady flight at a free-stream Mach number of 0.8395 and an angle of attack of 3.06°. The Euler solver provided by the open source computational fluid dynamics (CFD) simulation package SU2 (Palacios et al. 2013) was used to evaluate each design. A single CFD simulation required approximately five minutes on 8 CPU cores of a 3.7 GHz Ryzen™ 2700X desktop computer. Given this computational burden, a strict limit of 500 function evaluations was specified when optimizing this problem. It is noted that, although derivatives of this objective function may be obtained using algorithmic differentiation, this problem still provides a useful test problem for high-dimensional, computationally intensive design optimization.
Figure 7. Data and performance profiles for problems of high dimension from the CUTEst test set at $\tau = 10^{-1}$ and $\tau = 10^{-5}$.
(a) Data profile: $\tau = 10^{-1}$. (b) Performance profile: $\tau = 10^{-1}$. (c) Data profile: $\tau = 10^{-5}$ and (d) Performance profile: $\tau = 10^{-5}$.

Figure 8. Convergence of the ONERA M6 design optimization problem with $n = 100$ variables.
For this problem, both OMoRF \((d = 1)\) and OMoRF \((d = 2)\) were included in the solver comparison. Additionally, BOBYQA with both \(n + 2\) and \(2n + 1\) points have been included. Figure 8 shows the convergence plot for this design optimization problem. Although BOBYQA \((n + 2)\) shows very quick initial progress, achieving a drag coefficient of less than \(3 \times 10^{-3}\) within 200 function evaluations, its progress thereafter stalls. In fact, OMoRF \((d = 2)\) outperforms it from 400 function evaluations onward, and from 450 function evaluations onward so do COBYLA and BOBYQA \((2n + 1)\). Moreover, not only does OMoRF \((d = 2)\) show very rapid progress, it ultimately outperforms all the other solvers by achieving the smallest drag coefficient within the computational budget. Interestingly, although OMoRF \((d = 1)\) generally performed quite well in the previous test problems, its performance was significantly worse for this problem. It is hypothesized that, in this case, the underlying problem dimension is best described using more than one dimension. This is in agreement with the findings from previous studies (Lukaczyk et al. 2014; Qiu et al. 2018). In particular, Lukaczyk et al. (2014) discovered that the drag coefficient response of the ONERA-M6 wing parameterized by 50 FFD points was best described using at least a two-dimensional subspace.

6. Conclusion

A novel DFTR method that leverages output-based dimension reduction in a trust region framework has been presented. This approach is based upon the idea that, by reducing the effective problem dimension, functions of moderate to high dimension may be modelled using fewer samples. Using these reduced dimension surrogate models for model-based optimization may then lead to accelerated convergence. Although many functions cannot be modelled to sufficient accuracy by globally defined ridge functions, the use of local subspaces allows for greater flexibility, while also maintaining the computational benefits of dimension reduction. The full linearity of ridge function models has been discussed and, using this discussion, a motivation for using moving ridge functions has been presented. The efficacy of this algorithm was demonstrated on a number of test problems, including high-dimensional aerodynamic design optimization. Future work will focus on providing further theoretical statements on the convergence properties of this algorithm, extending this method to the case of general nonlinear constraints and applying this approach to other optimization problems.

Acknowledgments

The authors offer their sincere thanks Dr Pranay Seshadri for his invaluable assistance and advice when developing this algorithm. The first author would also like to thank Dr Hui Feng for her useful suggestions and complete support.

Disclosure statement

No potential conflict of interest was reported by the author(s).

Funding

This work was supported by the Weir Advanced Research Centre (WARC) through an Engineering and Physical Sciences Research Council (EPSRC) Industrial Cooperative Awards in Science & Technology (CASE) Studentship [Grant Number RG94532].

References

Bandeira, Afonso, Katya Scheinberg, and Luís Vicente. 2012. “Computation of Sparse Low Degree Interpolating Polynomials and Their Application to Derivative-Free Optimization.” Mathematical Programming 134: 223–257. doi:10.1007/s10107-012-0578-z.

Bandeira, Afonso, Katya Scheinberg, and Luís Vicente. 2014. “Convergence of Trust-Region Methods Based on Probabilistic Models.” SIAM Journal on Optimization 24 (3): 1238–1264.

Bergstra, James, and Yoshua Bengio. 2012. “Random Search for Hyper-Parameter Optimization.” Journal of Machine Learning Research 13: 281–305.
Cartis, Coralia, Ian Fiala, Benjamin Marteau, and Lindon Roberts. 2019. “Improving the Flexibility and Robustness of Model-Based Derivative-Free Optimization Solvers.” *ACM Transactions on Mathematical Software* 45 (3): 1–41.

Cartis, Coralia, and Adilet Otemissov. 2020. “A Dimensionality Reduction Technique for Unconstrained Global Optimization of Functions with Low Effective Dimensionality.” arXiv:2003.09673.

Choromanski, Krzysztof, Aldo Pacchiano, Jack Parker-Holder, Yunhao Tang, and Vikas Sindhwani. 2019. “From Complexity to Simplicity: Adaptive ES-Active Subspaces for Blackbox Optimization.” In *Proceedings of the Thirty-Third Conference on Neural Information Processing Systems* (NeurIPS 2019), 10299–10309. https://research.google/pubs/pub48818/.

Conn, Andrew, Nicholas Gould, and Philippe Toint. 2000. *Trust-Region Methods*. Philadelphia, PA: SIAM.

Conn, Andrew, Katya Scheinberg, and Luis Vicente. 2008. “Geometry of Interpolation Sets in Derivative Free Optimization.” *Mathematical Programming* 111 (1): 141–172.

Conn, Andrew, Katya Scheinberg, and Luis Vicente. 2009. “Global Convergence of General Derivative-Free Trust-Region Algorithms to First- and Second-Order Critical Points.” *SIAM Journal on Optimization* 20 (1): 387–415.

Conn, Andrew, Katya Scheinberg, and Luis Vicente. 2009. *Introduction to Derivative-Free Optimization*. Philadelphia, PA: SIAM.

Constantine, Paul G. 2015. *Active Subspaces: Emerging Ideas for Dimension Reduction in Parameter Studies*. Philadelphia, PA: SIAM Spotlights.

Constantine, Paul, and Alireza Doostan. 2017. “Time-Dependent Global Sensitivity Analysis with Active Subspaces for a Lithium Ion Battery Model.” *Statistical Analysis and Data Mining* 10 (5): 243–262.

Constantine, Paul, Eric Dow, and Qiqi Wang. 2014. “Active Subspace Methods in Theory and Practice: Applications to Kriging Surfaces.” *SIAM Journal on Scientific Computing* 36 (4): A1500–A1524.

Diez, Matteo, Emilio Campana, and Frederick Stern. 2015. “Design-Space Dimensionality Reduction in Shape Optimization by Karhunen–Loève Expansion.” *Computer Methods in Applied Mechanics and Engineering* 283 (1): 1525–1544.

Eldar, Yonina, and Gitta Kutyniok. 2012. *Compressed Sensing: Theory and Applications*. Cambridge, UK: Cambridge University Press.

Fasano, Giovanni, José Luis Morales, and Jorge Nocedal. 2009. “On the Geometry Phase in Model-Based Algorithms for Derivative-Free Optimization.” *Optimization Methods and Software* 24 (1): 145–154.

Ghanbari, Hiva, and Katya Scheinberg. 2017. “Black-Box Optimization in Machine Learning with Trust Region Based Derivative Free Algorithm.” arXiv:1703.06925.

Glaws, Andrew, Paul Constantine, John N. Shadid, and Tim Wildey. 2017. “Dimension Reduction in MHD Power Generation Models: Dimensional Analysis and Active Subspaces.” *Statistical Analysis and Data Mining: The ASA Data Science Journal* 10: 312–325.

Gould, Nicholas, Dominique Orban, and Philippe Toint. 2015. “CUTEst: A Constrained and Unconstrained Testing Environment with Safe Threads for Mathematical Optimization.” *Computational Optimization and Applications* 60 (3): 545–557.

Gould, Nicholas, and Jennifer Scott. 2016. “A Note on Performance Profiles for Benchmarking Software.” *ACM Transactions on Mathematical Software* 43 (2): 1–5.

Gross, James, Pranay Seshadri, and Geoffrey Parks. 2020. “Optimisation with Intrinsic Dimension Reduction: A Ridge Informed Trust-Region Method.” In *Proceedings of the AIAA SciTech Forum and Exposition*, Paper No. AIAA 2020-0157. Reston, VA: AIAA. doi:10.2514/6.2020-0157.

Gu, Lei. 2001. “A Comparison of Polynomial Based Regression Models in Vehicle Safety Analysis.” In *Proceedings of the 27th ASME Design Automation Conference*, Vol. 2. New York: ASME.

Hokanson, Jeffrey, and Paul Constantine. 2017. “Data-Driven Polynomial Ridge Approximation Using Variable Projection.” *SIAM Journal on Scientific Computing* 40 (3): 1566–1589.

Johnson, Steven. 2018. “The NLopt Nonlinear-Optimization Package.” http://github.com/stevengj/nlopt.

Kannan, Aswin, and Stefan Wild. 2012. *Obtaining Quadratic Models of Noisy Functions*. Technical Report, Argonne National Laboratory, Argonne, IL. http://www.mcs.anl.gov//uploads/cels/papers/P1975-1111.pdf.

Kipouros, Timoleon, Daniel Jaeggi, William Dawes, Geoffrey Parks, Mark Savill, and John Clarkson. 2008. “Biobjective Design Optimization for Axial Compressors Using Tabu Search.” *AIAA Journal* 46 (3): 701–711.

Kozak, David, Stephen Becker, Alireza Doostan, and Luis Tenorio. 2019. “Stochastic Subspace Descent.” arXiv:1904.01145.

Levina, Tatiana, Yuri Levin, Jeff McGill, and Mikhail Nediak. 2009. “Dynamic Pricing with Online Learning and Strategic Consumers: An Application of the Aggregating Algorithm.” *Operations Research* 57 (2): 327–341.

Lukaczyk, Trent, Paul Constantine, Francisco Palacios, and Juan Alonso. 2014. “Active Subspaces for Shape Optimization.” In *Proceedings of the 10th AIAA Multidisciplinary Design Optimization Specialist Conference*. Reston, VA: AIAA.

Moré, Jorge, and Stefan Wild. 2009. “Benchmarking Derivative-Free Optimization Algorithms.” *SIAM Journal on Optimization* 20 (1): 172–191.

Nelder, John, and Roger Mead. 1965. “A Simplex Method for Function Minimization.” *The Computer Journal* 7 (4): 308–313.
Palacios, Francisco, Michael Colonno, Aniket Aranake, Alejandro Campos, Sean Copeland, Thomas Economon, Amrita Lonkar, Trent Luu, and Juan Alonso. 2013. “Stanford University Unstructured (SU2): An Open-Source Integrated Computational Environment for Multi-Physics Simulation and Design.” In Proceedings of the 51st AIAA Aerospace Sciences Meeting including the New Horizons Forum and Aerospace Exposition 2013. Reston, VA: AIAA.

Palacios, Francisco, and Heather Kline. 2017. “Constrained Shape Design of a Transonic Inviscid Wing.” https://su2code.github.io/tutorials/Inviscid_3D_Constrained_ONERAM6/.

Parsons, Lance, Ehtesham Haque, and Huan Liu. 2004. “Subspace Clustering for High Dimensional Data: A Review.” SIGKDD Explorations, Newsletter of the ACM Special Interest Group on Knowledge Discovery and Data Mining 6 (1): 90–105.

Pinkus, Allan. 2015. Ridge Functions. Cambridge, UK: Cambridge University Press.

Powell, Michael. 1994. “A Direct Search Optimization Method that Models the Objective and Constraint Functions by Linear Interpolation.” In Advances in Optimization and Numerical Analysis, 51–67. Dordrecht, The Netherlands: Springer. doi:10.1007/978-94-015-8330-5_4.

Powell, Michael. 2006. “The NEWUOA Software for Unconstrained Optimization Without Derivatives.” In Large-Scale Nonlinear Optimization, 255–297. Boston, MA: Springer. doi:10.1007/0-387-30065-1_16.

Powell, Michael. 2009. The BOBYQA Algorithm for Bound Constrained Optimization Without Derivatives. Technical Report DAMTP 2009/NA06, Department of Applied Mathematics and Theoretical Physics, University of Cambridge, Cambridge, UK. http://www.damtp.cam.ac.uk/user/na/NA_papers/NA2009_06.pdf.

Qiu, Yasong, Junqiang Bai, Nan Liu, and Chen Wang. 2018. “Global Aerodynamic Design Optimization Based on Data Dimensionality Reduction.” Chinese Journal of Aeronautics 31 (4): 643–659.

Seshadri, Pranay, and Geoffrey Parks. 2017. “Effective-Quadratures (EQ): Polynomials for Computational Engineering Studies.” The Journal of Open Source Software 2 (11). doi:10.21105/joss.00166.

Seshadri, Pranay, Shahrokh Shahpar, Paul Constantine, Geoffrey Parks, and Mike Adams. 2018. “Turbomachinery Active Subspace Performance Maps.” Journal of Turbomachinery 140 (4): 041003. doi:10.1115/1.4039991.

Virtanen, Pauli, Ralf Gommers, Travis Oliphant, Matt Haberland, Tyler Reddy, David Cournapeau, Evgeni Burovski, et al. 2020. “SciPy 1.0: Fundamental Algorithms for Scientific Computing in Python.” Nature Methods 17: 261–272.

Wendor, Andrew, Emilio Botero, and Juan Alonso. 2016. “Comparing Different Off-the-Shelf Optimizers’ Performance in Conceptual Aircraft Design.” In Proceedings of the 17th AIAA/ISSMO Multidisciplinary Analysis and Optimization Conference. Reston, VA: AIAA.

Wong, Chun Yui, Pranay Seshadri, Geoffrey Parks, and Mark Girolami. 2019. “Embedded Ridge Approximations: Constructing Ridge Approximations Over Localized Scalar Fields for Improved Simulation-Centric Dimension Reduction.” arXiv:1907.07037.

Zhao, Muhan, Shahrouz Ryan Alimo, and Thomas Bewley. 2018. “An Active Subspace Method for Accelerating Convergence in Delaunay-Based Optimization via Dimension Reduction.” In Proceedings of the IEEE Conference on Decision and Control, 2765–2770. Piscataway, NJ: IEEE.