Optimisation with Intrinsic Dimension Reduction: A Ridge Informed Trust-Region Method

James C. Gross[*], Pranay Seshadri[†], Geoffrey T. Parks[‡]

University of Cambridge, Cambridge, CB2 1PZ, U.K.

The Alan Turing Institute, London, NW1 2DB, U.K.

High-dimensional design optimisation continues to present many challenges for engineers. Traditional model-based optimisation strategies can be difficult to implement for high-dimensional problems, as surrogate models often require sample sizes which are exponential in the number in number of variables. In this paper, we present an algorithm which combines ideas from ridge function approximations and trust-region methods to perform optimisation on high-dimensional functions with underlying low-dimensional structure. This approach allows us to efficiently explore the design space by focusing on the few directions in which the objective varies the most. By slowly increasing the problem dimension, this algorithm also can further explore regions of interest in detail. This allows for more accurate solutions than previous approaches which have used ridge functions during optimisation. We test this algorithm against a number of common model-based optimisation methods, and it is shown to be effective for a class of high-dimensional problems. In particular, we apply the method to a NACA 0012 aerofoil to obtain an optimal design with respect to drag. During this study, we demonstrate how ridge functions can be vital tools for design space exploration, and, with our algorithm, how they can be used for optimisation refinement.

Nomenclature

Optimisation

$\mathbb{R}^n$	Set of real numbers of dimension n
$\mathbf{a}$, $\mathbf{b}$	Lower and upper bounds
$\mathbf{x}$	Full-space design vector
$\mathbf{y}$	Reduced-space design vector
$\mathcal{X}$	Design space of $\mathbf{x}$
$\mathcal{Y}$	Design space of $\mathbf{y}$
$f(\cdot)$	True function
$\nabla(\cdot)$	Gradient of a scalar function
$\nabla^2(\cdot)$	Hessian of a scalar function
$\|\cdot\|$	Vector or matrix norm
$\mathbf{x}_k$	Solution at k
$m_k(\cdot)$	Model of true function at k
Δ_k	Trust-region radius at k
r_k	Ratio of actual to predicted reduction at k

Active subspaces

$\mathbf{C}$	Covariance matrix
$\mathbf{W}$	Eigenvectors of $\mathbf{C}$
$\boldsymbol{\Lambda}$	Eigenvalues of $\mathbf{C}$
$\mathbf{W}_1$	Active subspaces
$\mathbf{W}_2$	Inactive subspaces
$\boldsymbol{\Lambda}_1$	Eigenvalues corresponding to $\mathbf{W}_1$
$\boldsymbol{\Lambda}_2$	Eigenvalues corresponding to $\mathbf{W}_2$
$\mathbb{E}(\cdot)$	Expected value of a random quantity
$g(\cdot)$	Conditional expectation function

Subscript

i	Coordinate number
k	Iteration number

Superscript

i	Point in sample set

[*]PhD Student, Department of Engineering, University of Cambridge, `jg847@cam.ac.uk`.

[†]Post-Doctoral Research Fellow, Department of Engineering, University of Cambridge and Group Leader (Aerospace), Data-Centric Engineering, The Alan Turing Institute, London, `ps583@cam.ac.uk`.

[‡]Reader in Nuclear Engineering, Department of Engineering, University of Cambridge, `gtp10@cam.ac.uk`.

I. Introduction

Given a quantity of interest (qoi) $f(\mathbf{x})$ with input parameters $\mathbf{x} \in \mathbb{R}^n$, a *ridge function* approximation is defined as

$$\hat{f}(\mathbf{W}_1^T \mathbf{x}) \approx f(\mathbf{x}), \tag{1}$$

where $\hat{f} : \mathbb{R}^d \to \mathbb{R}$ is a parametric function and $\mathbf{W}_1 \in \mathbb{R}^{n \times d}$ with $d < n$. That is, a qoi may be approximated by a function of d linear combinations of the input variables. The benefits of using ridge function approximations become apparent as the number of model inputs n increases. To see this, consider an n-dimensional surrogate model of an unknown function f, where evaluating $f(\mathbf{x})$ is computationally prohibitive; examples include computational fluid dynamics (CFD) and finite element method (FEM) simulations. Precisely determining the parameters (also known as hyperparameters) of the model $\hat{f}$ requires at least as many function evaluations as the number of unknown parameters—and usually much more. This is typically undertaken using an appropriately tailored *design of experiment* (doe). At the same time, the number of unknown parameters of a surrogate often increases exponentially with n, increasing the need for more doe points and thus simulation evaluations. This phenomenon is observed in many surrogate models, and has been appropriately termed the *curse of dimensionality*.[1] To clarify this notion, consider a doe with a range of 10 values per variable. Even for a problem of only 10 variables, we would require 10^{10} samples. If each simulation takes only 1 second, it would take well over 300 years to acquire all of the simulation qois! On the other hand, if the dimensions of the model were reduced to two, one would only require 10^2 samples — and less than 2 minutes to simulate these. Therefore, it is apparent that if the model output admits some low-dimensional structure, then significant computational savings can be had.

Fortunately, many physical problems naturally exhibit such low-dimensional behaviour:[2] for instance, turbomachinery design,[3] wing design,[4] and magnetohydrodynamic power generation.[5] For problems with such low-dimensional structure, ridge function approximations have increasingly been used in reduced order modelling,[6,7] integration,[5] sensitivity studies,[3,8] and even data visualisation in a virtual reality environment.[9] However, their use in optimisation has been rather limited.[4,10,11] Given the opportunities these ridge function approximations afford, a closer examination of ridge functions in the context of optimisation is necessary.

Lukaczyk et al.[4] used ridge function approximations to identify possible optimal designs for a constrained design optimisation of an ONERA-M6 wing with 50 free-form deformation (FFD) parameters. The objective for this problem was to minimise the drag coefficient with the constraint that the lift coefficient did not fall below a given value. Global ridge function approximations were constructed for both the drag and lift coefficients of the wing. By interrogating the ridge functions in their respective subspaces, they were able to determine candidate optimal solutions. Although their research was focused on using ridge functions for preliminary design optimisation studies, they did not address the possibility of using such ridge approximations for greater refinement during optimisation. Zhao et al.[10] demonstrated that such optimisation refinement studies were possible by using ridge function surrogates in a Delaunay-based optimisation via global surrogates (Δ-DOGS) method. This approach sought to approximately solve the bound-constrained optimisation problem

$$\begin{aligned} \min_{\mathbf{x}} \quad & f(\mathbf{x}) \\ \text{subject to} \quad & \mathbf{a} \leq \mathbf{x} \leq \mathbf{b}, \end{aligned} \tag{2}$$

where $f(\mathbf{x})$ is an unknown objective function which admits a ridge approximation (1) and $\mathbf{a}, \mathbf{b} \in \mathbb{R}^n$ are lower and upper bounds, respectively. This was achieved by iteratively determining the location within the design space with the highest probability to achieve a function value less than or equal to some prescribed target f_0. In an approach similar to Bayesian optimisation,[12] a cheap surrogate model is constructed from an interpolating spline and an uncertainty function, and minimised at each iteration to determine the location of interest for the next iteration. The novelty of this approach was that the spline model was constructed in a reduced space using active subspaces.[13] This approach allowed the Δ-DOGS method to be applied to problems of higher dimensions than previous studies.

In this paper, we propose a trust-region algorithm to solve the bound-constrained optimisation problem (2) for objectives with inherent low-dimensional structure. Trust-region algorithms have been chosen owing to the simplicity of their underlying surrogate models and their useful convergence properties (Conn et al.,[14] Chapters 6 and 9). However, as the problem dimension increases, the number of function evaluations for surrogate model construction may increase drastically, making the approach less feasible for high-dimensional design optimisation. In our approach, ridge function surrogates are used to reduce the effective dimension of the problem, thereby reducing the number of evaluations needed for surrogate model construction. Potential solutions are then obtained by minimising these *reduced dimension surrogates* with the added constraint that the solution must be contained within a small region of trust. The size of this *trust-region* will be updated at each iteration by determining if the model reduction sufficiently predicts the true function reduction. The benefits of our ridge informed trust-region (RITR) method are two-fold: (i) by using ridge function approximations, the number of required function evaluations can be drastically reduced; and (ii) by constructing local models around a current iterate, the model accuracy can be improved within the trust-region. It will be shown that, in many cases, this approach can significantly reduce the number of function evaluations needed to converge to obtain a near-optimal solution, thereby facilitating its use within the aerodynamic design community.

The remainder of this paper is structured as follows. In Section II, we survey trust-region methods for optimisation and discuss ridge approximations. Ideas from our examination of prior work set the stage for a new optimisation algorithm. We present the RITR method in Section III; its performance is evaluated and compared with that of other optimisation techniques for both high-dimensional optimisation of analytical functions and for an aerofoil design optimisation problem in Section IV. Finally, Section V sets out our concluding remarks and highlights avenues of future research.

II. Trust-region methods and ridge functions

II.A. Trust-region methods

Trust-region methods replace the unconstrained optimisation problem

$$\min_{\mathbf{x}} \quad f(\mathbf{x}) \tag{3}$$

with a sequence of trust-region constrained optimisation subproblems

$$\begin{aligned} \min_{\mathbf{s}} \quad & m_k(\mathbf{x}_k + \mathbf{s}) \\ \text{subject to} \quad & \|\mathbf{s}\| \leq \Delta_k, \end{aligned} \tag{4}$$

where $\mathbf{x}_k$ is the current iterate and $m_k(\mathbf{x})$ is a simple model which approximates the function $f(\mathbf{x})$ in the trust-region

$$B(\mathbf{x}_k, \Delta_k) := \{\mathbf{x} \in \mathbb{R}^n \mid \|\mathbf{x} - \mathbf{x}_k\| \leq \Delta_k\}, \tag{5}$$

with trust-region radius Δ_k. The particular choice of norm $\|\cdot\|$ is an algorithmic decision with little effect on the convergence of the algorithm in the unconstrained case.[15] Solving the subproblem (4) gives the step $\mathbf{s}_k$. Given a current iterate $\mathbf{x}_k$, the candidate for the next iterate $\mathbf{x}_{k+1}$ is then $\mathbf{x}_k + \mathbf{s}_k$. The choice of accepting or rejecting a candidate is undertaken by comparing the model m_k and the true function f. More specifically, we calculate the ratio

$$r_k = \frac{\text{actual reduction}}{\text{predicted reduction}} := \frac{f(\mathbf{x}_k) - f(\mathbf{x}_k + \mathbf{s}_k)}{m_k(\mathbf{x}_k) - m_k(\mathbf{x}_k + \mathbf{s}_k)}. \tag{6}$$

The predicted reduction will always be nonnegative as the step $\mathbf{s}_k$ is obtained by minimising the model m_k over a feasible region including $\mathbf{s}_k = \mathbf{0}$. Therefore, a negative r_k indicates the candidate has led to an increase in the true function, and so is automatically rejected; whereas, a positive r_k indicates that the candidate has led to a decrease in the true function. If r_k is close to one, then the model shows good agreement with the true function, and a value close to zero shows otherwise. Therefore, if r_k is greater than some critical positive value η_1, the candidate is accepted and the trust-region radius is increased. Conversely, if r_k is less than some small positive value η_0, the candidate is rejected and the trust-region radius is shrunk. If $\eta_0 \leq r_k < \eta_1$, the candidate is accepted, but the trust-region radius is maintained. The convergence of this strategy depends on increasing model accuracy as the trust-region radius decreases (Conn et al.,[14] Sections 6.4 and 6.5). For many *standard* trust-region methods, a common model choice is the Taylor series expansion centered around the current iterate $\mathbf{x}_k$, given by

$$m_k(\mathbf{x}_k + \mathbf{s}) = f(\mathbf{x}_k) + \nabla f(\mathbf{x}_k)^T \mathbf{s} + \frac{1}{2}\mathbf{s}^T \mathbf{B}_k \mathbf{s}, \tag{7}$$

where $\mathbf{B}_k$ is a symmetric matrix approximating the Hessian $\nabla^2 f(\mathbf{x}_k)$. If this particular model is used, the difference between $m_k(\mathbf{x}_k + \mathbf{s})$ and $f(\mathbf{x}_k + \mathbf{s})$ is $\mathcal{O}(\|\mathbf{s}\|^2)$ (Nocedal and Wright,[15] page 67). The algorithm is said to have converged to a locally optimal solution when Δ_k decreases below some small critical value $\Delta_{\min} > 0$, signifying no further improvements can be made.

In situations where derivatives are too noisy to be reliably used or computationally too expensive to obtain, one may employ trust-region methods which use interpolation or regression models. Given a set of samples $\mathbf{X} = \{\, \mathbf{x}^0, \mathbf{x}^1, \ldots, \mathbf{x}^p \,\}$ and basis functions $\phi(\mathbf{x}) = \{\, \phi_0(\mathbf{x}), \phi_1(\mathbf{x}), \ldots, \phi_q(\mathbf{x}) \,\}$, a regression model is formed by

$$m_k(\mathbf{x}) = \sum_{j=0}^{q} \alpha_j \phi_j(\mathbf{x}), \tag{8}$$

where α_j for $j = 0, \ldots, q$ are the coefficients of the model. In the case of interpolation, these coefficients are determined by solving the linear system

$$\mathbf{M}(\phi, \mathbf{X})\boldsymbol{\alpha} = \mathbf{f}, \tag{9}$$

where

$$\mathbf{M}(\phi, \mathbf{X}) = \begin{bmatrix} \phi_0(\mathbf{x}^0) & \phi_1(\mathbf{x}^0) & \ldots & \phi_q(\mathbf{x}^0) \\ \phi_0(\mathbf{x}^1) & \phi_1(\mathbf{x}^1) & \ldots & \phi_q(\mathbf{x}^1) \\ \vdots & \vdots & \vdots & \vdots \\ \phi_0(\mathbf{x}^p) & \phi_1(\mathbf{x}^p) & \ldots & \phi_q(\mathbf{x}^p) \end{bmatrix}, \quad \boldsymbol{\alpha} = \begin{bmatrix} \alpha_0 \\ \alpha_1 \\ \vdots \\ \alpha_q \end{bmatrix} \quad \text{and} \quad \mathbf{f} = \begin{bmatrix} f(\mathbf{x}^0) \\ f(\mathbf{x}^1) \\ \vdots \\ f(\mathbf{x}^p) \end{bmatrix}. \tag{10}$$

Interpolation/regression models by default do not possess the trait that decreasing the trust-region radius yields greater accuracy. This nullifies any convergence guarantees established for standard trust-region methods. Therefore, to ensure convergence of these methods one must ensure such models satisfy Taylor-like error bounds

$$\begin{aligned} |f(\mathbf{x}_k + \mathbf{s}) - m_k(\mathbf{x}_k + \mathbf{s})| &\leq \kappa_1 \Delta_k^2 \\ \|\nabla f(\mathbf{x}_k + \mathbf{s}) - \nabla m_k(\mathbf{x}_k + \mathbf{s})\| &\leq \kappa_2 \Delta_k \end{aligned} \tag{11}$$

with constants $\kappa_2, \kappa_2 > 0$ for all $\mathbf{x}_k + \mathbf{s} \in B(\mathbf{x}_k, \Delta_k)$. Models which satisfy (11) are termed *fully linear within the trust-region* $B(\mathbf{x}_k, \Delta_k)$. Many types of models can satisfy full linearity, including polynomial response surfaces,[16,17] radial basis functions,[18] and Gaussian process regression models.[19] In this paper, we will only consider the case where m_k is a polynomial response surface of degree r.

To ensure full linearity, certain geometric conditions on the sample set $\mathbf{X}$ must be satisfied. Foremost, $\mathbf{X}$ must be such that the solution to the linear system (9) exists. In the case of interpolation, the set $\mathbf{X}$ is termed *poised* if the corresponding matrix $\mathbf{M}(\phi, \mathbf{X})$ is nonsingular (Conn et al.,[20] Definitions 3.1 and 4.2). If our polynomial model m_k is constructed from a nonpoised set $\mathbf{X}$, then there exists nonzero coefficients $\boldsymbol{\alpha}$ such that $m_k(\mathbf{x}) = 0$ for all $\mathbf{x} \in \mathbf{X}$. Intuitively, $\mathbf{X}$ is nonpoised if all its constituent points lie on a polynomial manifold of degree r or less. For instance, six points that lie on a line or a circle are nonpoised for quadratic interpolation in two dimensions. On the other hand, a model m_k built from a set which is nearly nonpoised (e.g. six points that nearly lie on a line or circle) is not guaranteed to be fully linear. Figure 1 shows several examples of poised sets for quadratic interpolation in 2D. These figures also show the true function to be approximated (blue surface) and the quadratic model constructed from the poised set (orange surface). From inspection of these figures, one can see that, although all of these interpolations sets are poised, the quality of the model can still be quite poor if the set is nearly nonpoised.

It is clear then that one needs to use *well-poised* sample sets, i.e. sets which have sufficient distance from a nonpoised set. A poised set $\mathbf{X} = \{\, \mathbf{x}^0, \mathbf{x}^1, \ldots, \mathbf{x}^p \,\}$ is said to be Λ*-poised for interpolation* in a region $B \subset \mathbb{R}^n$, if

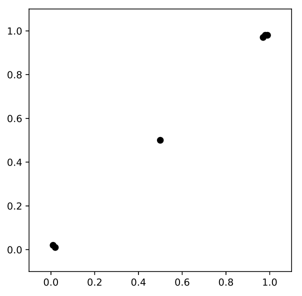
$$\max_{0 \leq i \leq p} \max_{\mathbf{x} \in B} |\Lambda_i(\mathbf{x})| \leq \Lambda \tag{12}$$

where $\Lambda_0, \ldots, \Lambda_p$ are the Lagrange polynomials for $\mathbf{x}^0, \mathbf{x}^1, \ldots, \mathbf{x}^p$ (Conn et al.,[20] Definitions 3.3 and 3.6)). If $\mathbf{X}$ is Λ-poised with sufficiently small Λ, then it is considered well-poised. Interestingly, it has been proven (Conn et al.,[21] Theorem 1) that if $\mathbf{X}$ is Λ-poised in $B(\mathbf{0}, 1)$ (i.e. the unit ball centred at the origin), then

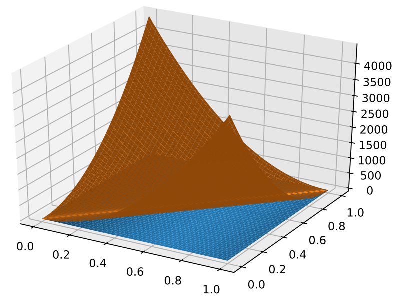
$$\|\mathbf{M}(\bar{\phi}, \mathbf{X})^{-1}\| \leq \theta \Lambda \tag{13}$$

where $\theta > 0$ is dependent on n and r, and $\bar{\phi}(\mathbf{x})$ is the natural monomial basis

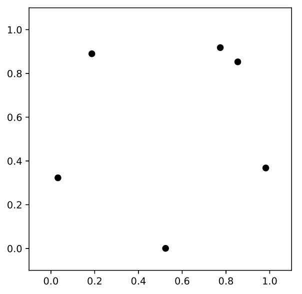
$$\bar{\phi}(\mathbf{x}) = \{\, 1, x_1, \ldots, x_n, \frac{1}{2}x_1^2, x_1 x_2, \ldots, \frac{1}{r!}x_n^r \,\}. \tag{14}$$



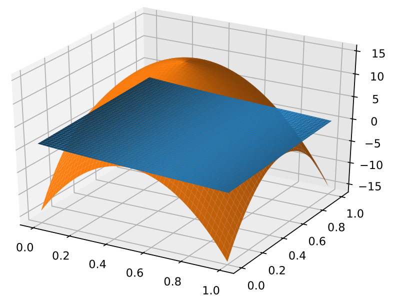
(a)



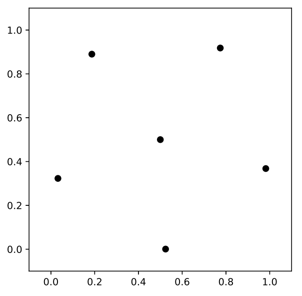
(b)



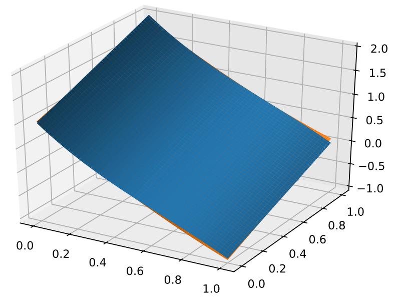
(c)



(d)



(e)



(f)

Figure 1: Examples of poised sets and the corresponding quadratic interpolation model (orange surface) for approximating the true function (blue surface). Subfigures (a) and (b) correspond to a poised set nearly on a line, (c) and (d) correspond to a poised set nearly on a circle, and (e) and (f) correspond to a well-poised set.

Numerous algorithms for constructing and maintaining well-poised sample sets are presented by Conn, Scheinberg, and Vicente (see Chapter 6 in[20]).

A basic interpolation-based trust-region algorithm is shown in Algorithm 1. There are a few remarks to make regarding this algorithm. First, the model construction step is particularly critical. Choosing sample points which are well-poised is paramount to the accuracy of the models, and hence to the convergence of Algorithm 1. However, it is not necessary for the points to be well-poised at every iteration. Rather, the sample set $\mathbf{X}$ must be well-poised at certain key steps of the algorithm (e.g. when decreasing the trust-region radius) (Conn et al.,[14] Chapter 9). In practice, it is important for the sample points to be sufficiently local to the current iterate $\mathbf{x}_k$. Many algorithms will satisfy this constraint by periodically replacing the point within the sample set $\mathbf{X}$ which is furthest from $\mathbf{x}_k$ by a new point contained within the trust-region $B(\mathbf{x}_k, \Delta_k)$.[22,23] Second, if m_k is a quadratic, the trust-region subproblem (4) may be solved nearly exactly when either the

Algorithm 1 Basic interpolation-based trust-region algorithm for unconstrained optimisation problems

```
1:  Phase 0: Algorithm initialisation
2:    Let initial point $\mathbf{x}_0$, initial trust-region radius $\Delta_k$, minimum trust-region radius $\Delta_{\min}$, maximum
      trust-region radius $\Delta_{\max}$, $0 < \gamma_0 < 1 \le \gamma_1$, and $0 \le \eta_0 \le \eta_1 < 1$ be given
3:  while $\Delta_k \ge \Delta_{\min}$ do
4:    Phase 1: Model construction step
5:      Choose sample points $\mathbf{X}$ and construct model $m_k$
6:    Phase 2: Step calculation
7:      Compute step $\mathbf{s}_k$ by solving (4)
8:    Phase 3: Trust-region update
9:      Compute $r_k$ using (6)
10:     if $r_k \ge \eta_1$ then
11:       $\Delta_{k+1} \leftarrow \min\{\gamma_1 \Delta_k,\ \Delta_{\max}\}$
12:     else if $\eta_0 \le r_k < \eta_1$ then
13:       $\Delta_{k+1} \leftarrow \Delta_k$
14:     else
15:       $\Delta_{k+1} \leftarrow \gamma_0 \Delta_k$
16:     end if
17:   Phase 4: Current iterate update
18:     Choose $\mathbf{x}_{k+1}$ to be sample point with smallest function value
19:     $k \leftarrow k + 1$
20: end while
```

Euclidean or Chebyshev norms are used (Conn et al.,[14] Chapter 7). However, exact solutions to (4) are not required to guarantee convergence of Algorithm 1 (Conn et al.,[14] Chapter 6). Therefore, in the case of quadratic models m_k, a number of algorithms which seek approximate solutions to (4) very efficiently may be used (Nocedal and Wright,[15] Section 4.1). Finally, in a standard trust-region approach, the new iterate is chosen strictly from the solution to the step calculation phase. For an interpolation-based method, new samples are regularly being drawn from the input space to construct new models or to ensure full linearity. These samples may, in fact, provide a better reduction than the candidate solution, so it may be beneficial to select the new iterate as any point which has been evaluated thus far.

The efficacy of interpolation-based methods has been demonstrated for a number of optimisation problems of dimensions less than 20.[24–26] However, these methods are often susceptible to the curse of dimensionality. For instance, if an order r polynomial model of a n-dimensional function is used, then there are $\binom{n+r}{r}$ coefficients to be determined at each step. Even for low-order approximations (e.g. $r = 1$ or 2), the number of function evaluations to obtain these coefficients may be prohibitively high. A number of algorithms account for this in a variety of ways. The well-known algorithm COBYLA[27] uses local linear models — only requiring $n+1$ samples. However, linear models do not contain curvature information, so convergence may be slow.[28,29] Some algorithms reduce computation by constructing under-determined quadratic interpolation models, using more points than necessary for a linear model but fewer than for a quadratic model. For example, the algorithm BOBYQA[22] for bound-constrained optimisation problems requires $2n + 1$ points to construct local quadratic models at each iteration. Although this greatly reduces the number of function evaluations, it may still be too much for design optimisation problems of high-dimension. The Nelder-Mead method[30] is a direct search method which maintains a set of $n + 1$ samples at each iteration to construct an n dimensional simplex. Although this method is not an interpolation-based method, it has been included for the sake of comparison due to its popularity over the last 50 years. The Nelder-Mead method is known to converge to suboptimal points for even strictly convex functions.[31,32] However, it can often give 'good' reductions in the function value after a relatively small number of function evaluations,[33] so its popularity has remained steady. Unfortunately, the performance of the Nelder-Mead method may also significantly deteriorate as the problem dimension increases.[34]

II.B. Ridge functions

Ridge function approximations can be used to reduce the effective dimension of a high-dimensional problem and have been constructed using a number of methods.[13,35–37] One promising set of ideas has been rooted in *active subspaces*[13]—a subspace-based dimension reduction technique framework—which seeks to find the few linear combinations of the input space which best describe the variability of a qoi. The effective dimension of the problem is reduced by projecting the input parameters over this subspace, resulting in a new set of design variables with reduced dimension—the *active variables.* Once the effective dimension of the problem has been reduced by finding the active variables, one can construct a reduced dimension surrogate model (i.e. a ridge function approximation).

Consider a scalar function f of design variables $\mathbf{x} \in \mathcal{X} = [-1,1]^n$ supported by the uniform probability density function $\rho(\mathbf{x})$. The active subspaces of this function can be found by determining the covariance matrix

$$\mathbf{C} = \int (\nabla f(\mathbf{x}))(\nabla f(\mathbf{x}))^T \rho \ d\mathbf{x}. \tag{15}$$

This matrix is symmetric, positive semidefinite, so we may calculate its real eigenvalue decomposition

$$\mathbf{C} = \mathbf{W}\mathbf{\Lambda}\mathbf{W}^T. \tag{16}$$

By partitioning the eigenvectors and eigenvalues

$$\mathbf{W} = \begin{bmatrix}\mathbf{W}_1 & \mathbf{W}_2\end{bmatrix}, \quad \mathbf{\Lambda} = \begin{bmatrix}\mathbf{\Lambda}_1 & \\ & \mathbf{\Lambda}_2\end{bmatrix} \tag{17}$$

where $\mathbf{W}_1$ and $\mathbf{\Lambda}_1$ contain the first d eigenvectors and eigenvalues, the active variables can be defined as the reduced coordinates

$$\mathbf{y} = \mathbf{W}_1^T\mathbf{x}. \tag{18}$$

A ridge function approximation can then be determined by constructing a surrogate model over the active coordinates $\mathbf{y}$, resulting in the desired form

$$f(\mathbf{x}) \approx \hat{f}(\mathbf{y}) = \hat{f}(\mathbf{W}_1^T\mathbf{x}). \tag{19}$$

We remark here that, because our design space is constrained to lie within the hypercube $[-1,1]^n$, the bound-constrained optimisation problem (2) should be scaled and translated such that this assumption holds true. Additionally, computing the covariance matrix $\mathbf{C}$ (16) requires n-dimensional integration, which is impractical in all but special cases. In practice, $\mathbf{C}$ is approximated using a number of techniques. Perhaps the most common method is through the Monte Carlo estimate

$$\mathbf{C} \approx \frac{1}{M}\sum_{i=1}^{M}(\nabla f(\mathbf{x}_i))(\nabla f(\mathbf{x}_i))^T, \tag{20}$$

where $\mathbf{x}_i$ are drawn randomly (uniformly) from $[-1,1]^n$ with M being the number of samples. Constantine[13] suggests choosing $M = \alpha k \log n$, where α is an oversampling factor with a value between 2 and 10 and k is a value one greater than the expected reduced dimension d. If d is not known a priori, one can choose $k = n+1$, but significant computational savings may be had by choosing a smaller value. The reduced dimension d is chosen such that the remaining eigenvectors $\mathbf{W}_2$ are approximately in the nullspace of the covariance matrix $\mathbf{C}$, meaning the *inactive variables* $\mathbf{z} = \mathbf{W}_2^T\mathbf{x}$ have significantly less of an effect on the function $f(\mathbf{x})$. The choice of d is left to the designer; however, the eigenvalues $\mathbf{\Lambda}_2$ corresponding to $\mathbf{W}_2$ should have values significantly lower than those in $\mathbf{\Lambda}_1$. Therefore, d is often chosen heuristically by investigating the eigenvalue decay (see Section 1.2 in Constantine[13]). Finally, given a reduced coordinate $\mathbf{y} = \mathbf{W}_1^T\mathbf{x}$, there are often many $\mathbf{x}$ which map to $\mathbf{y}$. Unless f is an exact ridge function, the points $\mathbf{x}$ will give slightly different function evaluations $f(\mathbf{x})$. It then makes sense to consider the conditional expectation of f given the active coordinate $\mathbf{y}$

$$g(\mathbf{y}) = \mathbb{E}[f|\mathbf{y}] := \int f(\mathbf{W}_1\mathbf{y} + \mathbf{W}_2\mathbf{z})\pi_{Z|Y} d\mathbf{z} \tag{21}$$

where $\pi_{Z|Y}$ is the conditional density of the inactive variables $\mathbf{z}$ given a specific choice of active variable $\mathbf{y}$. The domain of g is the d-dimensional *zonotope*

$$\mathcal{Y} = \{\, \mathbf{y} = \mathbf{W}_1^T\mathbf{x} \mid \mathbf{x} \in [-1,1]^n \,\}. \tag{22}$$

Essentially, $\mathcal{Y}$ is the d-dimensional 'shadow' of the n-dimensional design space $\mathcal{X}$. Although the conditional expectation g is generally not known, we may derive confidence intervals using a Monte Carlo estimate

$$\hat{g}(\mathbf{y}) = \frac{1}{N}\sum_{i=1}^{N} f(\mathbf{W}_1\mathbf{y} + \mathbf{W}_2\mathbf{z}_i) \tag{23}$$

where samples $\mathbf{z}_i$ are drawn independently from $\pi_{Z|Y}$. The benefit of using active subspaces is that, provided the eigenvalues in $\mathbf{\Lambda}_2$ are sufficiently small, it may be sufficient to use $N = 1$ (Constantine et al.,[38] Theorem 3.1), thus significantly reducing the computational overhead during surrogate model construction.

III. A ridge informed trust-region method

Our proposed approach seeks to determine the solution to the bound-constrained optimisation problem (2) through a sequence of d-variate optimisation problems of the conditional expectation function $g(\mathbf{y})$ (see (21)), where d is the reduced problem dimension at the current iteration. To be more precise, as d increases, we seek to solve the optimisation problem

$$\begin{aligned} \min_{\mathbf{y}} \quad & g(\mathbf{y}) \\ \text{subject to} \quad & \mathbf{y} \in \mathcal{Y}, \end{aligned} \tag{24}$$

where $\mathbf{y} \in \mathbb{R}^d$, g is as in (21), and $\mathcal{Y} = \{\, \mathbf{y} = \mathbf{W}_1^T\mathbf{x} \mid \mathbf{x} \in \mathcal{X} \,\}$ with $\mathbf{W}_1 \in \mathbb{R}^{n\times d}$. This is achieved by solving a series of trust-region subproblems

$$\begin{aligned} \min_{\mathbf{s}} \quad & m_k(\mathbf{W}_1^T(\mathbf{x}_k + \mathbf{s})) \\ & \|\mathbf{W}_1^T\mathbf{s}\|_2 \le \Delta_k \\ & \mathbf{x}_k + \mathbf{s} \in \mathcal{X} \end{aligned} \tag{25}$$

using the modified ratio

$$r_k = \frac{f(\mathbf{x}_k) - f(\mathbf{x}_k + \mathbf{s}_k)}{m_k(\mathbf{W}_1^T\mathbf{x}_k) - m_k(\mathbf{W}_1^T(\mathbf{x}_k + \mathbf{s}_k))} \tag{26}$$

to determine the approach taken during the next iteration. If $r_k \ge \eta_1$ the iteration is *successful*, and if $\eta_0 \le r_k < \eta_1$ it is *acceptable*. Conversely, if $r_k < \eta_0$ the iteration is *unsuccessful*. The algorithm terminates when a user-defined maximum number of iterations has been reached. Details of the RITR algorithm are presented in Algorithm 2.

III.A. Algorithm initialisation

During initialisation of Algorithm 2, the eigenvectors $\mathbf{W}$ are computed using M DOE samples. The Monte Carlo sampling method (20) requires $M = \alpha k \log(n)$ gradient samples. To keep this number to a minimum, the number of gradient samples taken during this DOE is a user-specified quantity with a default value of $M = 15\lceil \log(n) \rceil$. We note here that this is a particularly low number when estimating the active subspaces. Therefore, the active subspaces obtained may not be as accurate. To mitigate this, we encode a noise-handling capability; see III.C.

III.B. Model construction step

The aim of the model construction step is to ensure the quality of the model m_k. In theory, if the sample set $\mathbf{Y} = \{\, \mathbf{y}^0, \mathbf{y}^1, \ldots, \mathbf{y}^p \,\}$ where $\mathbf{y}^i = \mathbf{W}_1^T\mathbf{x}^i$ is not well-poised within the trust-region $B(\mathbf{W}_1^T\mathbf{x}_k, \Delta_k)$, it should be made so by performing a geometry improving step (Conn et al.,[20] Algorithms 6.2 – 6.6). In practice, we perform a simplified improvement step based on the approach taken by Cartis et al.,[23,39] where only one or two points are altered at each iteration. If the last iteration was acceptable, we simply remove either the point which is furthest from the current iterate $\mathbf{y}_k = \mathbf{W}_1^T\mathbf{x}_k$ or the point which would improve the well-poisedness of $\mathbf{Y}$ the most. If the iteration was unsuccessful, we check if the sample set $\mathbf{Y}$ needs to be *improved* before we reduce the trust-region radius. If for some $\mathbf{y} \in \mathbf{Y}$,

$$\max_{\mathbf{y}} \|\mathbf{y} - \mathbf{y}_k\|_2 > \delta \tag{27}$$

Algorithm 2 Ridge informed trust-region algorithm

```
1: Phase 0: Algorithm initialisation
2:   Let $\mathbf{x}_0$, $0 < \Delta_0 < \Delta_{\max}$, $0 < \gamma_0 < 1 \leq \gamma_1$, and $0 < \eta_0 < \eta_1 < 1$ be given
3:   Calculate $\mathbf{W}$ (16) using $M$ samples and choose a suitable initial $d$
4:   Choose sample points $\mathbf{Y}$ (see Algorithm 3) and construct model $m_0$
5: for $k = 0,\ 1,\ \dots$ do
6:   Phase 1: Step calculation
7:     Compute step $\mathbf{s}_k$ by solving Equation (25)
8:   Phase 2: Trust-region and iterate update
9:     Compute $r_k$ using Equation (26)
10:    if $r_k \geq \eta_1$ then
11:      Set $\Delta_{k+1} \leftarrow \min\{\gamma_1\Delta_k,\ \Delta_{\max}\}$ and $\mathbf{x}_{k+1} \leftarrow \mathbf{x}_k + \mathbf{s}_k$
12:    else if $r_k \geq \eta_0$ then
13:      Set $\Delta_{k+1} \leftarrow \Delta_k$ and $\mathbf{x}_{k+1} \leftarrow \mathbf{x}_k + \mathbf{s}_k$
14:    else if $\mathbf{Y}$ can be improved (see Section III.B) then
15:      Set $\Delta_{k+1} \leftarrow \Delta_k$ and $\mathbf{x}_{k+1} \leftarrow \mathbf{x}_k$
16:    else if Noise auto-detected (see Section III.C) then
17:      Increase $d$ and set $\Delta_{k+1} \leftarrow \Delta_0$ and $\mathbf{x}_{k+1} \leftarrow \mathbf{x}_k$
18:    else
19:      Set $\Delta_{k+1} \leftarrow \gamma_0\Delta_k$ and $\mathbf{x}_{k+1} \leftarrow \mathbf{x}_k$
20:    end if
21:  Phase 3: Model construction step
22:    Choose sample points $\mathbf{Y}$ and construct model $m_{k+1}$
23: end for
24: return $\mathbf{x}_k$
```

where δ is a scalar multiple of Δ_k, we say $\mathbf{Y}$ needs to be improved. Here, we improve $\mathbf{Y}$ using a modified pivotal algorithm proposed by Conn et al.[20] (Algorithm 6.6). This algorithm seeks to use the matrix bound from Equation (13) to improve the well-poisedness of the set $\mathbf{Y}$ using a pivoted LU factorisation of the Vandermonde matrix $\mathbf{M}(\bar{\boldsymbol{\phi}}, \mathbf{Y})$ where

$$\bar{\phi}(\mathbf{y}) = \{\, 1, y_1, \dots, y_d, \frac{1}{2}y_1^2, y_1 y_2, \dots, \frac{1}{2}y_d^2 \,\} \tag{28}$$

and $\mathbf{Y}$ has been scaled and translated such that $\mathbf{Y} \in B(\mathbf{0}, 1)$. Upon termination of this algorithm, there is a set of *pivot polynomials* $\mu = \{\, \mu_0(\mathbf{y}), \dots, \mu_p(\mathbf{y}) \,\}$ and the corresponding Vandermonde matrix $\mathbf{M}(\mu, \mathbf{Y})$ is the upper triangular matrix of the LU factorisation of $\mathbf{M}(\bar{\boldsymbol{\phi}}, \mathbf{Y})$. Moreover,

$$\|\mathbf{M}(\bar{\boldsymbol{\phi}}, \mathbf{Y})^{-1}\| \leq \frac{C\sqrt{p+1}}{\xi} \tag{29}$$

where ξ is the lower bound on the absolute value of the pivot polynomials μ and C is a constant which is dependent on the factorisation. Comparing this bound to the one in Equation (13) demonstrates the importance of maximising the absolute value of the pivot polynomials. That is, by choosing a sufficiently small threshold ξ, one can guarantee the Λ-poisedness of the set $\mathbf{Y}$. Conn et al.[21] (Lemma 2) prove that a value of $\xi = 0.25$ is a suitable choice for quadratic models.

Our modified version of Conn's original algorithm is presented in Algorithm 3. Just as in Conn's algorithm, Algorithm 3 determines the best point for replacement by pivoting rows (i.e. points $\mathbf{y}^i$) which result in a small pivot value to the bottom of $\mathbf{Y}$. When the last point $\mathbf{y}^p$ is reached, the value of its corresponding pivot $|\mu_p(\mathbf{y}^p)|$ is checked against a minimum threshold $\xi = 0.25$. If the pivot is above this threshold, it is accepted; otherwise, it is rejected and a new point is generated. The optimisation problem (30) seeks to determine a new point which has maximal pivot value in both the feasible region (i.e. within zonotope $\mathcal{Y}$) and the trust-region. Note, this optimisation problem does not need to be solved exactly. Rather a solution $\mathbf{y}^p$ which has pivot value $|\mu_p(\mathbf{y}^p)| \geq \xi$ is all that is required. In practice, we use a combination of both stochastic and model-based optimisation methods to find a solution which has pivot value $\xi \geq 0.25$.

Algorithm 3 Incremental improvement of poisedness of $\mathbf{Y}$ via LU factorisation (modified)

```
1: Let $\mathbf{Y}$ be a poised set containing current iterate $\mathbf{y}_k$
2: Initialise $\mu_i(\mathbf{y}) = \bar{\phi}_i(\mathbf{y})$ for $i = 0, \dots, p$ where $\bar{\phi}(\mathbf{y}) = \{ 1, y_1, \dots, y_d, \frac{1}{2}y_1^2, y_1 y_2, \dots, \frac{1}{2}y_d^2 \}$
3: for $i = 0, \dots, p$ do
4:     Find
           $$\mathbf{y}^* = \max_{\mathbf{y} \in \mathbf{Y}} |\mu_i(\mathbf{y})|$$
5:     if $i < p$ then
6:         Swap the positions of $\mathbf{y}^*$ and $\mathbf{y}^i$
7:         for $j = i + 1, \dots, p$ do
8:             $$\mu_j(\mathbf{y}) \leftarrow \mu_j(\mathbf{y}) - \frac{\mu_j(\mathbf{y}^*)}{\mu_i(\mathbf{y}^*)} \mu_i(\mathbf{y})$$
9:         end for
10:    else
11:        if $|\mu_p(\mathbf{y}^*)| < \xi$ then
12:            Set $\mathbf{y}^p$ to be the (approximate) solution to
               $$\begin{aligned} \max_{\mathbf{y}} \quad & |\mu_p(\mathbf{y})| \\ \text{subject to} \quad & \|\mathbf{y} - \mathbf{y}_k\| \leq \Delta_k \\ & \mathbf{y} \in \mathcal{Y}. \end{aligned} \tag{30}$$
13:        end if
14:    end if
15: end for
16: return $\mathbf{Y}$
```

III.C. Noise detection step

Unless f is an exact ridge function, the points $\mathbf{x}$ which map to a given active coordinate $\mathbf{y} = \mathbf{W}_1^T \mathbf{x}$ will exhibit variation in their corresponding function evaluations $f(\mathbf{x})$, as the inverse map $\mathbf{y} \to \mathbf{x}$ is not unique. This variation can be treated as noise in the conditional expectation function $g(\mathbf{W_1}^T \mathbf{x})$ (21). That is, we can consider

$$f(\mathbf{x}) = g(\mathbf{W_1}^T \mathbf{x}) + \epsilon(\mathbf{x}) \tag{31}$$

where $\epsilon(\mathbf{x})$ is an unknown (but deterministic) noise dependent on the active subspace $\mathbf{W_1}$ and $\mathbf{x}$. This makes (24) a noisy optimisation problem. In such cases, as Δ_k gets small, the interpolation points in $\mathbf{Y}$ become densely packed and their corresponding objective values may all be within the noise level. Augustin and Marzouk[25] proved that in order for a trust-region method with noisy objective function values to converge to a first-order optimal point, one must have

$$\epsilon(\mathbf{x}) = \mathcal{O}(\Delta_k^2) \tag{32}$$

at convergence (Theorem 6.1). That is, as the algorithm progresses, the error must asymptotically tend to Δ_k^2. If this is not the case, models m_k constructed from $\mathbf{Y}$ cannot be trusted to give reliable information for the true function f. When this occurs, Δ_k can no longer be used to signify algorithmic convergence and the algorithm may stagnate in a region which is suboptimal. In their Python implementation of BOBYQA, Py-BOBYQA, Cartis et al.[23] propose a mechanism to detect when algorithmic progress has been halted by noise, and when it has, they perform a restart such that Δ_k is increased and points in $\mathbf{Y}$ may be moved. This restart is triggered if, over the last K iterations:

- Δ_k has not been increased (i.e. no successful iterations), and it has been decreased at least twice as many times as it has been kept constant;
- the slope and correlation coefficient of a linear fit through the points $\{ (k, \log \|\nabla m_k - \nabla m_{k-1}\| \}$ exceed given thresholds; and

- the slope and correlation coefficient of a linear fit through the points $\{ (k, \log \|\nabla^2 m_k - \nabla^2 m_{k-1}\|_F \}$ exceed given thresholds.

If the noise is insignificant, one would expect the slope of these lines to be either negative (gradients and/or Hessians consistently decreasing) or to be very small (not increasing significantly). Moreover, if the correlation coefficient is positive, this indicates that many of the last K iterations have shown significant increase.

Algorithm 2 uses the same mechanism for noise detection as above. However, unlike most noisy optimisation problems, the noise in the optimisation problem (24) may be decreased by increasing the dimension of the trust-region. Instead of performing a restart, the dimension is simply increased by including more columns of the eigenvector $\mathbf{W}$, the trust-region radius Δ_k is returned to the initial value Δ_0, and completely new interpolation points $\mathbf{Y}$ are chosen. This step allows the algorithm to progress by increasing the trust-region such that it is no longer within the noise level. Moreover, it effectively decreases the noise level in the optimisation problem (24) by decreasing the unexplained variance. In theory, if the dimension was consistently increased until the error bound in Equation (32) was met, convergence to a first-order optimal point for Algorithm 2 would be guaranteed. In practice, increasing the dimension too drastically leads to problems of dimensionality, so this step can be performed a maximum of five times before the algorithm is terminated.

IV. Computational results

IV.A. Analytical functions

We apply the proposed approach to the following bound-constrained optimisation test problems:

$$\begin{aligned} \min_{\mathbf{x}} \quad & f_1(\mathbf{x}) \\ \text{subject to} \quad & \mathbf{0} \leq \mathbf{x} \leq \mathbf{1} \end{aligned} \tag{33}$$

$$\begin{aligned} \min_{\mathbf{x}} \quad & f_2(\mathbf{x}) \\ \text{subject to} \quad & -\mathbf{1} \leq \mathbf{x} \leq \mathbf{1} \end{aligned} \tag{34}$$

$$\begin{aligned} \min_{\mathbf{x}} \quad & f_3(\mathbf{x}) \\ \text{subject to} \quad & \mathbf{0} \leq \mathbf{x} \leq \mathbf{1} \end{aligned} \tag{35}$$

$$\begin{aligned} \min_{\mathbf{x}} \quad & f_4(\mathbf{x}) \\ \text{subject to} \quad & -\mathbf{1} \leq \mathbf{x} \leq \mathbf{1} \end{aligned} \tag{36}$$

where

$$f_1(\mathbf{x}) = \frac{\cos(\sum_{k=1}^{n} x_k + \pi)}{1 + \sum_{k=1}^{n} x_k} \tag{37}$$

$$f_2(\mathbf{x}) = 100(\sum_{k=2}^{n} x_k - x_1^2)^2 + (x_1 - 1)^2 + 1 \tag{38}$$

$$f_3(\mathbf{x}) = -\frac{x_1}{2} \sin(500 x_1) + 0.001 \sum_{k=2}^{n} k x_k^2 \tag{39}$$

$$f_4(\mathbf{x}) = e^{0.2x_1} + e^{0.2x_2} + 10(x_2 - x_1^2)^2 + (x_1 - 1)^2 + 0.001 \sum_{k=3}^{n} (k x_k - 0.01)^2. \tag{40}$$

The objective functions are all smooth and continuous, so model-based methods are suitable. Moreover, the global optima of all of these problems are known, so we report performance in terms of relative error of the optimisation result compared to the global minimum. If the method has converged to the global optimum, the relative error will be zero. However, if the method converges to a local optimum with value significantly greater than the global optimum, the relative error may be large. Each of these optimisation problems was tested with increasing dimension n. We have chosen values of 20, 40, 60, 80, and 100 for n. Each optimisation run was specified with a maximum of $90\lceil \log(n) \rceil$ function evaluations. The optimisation runs were terminated when either this limit was reached or if they reached a convergence criterion of 10^{-6}. We compare the performance of RITR to COBYLA, BOBYQA, and Nelder-Mead. All optimisation runs for

COBYLA, BOBYQA, and Nelder-Mead were performed using the open source optimisation library NLopt.[40] It is worthwhile to note the RITR algorithm has a number of steps with random elements (e.g. sampling points in full space and determining new interpolation points). To make comparisons between these methods as fair as possible, we have performed 30 optimisation runs for each optimisation problem starting from the same random but feasible initial point for every algorithm for each run. The active subspaces were computed in RITR using (20) with $M = 15\lceil\log(n)\rceil$ gradient evaluations. Box plots demonstrating the performance of these four algorithms for the specified problems are presented in Figures 2 – 5.

As shown in Figures 2 – 5, the RITR algorithm generally outperforms the other three algorithms for these problems. Not only are the mean performances consistently better, but the range of error values obtained are also significantly lower (noting the logarithmic scale used for relative error). This indicates that the algorithm is not only able to find better solutions, but can do so regularly. We attribute this to our algorithm's ability to reduce the effective dimension of the problem, allowing it more readily to explore the dimensions which have the greatest effect on the objective. Moreover, by doing so, it could identify key regions which deserved greater study, and then perform refinement by increasing the effective dimension.

Figure 2 demonstrates the performance of these algorithms for problem (33). Although the objective of this problem admitted an exact 1D ridge function, it also had many local minima with a global minimum found at the boundary of the design space, making it a rather difficult optimisation problem. Nevertheless, we note that RITR was able to identify the global minimum the majority of the time, whereas the other algorithms often got trapped in local minima. This led to large mean relative errors in the results for the other three algorithms. However, RITR was not completely immune to getting trapped in local minima for this problem. In particular, in the case of $n = 20$, RITR was nearly evenly split between a poor local minimum and the global minimum, thus explaining the shape of its box plot. However, the other three algorithms were unable to find the global minimum in any of the optimisation runs, even for relatively small n. In contrast, RITR was able to more efficiently explore the full design space to find globally optimal solutions.

The performance of these algorithms in the case of problem (34) can be seen in Figure 3. The objective of this problem was an exact 2D ridge function with only one minimum value. It is clear from the box plots that RITR was able to find this minimum value nearly every time, often to machine precision, with only very few function evaluations for all the ranges of dimensions. On the other hand, although the other algorithms performed relatively well in the case of $n = 20$ and $n = 40$, it is very clear that their performance began to drop off around $n = 60$. In fact, for $n = 80$ and $n = 100$, the Nelder-Mead algorithm performed particularly poorly, with some results giving nearly 10^4 relative error. Moreover, the performance of the three other algorithms was generally very dependent on the starting point, as they gave a much larger range of error values for the same range of initial points. This demonstrates that in some cases the RITR algorithm is less sensitive to initial points than other similar algorithms.

In Figure 4, we see the performance of these algorithms for problem (35). The objective of this problem admitted an approximate 1D ridge function with many local minima. To obtain the global optimum, more information was needed than was just contained within the active variable. Therefore, this example provides a good illustration of how our algorithm may perform in cases in which the objective is not exactly a ridge and where many possible minimum solutions exist. From the box plots, we notice that the mean result is consistently lower in the case of RITR than the other three algorithms. Although the range may seem significantly greater in the case of RITR, we remark that the results are shown in log scale. Therefore, the range of error values on a linear scale is significantly lower for RITR than the other algorithms. Moreover, the global optimum of this problem proved very difficult to locate, with RITR obtaining a result within 10^{-5} accuracy just once, and the other algorithms failing to do so within the given computational budget for all dimensions.

Figure 5 portrays algorithmic performance of the tested methods for problem (36). The objective for this problem admitted an approximate 2D ridge function with a single minimum. Again, obtaining an exact solution for this problem required more information than was just contained within the active variables. Therefore, to obtain good solutions RITR had to increase its dimension after a number of iterations. Figure 5 again shows RITR outperforming the other algorithms in terms of mean relative error value. Moreover, the range of error values obtained by RITR was significantly lower than in the case of the other three algorithms. In particular, as the dimension increased, the performance of BOBYQA noticeably dropped off, with a great number of runs returning results with error values of unit order. This poor performance can possibly be attributed to the strict computational budget of only $90\lceil\log(n)\rceil$ function evaluations for each algorithm.

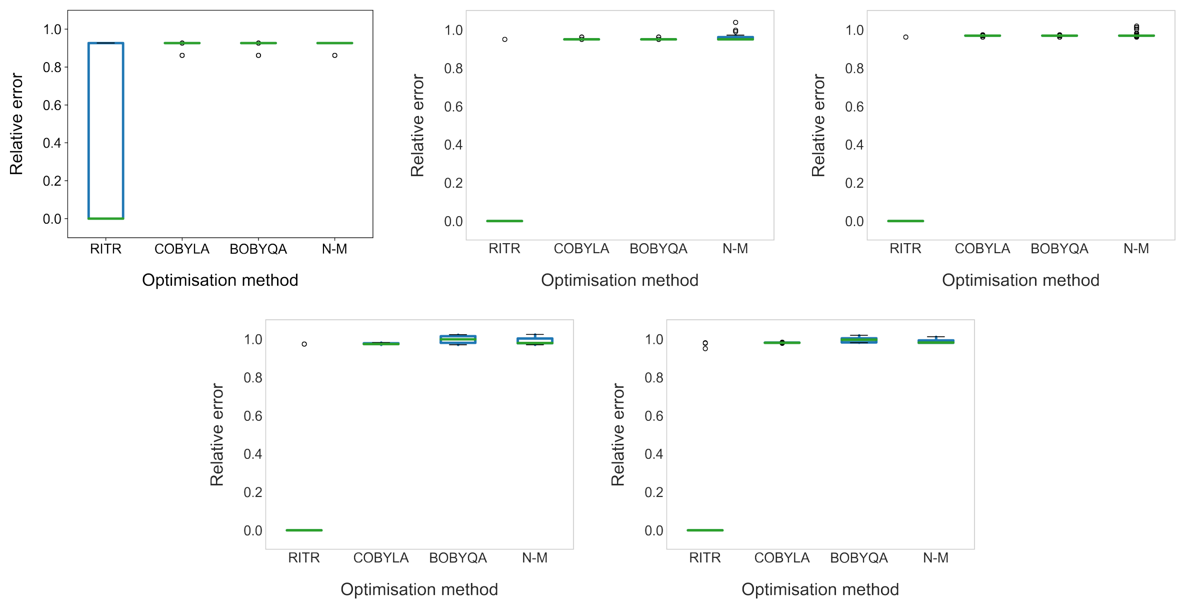


Figure 2: Box plots of optimisation performance for problem (33) with dimensions of 20 (top left), 40 (top centre), 60 (top right), 80 (bottom left), and 100 (bottom right). Each optimisation problem was run 30 times with the 30 different initial points being used by all optimisers. Each run was terminated after $90\lceil\log(n)\rceil$ function evaluations.

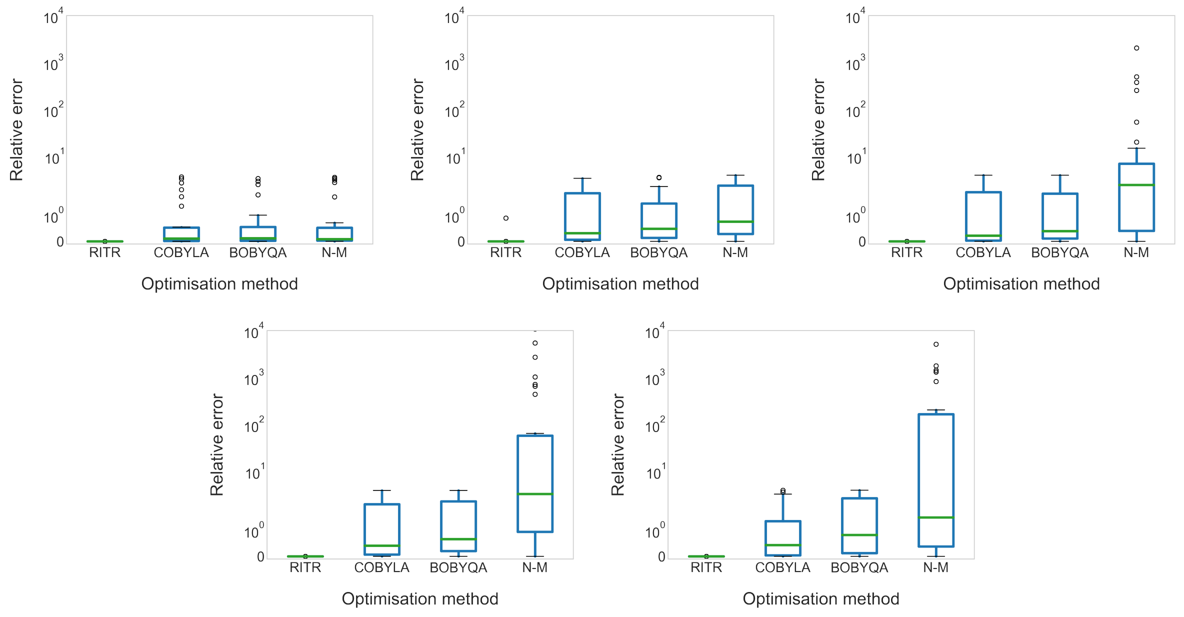


Figure 3: Box plots of optimisation performance for problem (34) with dimensions of 20 (top left), 40 (top centre), 60 (top right), 80 (bottom left), and 100 (bottom right). Each optimisation problem was run 30 times with the 30 different initial points being used by all optimisers. Each run was terminated after $90\lceil\log(n)\rceil$ function evaluations.

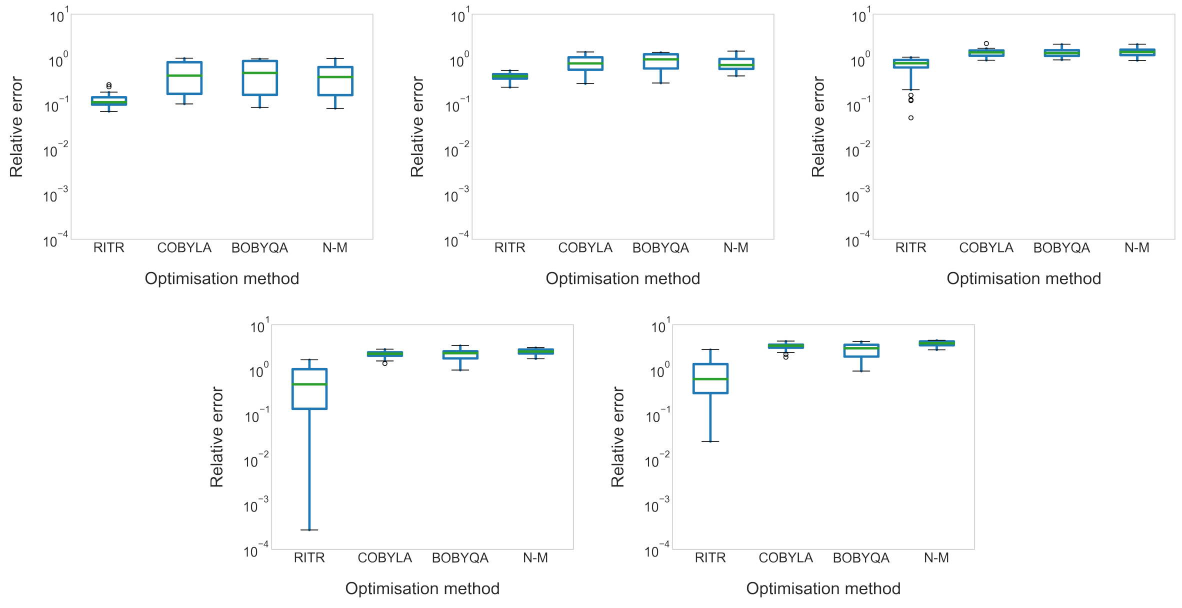


Figure 4: Box plots of optimisation performance for problem (35) with dimensions of 20 (top left), 40 (top centre), 60 (top right), 80 (bottom left), and 100 (bottom right). Each optimisation problem was run 30 times with the 30 different initial points being used by all optimisers. Each run was terminated after $90\lceil\log(n)\rceil$ function evaluations.

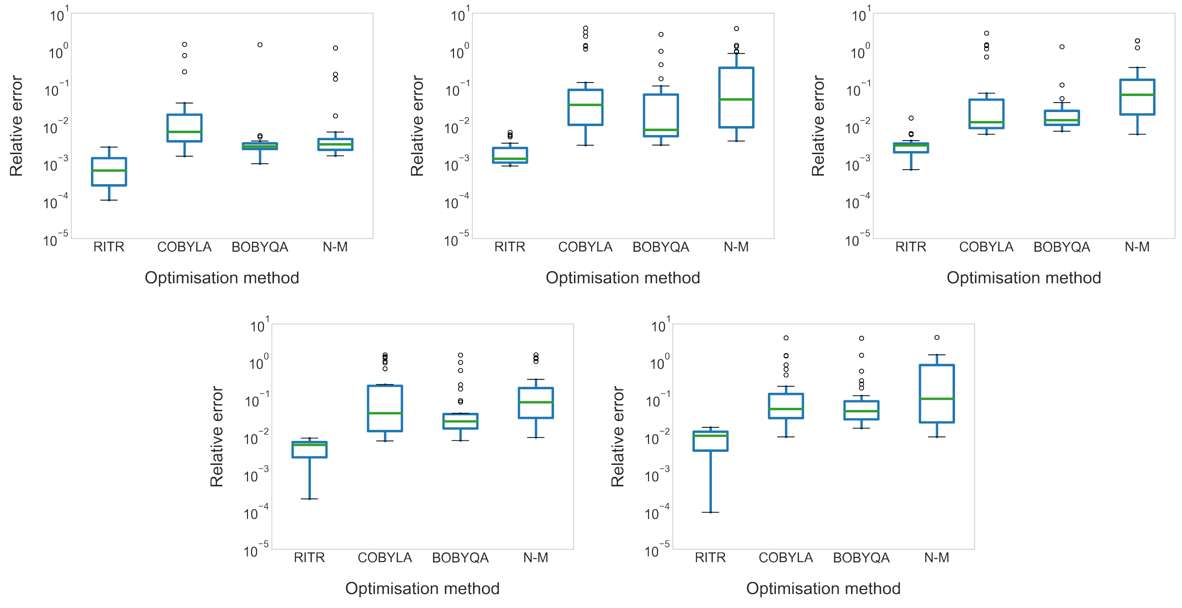


Figure 5: Box plots of optimisation performance for problem (36) with dimensions of 20 (top left), 40 (top centre), 60 (top right), 80 (bottom left), and 100 (bottom right). Each optimisation problem was run 30 times with the 30 different initial points being used by all optimisers. Each run was terminated after $90\lceil\log(n)\rceil$ function evaluations.

We remark that in the results above (and the ones that follow) we have counted the first $15\lceil\log(n)\rceil$ gradient evaluations as function evaluations. In other words, once the active subspace has been computed, RITR is permitted only $75\lceil\log(n)\rceil$ function evaluations compared to $90\lceil\log(n)\rceil$ for the other optimisation strategies. When comparing convergence, accounting for gradient evaluations in this manner is not strictly fair. At the same time, trying to determine how many function evaluations is representative of a gradient evaluation is tedious and would vary based on whether the gradients were obtained via adjoints, automatic differentiation or finite differences. That said, we maintain that the initial overhead in discovering the active subspace reaps benefits within the first few trust-region iterations along the active subspace.

IV.B. NACA 0012 aerofoil

We now apply the RITR algorithm to design optimisation of a NACA 0012 aerofoil. This is formulated as the following optimisation problem:

$$\begin{aligned} \min_{\mathbf{x}} \quad & C_D(\mathbf{x}) \\ \text{subject to} \quad & \mathbf{a} \leq \mathbf{x} \leq \mathbf{b}, \end{aligned} \tag{41}$$

where C_D is the drag coefficient and $\mathbf{x} \in \mathbb{R}^{50}$ are 50 Hicks-Henne bump functions which parametrise the aerofoil geometry (25 parameters each for the lower and upper surfaces). The flight conditions are a Mach number of $M = 0.8$ at an angle of attack of $1.25°$. Evaluations of C_D and its gradient with respect to the design variables were performed using the inviscid Euler solver contained within the SU2 software package.[41] This particular design problem is known to have underlying low-dimensional structure and has been the subject of previous studies in the context of ridge functions.[13,36] To demonstrate the existence of an exploitable low-dimensional structure in this problem, we ran an initial doe of $M = 30$ gradient evaluations to examine the structure. Again, we note here that this number can be seen as rather small. To very accurately capture the low-dimensional structure, a larger doe may be required. However, we will show that it is sufficiently accurate to use within the RITR method. We plot the eigenvalue decay for the drag coefficient in Figure 6. From inspection of this figure, a slight decay between the first and second eigenvalues can be seen, suggesting that a weak 1D active subspace exists for this problem.

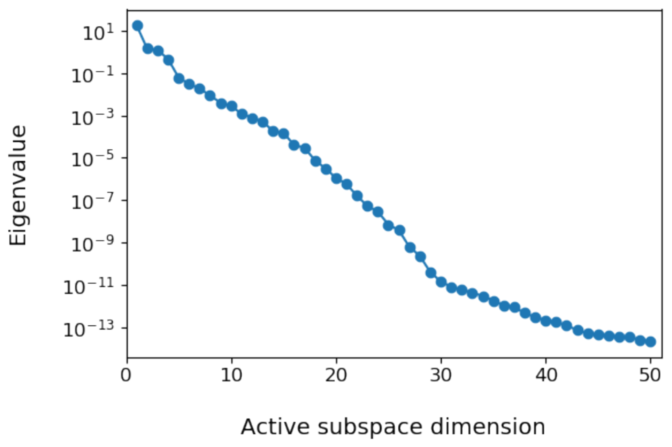
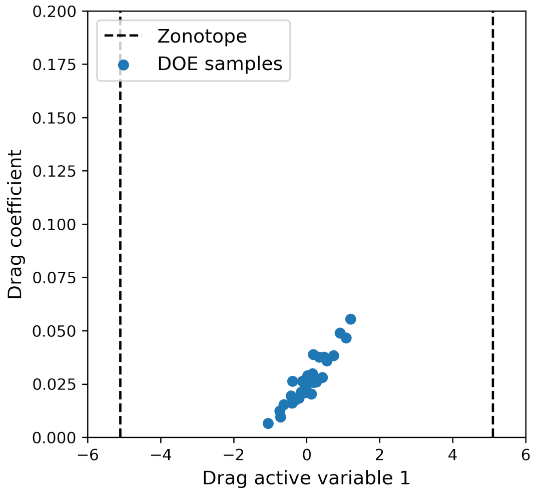
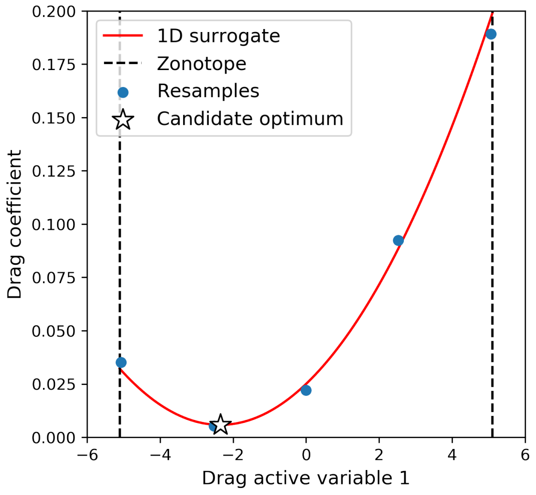


Figure 6: Eigenvalue decay for the drag coefficient for the aerofoil test case

We begin by using the active subspaces obtained from the initial doe in preliminary design optimisation studies. The eigenvalue decay suggests a 1D active subspace for this problem; hence, we investigate the variation of C_D along its first active variable. A 1D sufficient summary plot using these initial 30 doe samples (with gradients) is shown in Figure 7(a). In this figure, the design space has been projected onto the 1D active subspace, and is therefore represented by bounding intervals (1D zonotope) for the 1D active coordinate. Moreover, we see that the doe samples are largely clustered towards the centre of the zonotope. To provide a space-filling sample for surrogate model construction, we resample the design space in the active variable. The resulting C_D values suggest a relationship that is relatively quadratic, so we use a 1D quadratic surrogate model, which can be seen in Figure 7(b). Optimising with this surrogate model suggests a minimum C_D of around 5×10^{-3}.



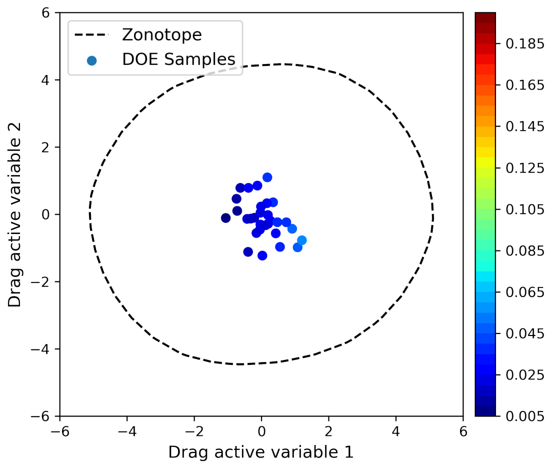
(a) 1D sufficient summary plot



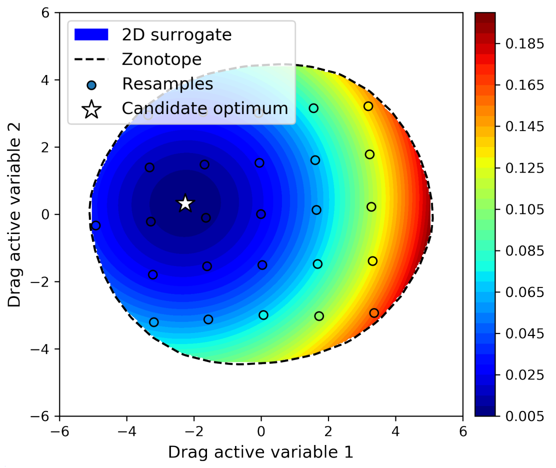
(b) Surrogate model in 1D active subspace

Figure 7: Active subspaces for preliminary design optimisation studies in 1D

A similar examination was performed along both the first and second active variables, using the same doe; these results are shown in Figure 8, where the design space has now been projected onto the 2D active subspace and is now represented as a set of 2D linear inequalities (2D zonotope). Figure 8(a) shows that the doe samples are again largely clustered toward the centre, so we perform a resample to obtain points more evenly spread out across the design space. This results in a relationship which seems quadratic, and so a 2D quadratic is fitted to the data in Figure 8(b). Optimising with this surrogate model again suggests a minimum C_D of around 5×10^{-3}.



(a) 2D sufficient summary plot



(b) Surrogate model in 2D active subspace

Figure 8: Active subspaces for preliminary design optimisation studies in 2D

Such studies can be vital tools for a designer as the global behaviour of the design optimisation problem can be easily visualised and regions of interest can be identified. However, using such studies for design optimisation can result in misleading conclusions. For instance, we suspect the optimal C_D for this problem to be significantly lower than 5×10^{-3}, as we have used an Euler solver for calculations. Due to the lack of a friction term, a C_D much closer to the theoretical ideal of $C_D = 0$ should be possible. Therefore, we use the active subspaces obtained from the doe to initialise our RITR algorithm and seek a more refined result.

For this study, we have performed a single design optimisation run for problem (41) for each of the four tested methods — RITR, COBYLA, BOBYQA, and Nelder-Mead — with a maximum number of function evaluations set to 500. The convergence of these optimisation runs is plotted in Figure 9. It is important to note that in this test case, as before, we acquired the active subspaces using an initial doe of function and gradient evaluations. These function evaluations have been included in the database of solutions which may be used for model construction in the RITR algorithm. Therefore, our algorithm is shown to begin at 30 function evaluations, whereas the other algorithms begin at 0 function evaluations. Nevertheless, RITR is quick to make progress, with a value of $C_D = 6 \times 10^{-4}$ being achieved within 100 function evaluations. In contrast, both COBYLA and BOBYQA make very slow progress until they have sufficiently many function evaluations to construct fully linear models (i.e. 51 for COBYLA and 101 for BOBYQA). Once these model-based algorithms have enough function evaluations, they progress much quicker, with COBYLA reaching a value of $C_D = 8 \times 10^{-4}$ within 100 function evaluations. However, progress for COBYLA from this point onwards is prohibitively slow and it is unable to achieve a solution better than $C_D = 7.03 \times 10^{-4}$. On the other hand, although BOBYQA required more function evaluations before significant progress could be made, it was able eventually to discover a solution much better than COBYLA could. This can be explained by the fact that BOBYQA uses quadratic models; hence, it can better capture the trends in the true function than methods which use linear models. Obtained optimal values from each optimiser are shown in Table 1.

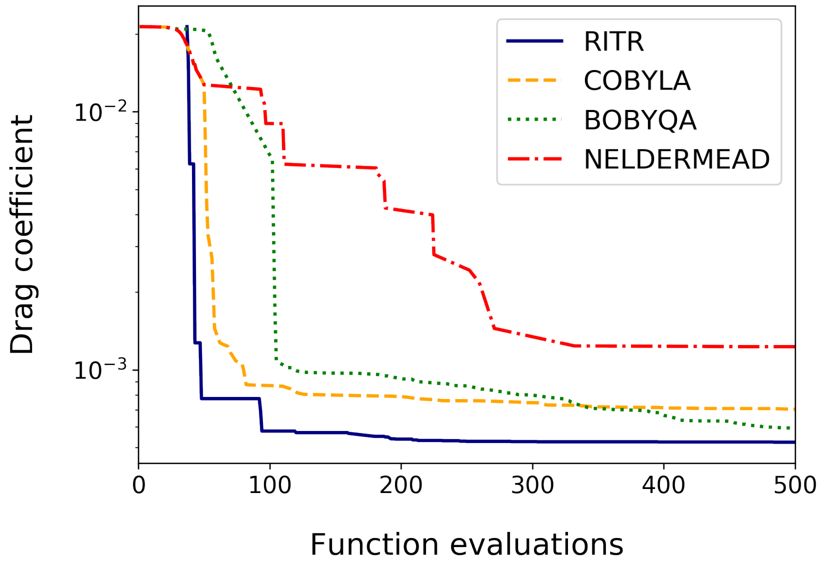
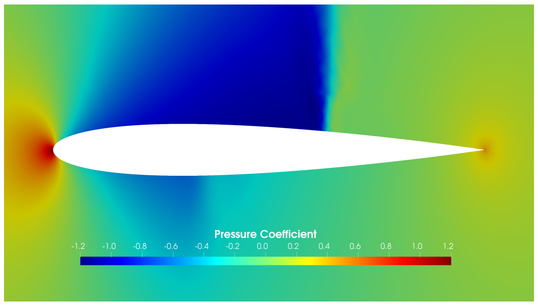


Figure 9: Convergence graph for the aerofoil design optimisation problem (41)

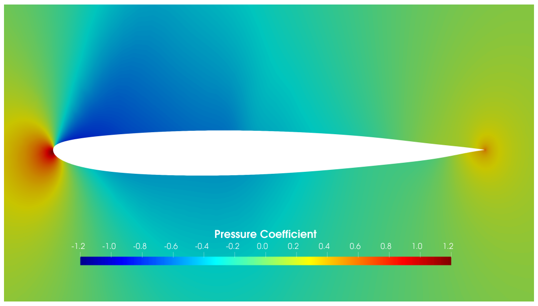
Table 1: Drag coefficient values for the datum design point and for each of the tested algorithms when solving the design optimisation problem (41)

Algorithm	Datum	RITR	COBYLA	BOBYQA	Nelder-Mead
Drag Coefficient	2.13×10^{-2}	5.24×10^{-4}	7.03×10^{-4}	5.94×10^{-4}	1.45×10^{-3}

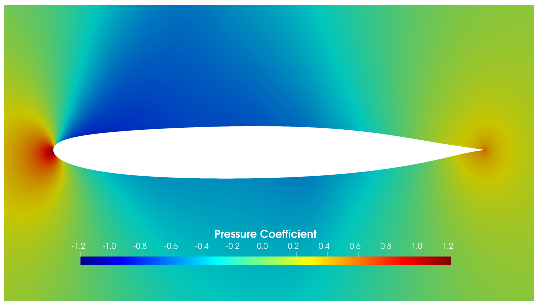
The aerofoils obtained from the optimisation runs for each tested algorithm, as well as the initial datum design (NACA 0012), are presented in Figure 10. The pressure coefficient contours for the NACA 0012 aerofoil (shown in Figure 10(a)) depict a very sharp shock occurring downstream of the mid-chord of the aerofoil. This is a significant source of drag for the aerofoil, and this shock should be removed to greatly reduce the drag. In fact, for all the optimised aerofoils, this sharp shock has been removed and the pressure more evenly distributed along the upper surface of the aerofoil. This effect is most apparent in the case of the aerofoils generated by the RITR and BOBYQA algorithms.



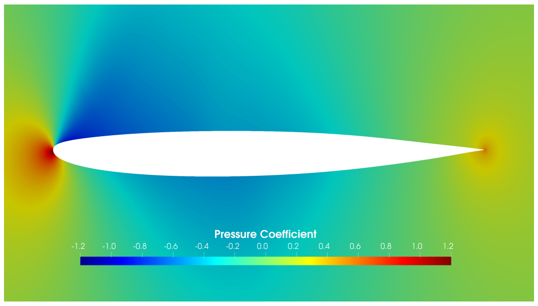
(a) NACA 0012



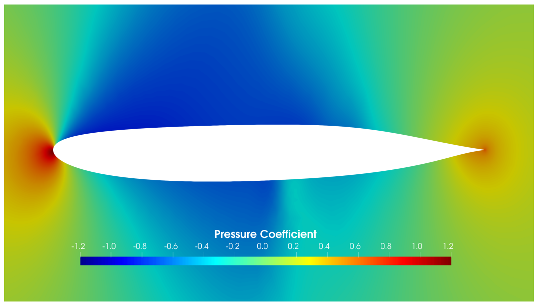
(b) RITR



(c) COBYLA



(d) BOBYQA



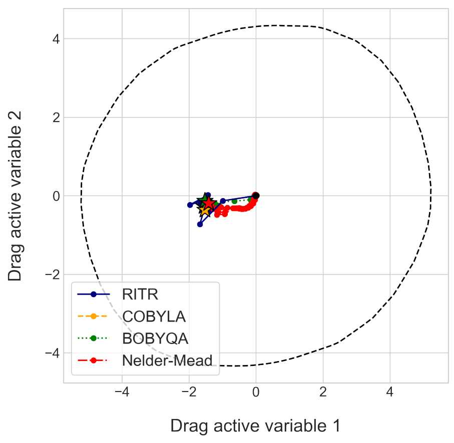
(e) Nelder-Mead

Figure 10: Aerofoils and contours of the corresponding pressure coefficient field for (a) the datum NACA 0012 design and (b) – (e) the optimal designs obtained from each of the optimisation runs

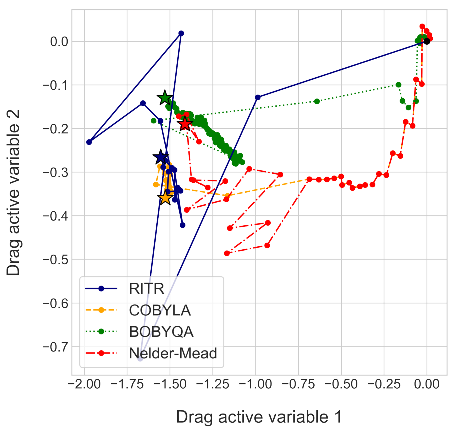
One feature of the RITR algorithm that is worth emphasising is its ability to effectively survey the design space for candidate optimal solutions. In Figure 11, we can see that RITR takes much larger steps in the direction of the active variables (i.e. the directions in which the true function shows greatest variation). This allows RITR to converge very quickly, and results in the large drops in objective values seen in Figure 9. It is interesting to note that, although RITR was able to survey the whole design space, it performed refinement in the same region to which the other algorithms converged. This suggests that, for this problem at least, the existence of only a single region of interest in the design space.

V. Conclusions

In this paper we developed a ridge-informed trust-region (RITR) algorithm and tested its performance on high-dimensional bound-constrained optimisation problems. We demonstrated that the RITR algorithm could efficiently explore the design space by finding and exploiting the inherent low-dimensional structure in these problems. In future work, we will develop the necessary capabilities to handle general nonlinear constraints which also admit low-dimensional structure. This may be performed by an augmented Lagrangian



(a) Complete design space



(b) Area of interest

Figure 11: Paths taken on the design optimisation problem (41) for all the tested optimisation algorithms projected onto the 2D active subspace

approach. This will require identifying and exploiting multiple active subspaces for each qoi. Furthermore, we aim to mitigate the need for initial gradient evaluations for finding the active subspace, to make our algorithm ideally suited for a wide range of industrial and academic applications.

Acknowledgments

The first author was supported by the Weir Advanced Research Centre (WARC) through an EPSRC CASE studentship. The second author was supported by a Rolls-Royce postdoctoral fellowship. The authors would like to thank Chun Yui Wong for his helpful comments and advice. The first author would also like to give special thanks to Dr Hui Feng for her useful suggestions and complete support.

References

[1]Bellman, R., *Dynamic Programming*, Princeton University Press, Princeton, USA, 1st ed., 1957.

[2]Constantine, P. G., del Rosario, Z., and Iaccarino, G., "Many Physical Laws are Ridge Functions," *arXiv preprint*, 2016, pp. 1—20.

[3]Seshadri, P., Shahpar, S., Constantine, P., Parks, G., and Adams, M., "Turbomachinery Active Subspace Performance Maps," *Journal of Turbomachinery*, Vol. 140, No. 4, 2018, pp. 041003–1—41003–11.

[4]Lukaczyk, T. W., Constantine, P., Palacios, F., and Alonso, J. J., "Active Subspaces for Shape Optimization," *10th AIAA Multidisciplinary Design Optimization Specialist Conference*, 2014.

[5]Glaws, A., Constantine, P., N. Shadid, J., and Wildey, T., "Dimension Reduction in MHD Power Generation Models: Dimensional Analysis and Active Subspaces," *Statistical Analysis and Data Mining: The ASA Data Science Journal*, Vol. 10, 2017, pp. 312—325.

[6]Loudon, T. and Pankavich, S., "Mathematical Analysis and Dynamic Active Subspaces for a Long Term Model of HIV," *Mathematical Biosciences and Engineering*, Vol. 14, 2017, pp. 709—733.

[7]Wong, Y. W., Seshadri, P., Parks, G. T., and Girolami, M., "Embedded Ridge Approximations: Constructing Ridge Approximations Over Localized Scalar Fields for Improved Simulation-Centric Dimension Reduction," *arXiv preprint*, 2019.

[8]Wong, Y. W., Seshadri, P., and Parks, G. T., "Extremum Global Sensitivity Analysis with Least Squares Polynomials and their Ridges," *arXiv preprint*, 2019.

[9]Tadeja, S. K., Seshadri, P., and Kristensson, P. O., "Exploring Aerospace Design in Virtual Reality with Dimension Reduction," *Proceedings of AIAA SciTech Forum and Exposition*, San Diego, CA, 2019.

[10]Zhao, M., Alimo, S. R., and Bewley, T. R., "An Active Subspace Method for Accelerating Convergence in Delaunay-based Optimization via Dimension Reduction," *Proceedings of the IEEE Conference on Decision and Control*, Miami Beach, Florida, USA, 2018, pp. 2765—2770.

[11]Li, J., Cai, J., and Qu, K., "Surrogate-based Aerodynamic Shape Optimization with the Active Subspace Method," *Structural and Multidisciplinary Optimization*, Vol. 59, 2019, pp. 403—419.

[12]Pelikan, M., Goldberg, D. E., and Cantú-Paz, E., "BOA: The Bayesian Optimization Algorithm," *Genetic and Evolutionary Computation*, Vol. 1, 1999, pp. 525—532.

[13]Constantine, P. G., *Active Subspaces: Emerging Ideas for Dimension Reduction in Parameter Studies*, SIAM Spotlights, Philadelphia, 2015.

[14]Conn, A. R., Gould, N. I., and Toint, P. L., *Trust-Region Methods*, Society for Industrial and Applied Mathematics : Mathematical Programming Society, Philadelphia, Pa., 2000.

[15]Nocedal, J. and Wright, S., *Numerical Optimization*, Springer, New York, USA, 2nd ed., 2012.

[16]Powell, M. J., "Least Frobenius Norm Updating of Quadratic Models that Satisfy Interpolation Conditions," *Mathematical Programming*, Vol. 100, No. 1, 2004, pp. 183—215.

[17]Conn, A. R., Scheinberg, K., and Vicente, L. N., "Global Convergence of General Derivative-free Trust-Region Algorithms to First- and Second-Order Critical Points," *SIAM Journal on Optimization*, Vol. 20, No. 1, 2009, pp. 387—415.

[18]Wild, S. M. and Shoemaker, C., "Global Convergence of Radial Basis Function Trust-Region Algorithms for Derivative-free Optimization," *SIAM Review*, Vol. 55, No. 2, 2013, pp. 349—371.

[19]Augustin, F. and Marzouk, Y. M., "A Trust-Region Method for Derivative-free Nonlinear Constrained Stochastic Optimization," *arXiv preprint*, 2017, pp. 1—35.

[20]Conn, A. R., Scheinberg, K., and Vicente, L. N., *Introduction to Derivative-free Optimization*, Society for Industrial and Applied Mathematics : Mathematical Programming Society, Philadelphia, Pa., 2009.

[21]Conn, A. R., Scheinberg, K., and Vicente, L. N., "Geometry of Interpolation Sets in Derivative-free Optimization," *Mathematical Programming*, Vol. 111, No. 1-2, 2008, pp. 141—172.

[22]Powell, M., "The BOBYQA Algorithm for Bound Constrained Optimization without Derivatives," Tech. rep., Department of Applied Mathematics and Theoretical Physics, University of Cambridge, Cambridge, 2009.

[23]Cartis, C. and Roberts, L., "A Derivative-free Gauss-Newton Method," *Mathematical Programming Computation*, 2019.

[24]Conn, A., Scheinberg, K., and Toint, P., "A Derivative-free Optimization Algorithm in Practice," *Proceedings of AIAA St Louis Conference*, 1998, pp. 1—11.

[25]Augustin, F. and Marzouk, Y. M., "NOWPAC: A Provably Convergent Derivative-free Nonlinear Optimizer with Path-Augmented Constraints," Tech. rep., MIT, 2014.

[26]Audet, C., Conn, A. R., Le Digabel, S., and Peyrega, M., "A Progressive Barrier Derivative-free Trust-Region Algorithm for Constrained Optimization," *Computational Optimization and Applications*, Vol. 71, No. 2, 2018, pp. 307—329.

[27]Powell, M. J. D., "A Direct Search Optimization Method That Models the Objective and Constraint Functions by Linear Interpolation," *Advances in Optimization and Numerical Analysis*, Springer Netherlands, 1994, pp. 51—67.

[28]Vanden Berghen, F. and Bersini, H., "CONDOR: A New Parallel, Constrained Extension of Powell's UOBYQA Algorithm: Experimental Results and Comparison with the DFO Algorithm," *Journal of Computational and Applied Mathematics*, Vol. 181, No. 1, 2005, pp. 157—175.

[29]Wendor, A. D., Botero, E., and Alonso, J. J., "Comparing Different Off-the-shelf Optimizers' Performance in Conceptual Aircraft Design," *17th AIAA/ISSMO Multidisciplinary Analysis and Optimization Conference*, 2016.

[30]Nelder, J. A. and Mead, R., "A Simplex Method for Function Minimization," *The Computer Journal*, Vol. 7, No. 4, 1965, pp. 308—313.

[31]Lagarias, J. C., Reeds, J. A., Wright, M. H., and Wright, P. E., "Convergence Properties of the Nelder-Mead Simplex Method in Low Dimensions," *SIAM Journal on Optimization*, Vol. 9, No. 1, 1998, pp. 112—147.

[32]McKinnon, K. I., "Convergence of the Nelder-Mead Simplex Method to a Nonstationary Point," *SIAM Journal on Optimization*, Vol. 9, No. 1, 1998, pp. 148—158.

[33]Lewis, R. M., Torczon, V., and Trosset, M. W., "Direct Search Methods: Then and Now," *Journal of Computational and Applied Mathematics*, Vol. 124, No. 1-2, 2000, pp. 191—207.

[34]Han, L. and Neumann, M., "Effect of Dimensionality on the Nelder-Mead Simplex Method," *Optimization Methods and Software*, Vol. 21, No. 1, 2006, pp. 1—16.

[35]Diez, M., Campana, E. F., and Stern, F., "Design-Space Dimensionality Reduction in Shape Optimization by Karhunen-Loève expansion," *Computer Methods in Applied Mechanics and Engineering*, Vol. 283, jan 2015, pp. 1525—1544.

[36]Hokanson, J. M. and Constantine, P. G., "Data-Driven Polynomial Ridge Approximation Using Variable Projection," *SIAM Journal on Scientific Computing*, Vol. 40, No. 3, 2017, pp. 1566–1589.

[37]Seshadri, P., Yuchi, S., and Parks, G. T., "Dimension Reduction via Gaussian Ridge Functions," *arXiv preprint arXiv:1802.00515*, 2018.

[38]Constantine, P. G., Dow, E., and Wang, Q., "Active Subspace Methods in Theory and Practice: Applications to Kriging Surfaces," *SIAM Journal on Scientific Computing*, Vol. 36, No. 4, 2014, pp. A1500—A1524.

[39]Cartis, C., Fiala, J., Marteau, B., and Roberts, L., "Improving the Flexibility and Robustness of Model-based Derivative-free Optimization Solvers," *ACM Transactions on Mathematical Software*, Vol. 45, No. 3, 2019, pp. 1—41.

[40]Johnson, S. G., "The NLopt Nonlinear-Optimization Package," .

[41]Palacios, F., Colonno, M. R., Aranake, A. C., Campos, A., Copeland, S. R., Economon, T. D., Lonkar, A. K., Lukaczyk, T. W., Taylor, T. W., and Alonso, J. J., "Stanford University Unstructured (SU2): An Open-Source Integrated Computational Environment for Multi-Physics Simulation and Design," *51st AIAA Aerospace Sciences Meeting including the New Horizons Forum and Aerospace Exposition 2013*, 2013.