Model-Based Trust Region Methods for Derivative-Free Optimization with Unrelaxable Linear Constraints

Trever Hallock and Stephen C. Billups

December 8, 2021

Abstract

We propose a model-based trust-region algorithm for constrained optimization problems with linear constraints in which derivatives of the objective function are not available and the objective function values outside the feasible region are not available. In each iteration, the objective function is approximated by an interpolation model, which is then minimized over a trust region. To ensure feasibility of all sample points and iterates, we consider two trust region strategies in which the trust regions are contained in the feasible region. Computational results are presented on a suite of test problems.

1 TO DO

1. Shorten literature review
2. Consider removing material on finding the maximum volume ellipsoid
3. Remove ellipsoid searches. Instead, just use the one center defined in the analysis.
4. Remove polyhedral trust region method.

2 Introduction

Derivative-free optimization (DFO) refers to mathematical programs involving functions for which derivative information is not explicitly available. Such problems arise, for example, when the functions are evaluated by simulations or by laboratory experiments. Applications of DFO appear in many fields, including photo-injector optimization [1], circuitry arrangements [2], machine learning [3], volume optimization [4], and reliability-based optimization [5]. In such applications, function evaluations are typically expensive, so it is sensible to invest significant computational resources to minimize the number of function evaluations required.

This work is aimed at developing algorithms to solve constrained optimization problems of the form

$$\begin{array}{ll} \min_{x\in\mathbb{R}^n} & f(x) \\ \text{subject to} & c_i(x) \leq 0, \qquad \forall 1 \leq i \leq m, \end{array} \tag{1}$$

where f and c_i for $1 \leq i \leq m$ are real-valued functions on $\mathbb{R}^n$ with at least one of these functions being a *black-box* function, meaning that derivatives cannot be evaluated directly. We let the feasible set be represented as

$$\mathcal{F} = \{x \in \mathbb{R}^n | c_i(x) \leq 0,\ \forall 1 \leq i \leq m\}. \tag{2}$$

We are interested in developing *model-based*, *trust-region* algorithms for solving these problems. Model-based methods work by constructing model functions to approximate the black box functions at each iteration. The model functions are determined by fitting previously evaluated function values on a set of sample points. In trust-region methods, the model functions are used to define a trust-region subproblem whose solution determines the next iterate. For example, the trust-region subproblem might have the form

$$\begin{array}{ll} \min_{\|s\|\leq\Delta_k} & {m_f}^{(k)}\left(x^{(k)}+s\right) \\ \text{subject to} & m_{c_i}^{(k)}\left(x^{(k)}+s\right) \leq 0 \qquad 1 \leq i \leq m, \\ & \|s\| \leq \Delta_k \end{array}$$

where $x^{(k)}$ is the current iterate, ${m_f}^{(k)}$ is the model function approximating f, and $m_{c_i}^{(k)}$ are the model functions approximating the constraint functions $c_i, \forall 1 \leq i \leq m$, and Δ_k is the radius of the trust-region. The key differences between the trust-region subproblem problem and the original are that all functions are replaced with their model functions, and a trust region constraint ($\|s\| \leq \Delta_k$) has been added.

Conceptually, the model functions are "trusted" only within a distance Δ_k of the current iterate $x^{(k)}$; so the trust-region subproblem restricts the length of step s to be no larger than Δ_k. To ensure that the model functions are good approximations of the true functions over the trust region, the sample points are typically chosen to lie within the trust-region.

We are specifically interested in applications where some of the black box functions cannot be evaluated outside the feasible region. Constraints of this type can arise in several ways, such as when a simulation cannot be carried out on particular inputs. For example, the authors of [6] optimize rapid-cycling synchroton lattice design, but some overlapping synchrotron elements cannot be simulated. Following the taxonomy in [7], such constraints are called *unrelaxable*.

For problems involving unrelaxable constraints, we require all iterates and sample points to be feasible. Notice that this feasibility requirement means that if the algorithm is stopped at any time, it will still produce a feasible output.

An important consideration in fitting the model functions is the "geometry" of the sample set. This will be discussed in more detail in Section 4.5, but the key point is that the relative positions of the sample points within the trust region have a significant effect on the accuracy of the model functions over the trust region. When the geometry of the sample set is poor, it is sometimes necessary to evaluate the functions at new points within the trust region to improve the geometry of the sample set. It is well understood how to do this for unconstrained problems, but for constrained problems our requirement that the sample points must be feasible poses some interesting challenges to maintaining good geometry.

As a first step toward developing algorithms to solve such problems, this paper considers a simplified problem where all of the constraints are linear; namely:

$$\begin{array}{ll} \min_{x \in \mathbb{R}^n} & f(x) \\ & Ax \leq b, \end{array}$$

where A is an $m \times n$ matrix and $b \in \mathbb{R}^m$.

We propose an algorithm for solving this problem whose main idea is to construct a feasible ellipsoid at each iteration that lies within the intersection of the feasible region and the current trust region. We call this ellipsoid the *sample trust region*. To choose well-poised sample points for this ellipsoidal region, we adapt a model improvement algorithm presented in [8] for spherical trust regions and establish error bounds on the accuracy of the model functions over this region.

Our convergence analysis is based on an algorithmic framework presented [9], which describes a class of trust-region algorithms for convex constrained minimization without derivatives. This framework assumes that a model of the objective function can be constructed at each iteration that satisfies certain accuracy assumptions, but does not specify how to construct such a model function. Here, we show that the model functions constructed by our algorithm satisfy these assumptions. Hence, using the results from [9], we show that the criticality measure for the iterates generated by our algorithm converges to zero. In particular, any accumulation point of the iterates satisfies the Karush-Kuhn-Tucker first order optimality conditions.

We then discuss several variants to improve on the algorithm, as well as provide numerical results. We present several variants of how to construct the feasible ellipsoid. We first show how to find the maximum volume ellipsoid contained within a polytope given a fixed center. We then explore several strategies for shifting the center of the ellipsoid.

New paragraph about geometry In unconstrained optimization, it is typical and natural that the sample trust region be centered set at the current iterate. Thus, much of the theory presented in [8] about the geometry of the sample set assumes that the sample points are chosen within a ball centered at the current iterate. However, for constrained problems, this is no longer a useful assumption. In particular, when iterates are close to the boundary of the feasible region, it is advantageous to shift the center of the sample trust region away from the current iterate. Because of this, we rework some of the standard results so that they don't require that one of the sample points be at the center of the sample trust region.

3 Notation

Throughout the thesis, we use the following notational conventions. Any variables that depend on the iteration will be super-scripted by k. For example, the k-th iterate is given by $x^{(k)}$, and the model of the objective function for iteration k is given by ${m_f}^{(k)}$. Subscripts on vectors are used as an index into the vector, while vectors in a sequence of vectors use superscripts: that is, $x_i^{(k)}$ is the i-th component of the k-th vector in the sequence $\{x^{(k)}\}_k$. We use $\text{Proj}_X(x) = \arg\min_{x' \in X} \|x' - x\|$ to denote the projection of x onto a convex set X. For any $m \in \mathbb{N}$, we define $[m] = \{i \in \mathbb{N} | 1 \leq i \leq m\}$.

Matrices. Matrices are denoted with capital letters. The i-th row of the matrix A is denoted A_i. For any index set $S \subseteq \{i \in \mathbb{N} | 1 \leq i \leq m\}$, the $|S| \times n$ matrix formed by only using rows of the $m \times n$ matrix A from the set S is A_S. Let the condition number of a matrix Q be denoted $\kappa(Q)$. We use e_i to denote the i-th unit vector and e to denote the vector of all ones. $B_k(c; \Delta)$ is the ball of radius Δ in the k norm, centered at point c. That is,

$$B_k(c; \Delta) = \left\{ x \in \mathbb{R}^n \,\middle|\, \|x - c\|_k \leq \Delta \right\}. \tag{3}$$

Note that some of the common norms are:

$$\|x\|_\infty = \max_{1 \leq i \leq n} |x_i|, \quad \|x\|_2 = \sqrt{\sum_{i=1}^{n} x_i^2}, \quad \text{and} \quad \|x\|_1 = \sum_{i=1}^{n} |x_i|.$$

When a norm is presented without a subscript, it is assumed to be the 2 norm: $\|x\| = \|x\|_2$ for any $x \in \mathbb{R}^n$. For matrices, we will also use the Frobenius norm $\|\cdot\|_F$ which is $\|A\|_F = \sqrt{\sum_i \sum_j A_{i,j}^2}$.

Sets. The complement of a set S is denoted as $\bar{S}$. Define a point x subtracted from a set S as $S - x = \{s \in \mathbb{R}^n | s - x \in S\}$. Set addition is $X + Y = \{x + y | x \in X, y \in Y\}$.

$\delta_{i,j}$ is the Kronecker delta, $\delta_{i,j} = \begin{cases} 1 & \text{if } i = j \\ 0 & \text{if } i \neq j \end{cases}$. We define the positive part of a real number to be

$$a^+ = \begin{cases} a & \text{if} \quad a \geq 0 \\ 0 & \text{otherwise} \end{cases}.$$

4 Background

4.1 Literature Review

Several books and survey articles provide good introductions to the field of derivative-free optimization. Within [8], derivative-free methods are developed in detail. This is the first textbook devoted to derivative-free optimization. It contains an explanation of how to ensure good sample set geometry and introduces the concept of poisedness for unconstrained problems, and it also covers other direct-search and line-search methods. A review of derivative-free algorithms and software libraries can be found in [10]. This compares

several software libraries and reviews the development of derivative-free optimization since it started. Other recent reviews can be found in [11] and [12].

Most algorithms for derivative-free optimization fall into two main categories: direct-search methods and model-based methods. Direct-search methods use comparatively simple strategies to explore the solution space, using only the relative ranks of function values rather than their numerical values. Early direct search methods for unconstrained optimization include coordinate descent [13], the Nelder-Mead algorithm [14], and pattern search methods [15, 16]. Generalizations of pattern-search methods include the GPS method [17, 18] and the mesh adaptive direct search (MADS) algorithm [19, 20].

Model-based methods guide the search using surrogate models of the black-box functions, which are constructed by fitting function values on a set of sample points. Among the more commonly used model functions are linear and quadratic interpolation models [21–23] and radial-basis function interpolation models [24–26].

The use of model functions allows curvature information to be incorporated into the search. As such, model-based methods tend to be more efficient than direct-search methods when the black box functions are smooth. In contrast, direct-search methods can be better suited for handling nonsmooth or noisy functions. They are also more straightforward to implement, and are easier to parallelize.

Hybrid methods, which incorporate ideas from both direct-search and model-based methods, have also been proposed. Some examples are described in [27], [28], and [29].

Constrained derivative-free algorithms. To discuss methods for constrained derivative-free optimization, we follow the constraint taxonomy of Le Digabel and Wild [7] and distinguish between whether constraints are *relaxable* or *unrelaxable*. An unrelaxable constraint is one that *must* be satisfied in order to obtain meaningful information about the objective function and/or constraint functions. Thus algorithms for unrelaxable constraints can use function evaluations only for feasible points. We also distinguish between whether constraints are *algebraic* or *black-box*. A constraint is algebraic if the constraint functions can be computed algebraically, whereas a *black-box constraint* can only be evaluated by running simulation software.

Relaxable constraints. The easiest class of constraints to deal with are the relaxable algebraic constraints. In this case, ideas from classical nonlinear programming methods can easily be adapted to the derivative-free setting. Some examples include penalty methods [30–32], and filter methods [33, 34]. Both penalty methods and filter methods allow iterates to violate constraints, requiring feasibility only in the limit. As such, they are viable approaches only when the constraints are relaxable.

For relaxable black-box constraints, several methods have been proposed that allow the black-box functions to be evaluated at infeasible points. These include penalty methods [35–40], filter methods [41–43], and funnel methods [44, 45].

Another approach is to construct algebraic models of the constraint functions and then require iterates to be feasible with respect to these modelled constraints. In [46], linear models of nearly active constraint functions are constructed and the iterates are accepted only if they are feasible with respect to these modeled constraints. A similar strategy is employed in the COBYLA method of Powell [47], which builds linear models of the objective and constraint functions based on a common set of sample points. A variant of the MADS algorithm is proposed in [48] which uses linear regression models of the constraint functions to guide the choice of search and poll steps. In [49], the classic sequential quadratic programming algorithm is implemented for equality-based constraints.

Unrelaxable algebraic constraints. An algebraic constraint function can always be evaluated, so the only way it is considered unrelaxable is if a black-box function (either the objective or some other constraint function) cannot be evaluated when the constraint is violated. Note that it is always possible to determine whether a point satisfies an algebraic constraint prior to attempting to evaluate any black-box functions. As such, it is relatively straightforward to modify direct-search methods to guarantee that only feasible points are considered. Many direct-search methods have been proposed that take this approach [30, 31, 50–62]

Developing model-based algorithms for unrelaxable constraints is complicated by the fact that the choice of sample points impacts the accuracy of the model functions. Thus, when restrictions on the choice of sample points are imposed, it can be difficult to ensure that the model functions are sufficiently accurate to guarantee convergence to a stationary point. (see Section 9 for a more detailed explanation of this difficulty). In the case of unrelaxable bound constraints, the restrictions on the sample points are not too difficult to mitigate. The BOBYQA algorithm [63] ensures that all points at which the objective function are evaluated satisfy the bound constraints, while still maintaining sufficient model accuracy to guarantee convergence. In [64], an active set method is used when solving the bound-constrained trust-region subproblems. In [65], Wild proposed a radial basis function method for bound constraints, which enforces the bounds when selecting sample points in the model improvement algorithm and when solving the trust region subproblems. In [66], Gratton et al. present a method which restricts the construction of the model functions to subspaces defined by nearly active bound constraints.

There has also been some progress in developing model-based methods for unrelaxable linear constraints. In [67], Gumma et al. propose the LCOBYQA algorithm which is an extension of Powell's NEWUOA algorithm that enforces the linear constraints both when solving the trust region subproblems and when choosing sample points for constructing the model functions. However, no convergence analysis is provided for the method.

Unrelaxable black-box constraints. We now turn our attention to the hardest case–that of unrelaxable black-box constraints. In this case, it is not possible to guarantee that black-box function evaluations will never be attempted at infeasible points. However, it is desireable to minimize the occurrence of such infeasible attempts.

There are several strategies for avoiding infeasible evaluations. The authors of [68] use a reformulation strategy by moving the constraints into the objective. Their work relies on a projection operator to avoid infeasible evaluations and handles non-smooth convex constraints. The authors of [69] use a path-augmentation strategy to ensure trial points are feasible. This is done with a local convexification of the constraints that parameterize a buffer of the constraints. The authors of [70] apply DFO to optimize the Fayans energy density functional, avoiding possible infeasible evaluations by altering the objective function to include a projection onto an L_1 ball. In [71], the authors develop a model-based trust region algorithm that uses an envelope to avoid infeasible trial point evaluations. The algorithm presented within [72] is also a model-based trust region method, and it ensures each iterate is feasible, although sample points may not be. More recently, [73] uses an interior point algorithm to solve derivative-free problems with unrelaxable, black-box constraints.

Of particular importance for our work is [9]. This reference provides an elegant convergence proof for a class of algorithms when the constraints are convex. Our analysis implements abstractions made in this paper and shows the implementation satisfies their requirements.

4.2 Model-Based Trust-Region Methods for DFO

The main idea of model-based, trust-region methods is that trial points are selected at each iteration by solving a trust-region subproblem. Each subproblem has the form

$$\begin{array}{ll} \min_{s\in\mathbb{R}^n} & m_f^{(k)}\left(x^{(k)}+s\right) \\ \text{such that} & x^{(k)}+s\in\mathcal{F}_m^{(k)} \\ & \|s\|\leq\Delta_k \end{array}$$

where $m_f^{(k)}$ is a model function approximating the objective f, Δ_k is the trust region radius, and $\mathcal{F}_m^{(k)}$ is an approximation of the feasible region. The solution $s^{(k)}$ of the trust-region subproblem determines a *trial point* $x^{(k)}+s^{(k)}$. The objective function and constraints are then evaluated at the trial point to determine whether to accept the trial point. If the trial point is rejected, the trust region radius is decreased and a new trial point is computed by solving a revised trust-region subproblem. If the trial point is accepted, then the trust region radius may be increased or decreased for the next iteration depending on how well the sample point improved upon the previous iterate. This will be discussed in Section 4.3.

4.3 Assessing Model Accuracy and Trust Region Radius Management

In trust region methods, each iteration that evaluates a trial point must also test the accuracy of the model functions. To test the accuracy, we calculate a quantity

$$\rho_k=\frac{f(x^{(k)})-f(x^{(k)}+s^{(k)})}{m_f{}^{(k)}(x^{(k)})-m_f{}^{(k)}(x^{(k)}+s^{(k)})}, \tag{4}$$

which measures the actual improvement of the trial point $x^{(k)}+s^{(k)}$ divided by the predicted improvement. If ρ_k is negative, the trial point is rejected and the trust region radius is decreased. On the other hand, if the trial point is accepted, the trust region radius for the next iteration may still be decreased since a small value of ρ_k implies that the model functions are not sufficiently accurate. For larger values of ρ_k the trial point is accepted and the trust region radius may be increased. This strategy has been widely used within trust region frameworks such as [74] and within a derivative-free context [8].

To implement the above strategy, we define parameters $0<\gamma_{\text{min}}<\gamma_{\text{suff}}\leq 1$ as thresholds on ρ_k and $0<\omega_{\text{dec}}<1\leq\omega_{\text{inc}}$ as decrement and increment factors to determine the trust region update policy. That is, when $\rho_k<\gamma_{\text{min}}$, the trust region is decreased by a factor of ω_{dec}, and the trust region is increased by a factor of ω_{inc} during some iterations in which $\rho_k>\gamma_{\text{suff}}$.

4.4 Interpolation-Based Models

Model-based methods for derivative-free optimization construct models of a function $f(x)$ from a family of functions spanned by a set of $p+1\in\mathbb{N}$ basis functions $\Phi=\{\phi_0,\phi_1,\ldots,\phi_p\}$. Each member of this family has the form $m_f(x)=\sum_{i=0}^{p}\alpha_i\phi_i(x)$ for some scalar coefficients $\alpha_i, i\in\{0,\ldots,p\}$.

We use interpolation to choose the coefficients $\alpha=[\alpha_0,\ldots,\alpha_p]^T$ so that $m_f{}^{(k)}$ agrees with f on a set of $p+1$ sample points $Y=\{y^{(0)},y^{(1)},\ldots,y^{(p)}\}$ at which the function f has been evaluated. Thus, the coefficients α must satisfy the *interpolation condition*

$$m_f(y^{(i)})=\sum_{j=0}^{p}\alpha_j\phi_j(y^{(i)})=f(y^{(i)})\quad\forall\quad 0\leq i\leq p. \tag{5}$$

This equation can be written more compactly in the form

$$V\alpha = \bar{f}, \tag{6}$$

where $\bar{f} = [f(y^{(0)}), f(y^{(1)}), \ldots, f(y^{(p)})]^T$ and the Vandermonde matrix V is defined by

$$V = M(\Phi, Y) := \begin{bmatrix} \phi_0(y^{(0)}) & \phi_1(y^{(0)}) & \ldots & \phi_p(y^{(0)}) \\ \phi_0(y^{(1)}) & \phi_1(y^{(1)}) & \ldots & \phi_p(y^{(1)}) \\ & & \vdots & \\ \phi_0(y^{(p)}) & \phi_1(y^{(p)}) & \ldots & \phi_p(y^{(p)}) \end{bmatrix}. \tag{7}$$

The interpolation equation (6) has a unique solution if and only if V is non-singular. In this case, we say that the sample set Y is *poised* for interpolation with respect to Φ. However, even when V is non-singular but "close" to singular, as measured by its condition number, the model's approximation may become inaccurate.

4.5 Sample Set Geometry

The term *geometry* describes how the distribution of points in the sample set Y affects the accuracy of the model function. In the case of polynomial model functions, a rigorous analysis of model accuracy can be performed using *Lagrange polynomials*. Let the space of polynomials on $\mathbb{R}^n$ with degree less than or equal to d be denoted $\mathcal{P}_n^d$ and have dimension $p+1$. The Lagrange polynomials $l_0, l_1, \ldots, l_p$ for the sample set Y are a basis of $\mathcal{P}_n^d$ such that

$$l_i(y^{(j)}) = \delta_{i,j} \tag{8}$$

where $\delta_{i,j} = \{0 \text{ if } i \neq j,\ 1 \text{ if } i = j\}$ is the Kronecker-delta function. Thus, as shown in [8], we can conveniently write

$$m_f(x) = \sum_{j=0}^{p} f(y^{(i)}) l_i(x). \tag{9}$$

We say that a set Y is Λ-*poised* for a fixed constant Λ with respect to a basis Φ on the set $B \subset \mathbb{R}^n$ if and only if the Lagrange polynomials l_i associated with Y satisfy

$$\Lambda \geq \max_{0 \leq i \leq p} \max_{x \in B} |l_i(x)|. \tag{10}$$

In the case of interpolation over the quadratic polynomials, $\mathcal{P}_n^2$, we say that Y is Λ-*poised for quadratic interpolation*.

As shown in [**?**,**?**, Lemmas 3.8 and 3.9] the poisedness constant Λ in the above definition does not depend on the scaling of the sample set and it is invariant with respect to a shift of coordinates. Thus, for any vector $\bar{y} \in \mathbb{R}^n$ and positive scalar $\Delta > 0$, a sample set $Y = \left\{y^{(0)}, \ldots, y^{(p)}\right\}$ is Λ-poised on a set B if and only if the shifted and scaled sample set $\hat{Y} := \frac{1}{\Delta}(Y - \bar{y})$ is Λ-poised on set $\hat{B} = \frac{1}{\Delta}(B - \bar{y})$.

The analysis presented in [8] constructs $\hat{Y}$ by choosing $\bar{y} = y^{(0)}$ and $\Delta = \max_{1 \leq i \leq p} \|y^{(i)} - y^{(0)}\|$. The resulting sample set $\hat{Y}$ is contained in the ball of radius one centered at $\hat{y}^{(0)} = 0$ and has at least one point on the boundary of the ball. The following result establishes a relationship between the Λ-poisedness of $\hat{Y}$ on the ball $B_2(0;1)$ (hence also of Y on $B_2(y^{(0)};\Delta)$ and the condition number of the Vandermonde matrix $\hat{M} = M(\bar{\Phi}, \hat{Y})$, where $\bar{\Phi}$ denotes the monomial basis for $\mathcal{P}_n^2$. Specifically,

$$\bar{\Phi} = \{\bar{\phi}_0, \ldots, \bar{\phi}_p\} = \{1, x_1, \ldots, x_n, x_1^2/2, \ldots x_n^2/2, x_1x_2, \ldots, x_{n-1}x_n\}. \tag{11}$$

Theorem 4.1 *([8, Theorem 3.14]) Let $\hat{M} = M(\bar{\Phi}, \hat{Y})$, and $p+1 = \frac{(n+1)(n+2)}{2}$. If $\hat{M}$ is non-singular and $\left\|\hat{M}^{-1}\right\|_2 \leq \Lambda$, then the set $\hat{Y}$ is $\Lambda\sqrt{p+1}$-poised for quadratic interpolation on the unit ball $B_2(0;1)$. Conversely, if the set $\hat{Y}$ is Λ-poised for quadratic interpolation on the unit ball, then $\left\|\hat{M}^{-1}\right\|_2 \leq \theta\Lambda\sqrt{p+1}$, where $\theta > 0$ is a constant dependent on n but independent of $\hat{Y}$ and Λ.*

This approach is sensible for unconstrained optimization, where we typically choose $y^{(0)}$ to be the current iterate and we are interested in constructing sample sets that are well-poised over the trust region $B_2(y^{(0)};\Delta)$. However, for constrained optimization, it is disadvantageous to require the current iterate to be at the center of our sample trust region. We therefore consider a different transformation by specifying an arbitrary center $\bar{y}$ and radius Δ. We have the following results:

Theorem 4.2 *Let $\Delta > 0$, Y be a sample set with $Y \subset B_2(0;\Delta)$ and let $M = M(\bar{\Phi}, Y)$ where $\bar{\Phi}$ is the monomial basis* (11)*. If M is non-singular, and $\left\|M^{-1}\right\|_2 \leq \Lambda$, then Y is $\hat{\Lambda}$-poised in $B_2(0;\Delta)$, where $\hat{\Lambda} = \Lambda\,(p+1)^{-\frac{1}{2}} \max\left\{1, \Delta, \frac{1}{2}\Delta^2\right\}$.*

Proof:

Let $x \in B_2(0;\Delta)$ be arbitrary and let $\ell(x) = (\ell_0(x), \ldots, \ell_p(x))^T$. As shown in [8], we can write $\ell(x) = M^{-T}\bar{\phi}(x)$, where $\bar{\phi}(x) = (\bar{\phi}_0(x), \ldots, \bar{\phi}_p(x))$. Thus,

$$\begin{aligned}
\|\ell(x)\|_\infty &\leq \left\|M^{-T}\right\|_\infty \left\|\bar{\phi(x)}\right\|_\infty \\
&\leq (p+1)^{-\frac{1}{2}} \left\|M^{-1}\right\|_2 \left\|\bar{\phi}(x)\right\|_\infty \\
&\leq \Lambda\,(p+1)^{-\frac{1}{2}} \max\left\{1, |x_1|, \ldots, |x_n|, \frac{1}{2}x_1^2, \ldots, \frac{1}{2}x_n^2, x_1x_2, \ldots, x_{n-1}x_n\right\} \\
&\leq \Lambda\,(p+1)^{-\frac{1}{2}} \max\left\{1, \Delta, \frac{1}{2}\Delta^2\right\} = \hat{\Lambda}.
\end{aligned}$$

□

To show the converse, we use the two auxilary results are found in [8, Lemma 3.10] and [8, Lemma 3.13].

Lemma 4.3 *Let $\bar{\Phi}$ be the monomial basis* (11)*. There exists a number $\sigma_\infty > 0$ such that, for any choice of v satisfying $\|v\|_\infty = 1$, there exists a $y \in B_2(0;1)$ such that $\left|v^T\bar{\Phi}(y)\right| \geq \sigma_\infty$.*

Lemma 4.4 *Let w be a normalized right-singular vector of a nonsingular $n \times n$ matrix A corresponding to its largest singular value. Then, for any vector $r \in \mathbb{R}^n$,*

$$\|Ar\| \geq |w^T r|\|A\|.$$

Lemma 4.5 *Let $\bar{\Phi}$ be the monomial basis* (11)*. Suppose that a set $Y \subseteq B_2(0;\Delta)$ of $p+1 = \frac{(n+1)(n+2)}{2}$ points is Λ poised over the ball $B_2(0;\Delta)$. Let $\hat{Y} = \frac{1}{\Delta}Y$ be the scaled sample set. Then there exists a constant κ_Λ depending only on p such that $M\left(\bar{\Phi}, \hat{Y}\right)$ as defined in* (7)*, satisfies $\left\|M\left(\bar{\Phi}, \hat{Y}\right)^{-1}\right\| \leq \kappa_\Lambda\Lambda$.*

Proof:

Because Λ-poisedness is scale invariant (this is an immediate consequence of [8, Lemma 3.8]), $\hat{Y}$ is Λ-poised on $B(0,1)$. Let v be a right singular vector of $M^{-1}\left(\bar{\Phi}, Y\right)$ corresponding to its largest singular value normalized so that $\|v\|_\infty = 1$. We know from Lemma 4.3 that there exists $y \in B_2(0;1)$ such that $\left|v^T \bar{\Phi}(y)\right| \geq \sigma_\infty$. Then, because $\hat{Y}$ is Λ-poised on $B_2(0;1)$, we have that

$$\Lambda \geq \|l(y)\|_\infty = \left\| M\left(\bar{\Phi}, Y\right)^{-T} \bar{\Phi}(y) \right\|_\infty \geq (p+1)^{-\frac{1}{2}} \left\| M\left(\bar{\Phi}, \hat{Y}\right)^{-T} \bar{\Phi}(y) \right\|.$$

By applying Lemma 4.4 with $A \leftarrow \bar{M}\left(\bar{\Phi}, \hat{Y}\right)^{-T}$, $w \leftarrow v$, and $r \leftarrow \bar{\Phi}(y)$, we have

$$\begin{aligned}\Lambda \geq (p+1)^{-\frac{1}{2}} \left\| M\left(\bar{\Phi}, \hat{Y}\right)^{-T} \bar{\Phi}(y) \right\| &\geq (p+1)^{-\frac{1}{2}} \left|v^T \bar{\Phi}(y)\right| \left\| M\left(\bar{\Phi}, Y\right)^{-T} \right\| \\ &\geq (p+1)^{-\frac{1}{2}} \sigma_\infty \left\| M\left(\bar{\Phi}, Y\right)^{-T} \right\|.\end{aligned}$$

The conclusion follows with $\kappa_\Lambda = \frac{\sqrt{p+1}}{\sigma_\infty}$.
□

Another limitation of Theorem 4.1 is its requirement to have a point evaluated at the origin. We remove this assumption in Theorem 4.7 by performing the same analysis as in [8, Theorem 3.16]. First, we state the following lemma, found in [75, Lemma 4.1.14].

Theorem 4.6 *Let $f : \mathbb{R}^n \to \mathbb{R}$ be twice continuously differentiable in an open convex set $D \subset \mathbb{R}^n$, and let $\nabla^2 f(x)$ be Lipschitz continuous in D with constant γ. Then, for any $x + p \in D$,*

$$\left| f(x+p) - \left(f(x) + \nabla f(x)^T p + \frac{1}{2} p^T \nabla^2 f(x) p \right) \right| \leq \frac{\gamma}{6} \|p\|^3.$$

Theorem 4.7 *Suppose that f is twice continuously differentiable, and has a Lipschitz continuous Hessian with constant L_{∇^2}. For some $\Delta > 0$, let $Y = \left\{y^{(0)}, y^{(1)}, \ldots y^{(p)}\right\}$ be a set of $p + 1 = \frac{(n+1)(n+2)}{2}$ points contained in $B_2(0;\Delta)$. For some $c \in \mathbb{R}, g \in \mathbb{R}^n, H \in \mathbb{R}^{n \times n}$, consider a quadratic model $m(y) = c + g^T y + \frac{1}{2} y^T H y$ satisfying*

$$f\left(y^{(i)}\right) = m\left(y^{(i)}\right) \quad \forall 0 \leq i \leq p. \tag{12}$$

Let $M(\bar{\Phi}, Y)$ be constructed as in (7) *and $\bar{\Phi}$ is the monomial basis* (11)*. Suppose that $M(\bar{\Phi}, Y)$ is non-singular, and define the scaled matrix*

$$\hat{M}\left(\bar{\Phi}, Y\right) = M\left(\bar{\Phi}, \frac{1}{\Delta} Y\right) = M\left(\bar{\Phi}, Y\right) \begin{pmatrix} 1 & 0 & 0 \\ 0 & \Delta^{-1} I_n & 0 \\ 0 & 0 & \Delta^{-2} I_{n + \frac{n(n-1)}{2}} \end{pmatrix}. \tag{13}$$

Then, there exist constants $\kappa_f, \kappa_g, \kappa_h > 0$ that depend only on p and L_{∇^2} such that for any $y \in B_2(0;\Delta)$:

$$|m(y) - f(y)| \leq \kappa_f \left\| \hat{M}^{-1}(\bar{\Phi}, Y) \right\|_2 \Delta^3, \tag{14}$$

$$\|\nabla m(y) - \nabla f(y)\| \leq \kappa_g \left\| \hat{M}^{-1}(\bar{\Phi}, Y) \right\|_2 \Delta^2, \tag{15}$$

$$\left\|\nabla^2 m(y) - \nabla^2 f(y)\right\| \leq \kappa_h \left\| \hat{M}^{-1}(\bar{\Phi}, Y) \right\|_2 \Delta. \tag{16}$$

Proof:

Let $y \in B_2(0;\Delta)$. Let us define error functions e_f, e_g and e_h so that

$$\begin{aligned} m(y) &= c + g^T y + \frac{1}{2} y^T H y = f(y) + e_f(y), \\ \nabla m(y) &= g + Hy = \nabla f(y) + e_g(y), \\ \nabla^2 m(y) &= H = \nabla^2 f(y) + e_h(y). \end{aligned} \tag{17}$$

For a fixed $0 \leq i \leq p$, we can subtract (17) from (12) to find that

$$\begin{aligned} f\left(y^{(i)}\right) - f(y) - e_f(y) &= m\left(y^{(i)}\right) - m(y) \\ &= \left(y^{(i)} - y\right)^T g + \frac{1}{2}\left(y^{(i)}\right)^T H \left(y^{(i)}\right) - \frac{1}{2} y^T H y \\ = \left(y^{(i)} - y\right)^T g + \frac{1}{2}\left(y^{(i)} - y\right)^T H \left(y^{(i)} - y\right) &+ \left(y^{(i)} - y\right)^T H y. \end{aligned} \tag{18}$$

By defining

$$e_O^{(i)}(y) = f\left(y^{(i)}\right) - \left[f(y) + \nabla f(y)^T \left(y^{(i)} - y\right) + \frac{1}{2}\left(y^{(i)} - y\right)^T \nabla^2 f(y) \left(y^{(i)} - y\right)\right],$$

we can use Theorem 4.6 with $\gamma = L_{\nabla^2}$ and $p = y^{(i)} - y$ to conclude

$$\left|e_O^{(i)}(y)\right| \leq \frac{1}{6} L_{\nabla^2} \left\|y^{(i)} - y\right\|^3 \leq \frac{4}{3} L_{\nabla^2} \Delta^3. \tag{19}$$

Furthermore, we notice that

$$\begin{aligned} f\left(y^{(i)}\right) - f(y) &= \nabla f(y)^T \left(y^{(i)} - y\right) + \frac{1}{2}\left(y^{(i)} - y\right)^T \nabla^2 f(y) \left(y^{(i)} - y\right) + e_O^{(i)}(y) \\ &= \left(y^{(i)} - y\right)^T (Hy + g - e_g(y)) + \frac{1}{2}\left(y^{(i)} - y\right)^T \nabla^2 f(y) \left(y^{(i)} - y\right) + e_O^{(i)}(y). \end{aligned}$$

If we subtract this from (18), we find

$$e_O^{(i)}(y) - e_f(y) = \left(y^{(i)} - y\right)^T e_g(y) + \frac{1}{2}\left(y^{(i)} - y\right)^T \left[H - \nabla^2 f(y)\right] \left(y^{(i)} - y\right)$$

or

$$\begin{aligned} e_O^{(i)}(y) = &\left[e_f(y) - y^T e_g(y) + \frac{1}{2} y^T \left[H - \nabla^2 f(y)\right] y\right] \\ &+ \left[e_g(y) + \left(H - \nabla^2 f(y)\right) y\right]^T y^{(i)} + \frac{1}{2}\left(y^{(i)}\right)^T \left[H - \nabla^2 f(y)\right] \left(y^{(i)}\right). \end{aligned} \tag{20}$$

To put this equation in matrix form, we introduce some notation. Given some $n \times n$ matrix A, let $e_U : \mathbb{R}^n \times \mathbb{R}^n \to \mathbb{R}^{n + \frac{n(n-1)}{2}}$ linearize the upper triangular portion of A:

$$e_U(A) = \left(\frac{1}{2}A_{1,1}\ ,\ \frac{1}{2}A_{2,2}\ ,\ \ldots\ ,\ \frac{1}{2}A_{n,n}\ ,\ A_{1,2}\ ,\ A_{1,3}\ ,\ \ldots\ ,\ A_{n-1,n}\right)^T$$

in a similar order to that used in $\bar{\Phi}$ (see (11)). Then, with $e_O(y) = \left(e_O^{(0)}(y), e_O^{(1)}(y), \ldots, e_O^{(p)}(y)\right)^T$, we have that (20) becomes

$$M(\Phi, Y)\begin{pmatrix} e_f(y) + y^T e_g(y) + y^T e_h(y) y \\ e_g(y) + e_h(y) y \\ e_U(e_h(y)) \end{pmatrix} = e_O(y).$$

Thus,

$$\begin{aligned}\left\| M(\bar{\Phi}, Y)\begin{pmatrix} e_f(y) + y^T e_g(y) + y^T e_h(y) y \\ e_g(y) + e_h(y) y \\ e_Q(e_h(y)) \end{pmatrix}\right\|_2 &\leq \|e_O(y)\|_2 \\ &\leq \sqrt{p+1}\,\|e_O(y)\|_\infty \\ &\leq \frac{4}{3}\sqrt{p+1} L_{\nabla^2} \Delta^3 \\ &= \kappa_0 \Delta^3,\end{aligned}$$

where $\kappa_0 = \frac{4}{3}\sqrt{p+1} L_{\nabla^2}$. To remove the dependence of Δ on $M(\bar{\Phi}, Y)$, we rewrite this with a scaling matrix:

$$\left\| M\left(\bar{\Phi}, Y\right)\begin{pmatrix} 1 & 0 & 0 \\ 0 & \Delta^{-1} I_n & 0 \\ 0 & 0 & \Delta^{-2} I_{n+\frac{n(n-1)}{2}} \end{pmatrix}\begin{pmatrix} e_f(y) + y^T e_g(y) + y^T e_H(y) y \\ \Delta\left(e_g(y) + e_H(y) y\right) \\ \Delta^2\left(e_Q\left(e_H(y)\right)\right) \end{pmatrix}\right\|_2 \leq \kappa_0 \Delta^3.$$

With (13), this provides

$$\left\| e_f(y) + y^T e_g(y) + y^T e_H(y) y \right\|_2 \leq \kappa_0 \left\| \hat{M}^{-1}(\bar{\Phi}, Y)\right\|_2 \Delta^3, \tag{21}$$

$$\Delta \left\| e_g(y) + e_H(y) y \right\|_2 \leq \kappa_0 \left\| \hat{M}^{-1}(\bar{\Phi}, Y)\right\|_2 \Delta^3, \tag{22}$$

$$\Delta^2 \left\| e_Q\left(e_H(y)\right)\right\|_2 \leq \kappa_0 \left\| \hat{M}^{-1}(\bar{\Phi}, Y)\right\|_2 \Delta^3. \tag{23}$$

We see that (23) immediately provides

$$\|e_H(y)\|_2 \leq \|e_H(y)\|_F \leq \sqrt{2}\left\|e_Q\left(e_H(y)\right)\right\|_2 \leq \sqrt{2}\kappa_0 \left\| \hat{M}^{-1}(\bar{\Phi}, Y)\right\|_2 \Delta$$

where $\|\cdot\|_F$ is the Frobenius norm $\|A\|_F = \sqrt{\sum_i \sum_j A_{i,j}^2}$. Similarily, the triangle inequality and (22) imply

$$\begin{aligned}\|e_g(y)\|_2 &\leq \|e_g(y) + e_H(y) y\|_2 + \|e_H(y)\|_2 \|y\|_2 \\ &\leq \kappa_0 \left\| \hat{M}^{-1}(\bar{\Phi}, Y)\right\|_2 \Delta^2 + \left[\sqrt{2}\kappa_0 \left\| \hat{M}^{-1}(\bar{\Phi}, Y)\right\|_2 \Delta\right] \Delta \\ &\leq \left(1 + \sqrt{2}\right) \kappa_0 \left\| \hat{M}^{-1}(\bar{\Phi}, Y)\right\|_2 \Delta^2.\end{aligned}$$

Finally, using (21),

$$\begin{aligned}\left|e^f(y)\right| &\leq \|e_g(y)\| \Delta + \|e_H(y)\| \Delta^2 + \kappa_0 \left\| \hat{M}^{-1}(\bar{\Phi}, Y)\right\|_2 \Delta^3 \\ &\leq \left[\left(1 + \sqrt{2}\right) + \sqrt{2} + 1\right] \kappa_0 \left\| \hat{M}^{-1}(\bar{\Phi}, Y)\right\| \Delta^3.\end{aligned}$$

□

4.6 Model Improvement Algorithms

Efficient implementations of model-based methods re-use sample points from previous iterations that fall within (or at least near) the current trust region. New points are then added to the sample set using a model improvement algorithm as described in [8] and stated here within Algorithm 1.

The model improvement algorithm starts with a set of $p+1$ sample points that have been shifted and scaled so that they lie within a ball $B_2(0;\Delta)$, with $\Delta \geq 1$. The algorithm then uses LU factorization with partial pivoting of the associated Vandermonde matrix to construct a set of pivot polynomials $\{u_0, \ldots, u_p\}$ that are closely related to the Lagrange polynomials.

Each iteration of the algorithm identifies a point to include in the final sample set. In particular, on the ith iteration, the points $y^{(0)}, \ldots, y^{(i-1)}$ have already been included. If a point $y^{(j)}$, $j \geq i$ from the original set can be found such that $u_i(y^{(j)})$ has a sufficiently large magnitude (i.e., $|u_i(y^{(j)})| \geq \xi_{\min}$), then that point is added to the final sample set (by swapping it with $y^{(i)}$). However, if no such point can be found, it indicates that including any of the remaining points in the final sample set would result in a poorly poised set. Therefore, the point $y^{(i)}$ is replaced by a new point that is obtained by maximizing $|u_i(x)|$ over the unit ball $B_2(0;1)$. The pivot polynomials are then updated so that $u_j(y^{(i)}) = 0$ for $j > i$. At the successful completion of the algorithm, the final set of points Y is Λ-poised over $B_2(0;\Delta)$, where Λ depends on Δ, n and $\xi_{\min}$, and is inversely proportional to $\xi_{\min}$ (see Theorem 4.10 below).

Note. Typically, we have fewer than $p+1$ previously evaluated sample points within the trust region at the beginning of each iteration. Since the Model Improvement Algorithm requires a starting set of $p+1$ points, we add copies of $y^{(0)}$ to create a set with $p+1$ points.

Algorithm 1: Model Improvement Algorithm

Step 0 (Initialization)
Given $\xi_{\text{min}} \in (0, 1/4]$, $\Delta \geq 1$, a set $Y = \left\{y^{(0)}, y^{(1)}, \ldots, y^{(p)}\right\} \subset B_2(0;\Delta)$ of $p+1$ points, initialize $i = 0$, and let $u_i(x) = \bar{\phi}_i(x)$ be the monomial basis for $\mathcal{P}_n^d$.

Step 1 (Pivot)
If $i = 0$, go to Step 3.
Compute the next pivot index $j_i^{\max} = \arg\max_{i \leq j \leq |Y|-1} \left|u_i\left(y^{(j)}\right)\right|$, and swap points $y^{(i)}$ and $y^{(j_i^{\max})}$ within Y.

Step 2 (Check threshold)
If $\left|u_i\left(y^{(i)}\right)\right| \geq \xi_{\text{min}}$ then go to Step 3.
If $\left|u_i\left(y^{(i)}\right)\right| < \xi_{\text{min}}$, then replace $y^{(i)} \leftarrow \arg\max_{x \in B_2(0;1)} |u_i(x)|$.
If $\left|u_i\left(y^{(i)}\right)\right| < \xi_{\text{min}}$ after this replacement, **Stop**: ξ_{min} was chosen too small.

Step 3 (Gaussian elimination)
For $j = i+1, \ldots, p$
Set $u_j(x) \leftarrow u_j(x) - \frac{u_j\left(y^{(i)}\right)}{u_i\left(y^{(i)}\right)} u_i(x)$
If $i = p$ then **Stop**, otherwise. Set $i \leftarrow i+1$ and go to Step 1

Notice that Algorithm 1 constructs a set of pivot polynomials $u = \{u_0, u_1, \ldots, u_p\}$, and that $M(u, Y)$ is the upper triangular factor of the LU factorization of $M(\bar{\Phi}, Y)$. The first instruction of Step 1 of Algorithm 1

ensures that the first point is not replaced. The first basis polynomial is always 1, so we can be sure that the first pivot will not need to be replaced.

Observe that Algorithm 1 uses a threshold parameter $\xi_{\min}$ to select the next sample point. If $\xi_{\min}$ is too large, it is possible that there might not exist a point $x \in B_2(0;1)$ in the unit ball such that $\|u_i(x)\| \geq \xi_{\min}$. In this case, the algorithm stops prematurely in Step 2 with a certification that $\xi_{\min}$ is too small. However, in Lemma 4.9 below, we show that this cannot happen as long as $0 < \xi_{\min} \leq \frac{1}{4}$. To establish the result, we first need the following Lemma from [8].

Lemma 4.8 *([8, Lemma 6.7])*
Let $v^T\bar{\Phi}(x)$ be a quadratic polynomial, where $\|v\|_\infty = 1$ and $\bar{\Phi}$ is the monomial basis. Then,

$$\max_{x \in B_2(0;1)} \left\| v^T \bar{\Phi}(x) \right\| \geq \frac{1}{4}.$$

Lemma 4.9 *For any given $\xi_{\min} \in (0, 1/4]$, Algorithm 1 computes a set Y of $p+1$ points in the ball $B_2(0;\Delta)$ for which the pivots of the LU factorization of $M = M(\bar{\Phi}, Y)$ satisfy*

$$\left\| u_i(y^{(i)}) \right\| \geq \xi_{\min}, \quad i = 0, \ldots, p.$$

Proof:

The first pivot polynomial is $u_0(x) = 1 = (1, 0, \ldots, 0)\cdot\bar{\Phi}(x)$, so that $\left\|u_0\left(y^{(0)}\right)\right\| = 1 \geq \xi_{\min}$. For $i > 1$, the $(i+1)$st pivot polynomial $u_i(x)$ can be expressed as $v^T\bar{\Phi}(x)$, where $v = (v_0, \ldots, v_{i-1}, 1, 0, \ldots, 0)^T$. Observe that $\|v\|_\infty \geq 1$. Let $\bar{v} = v/\|v\|_\infty$. Then by Lemma 4.8,

$$\max_{x \in B_2(0;1)} u_i(x) = \max_{x \in B_2(0;1)} \left| v^T \bar{\phi}(x) \right| = \|v\|_\infty \max_{x \in B_2(0;1)} \left| \bar{v}^T \bar{\phi}(x) \right| \geq \frac{1}{4}\|v\|_\infty \geq \frac{1}{4}.$$

It follows that the algorithm does not stop in Step 2 for $\xi_{\min} \leq 1/4$. Hence, at the end of the i-th iteration, we have that $\left|u_i\left(y^{(i)}\right)\right| \geq \xi_{\min}$. Moreover, after the i iteration has been completed, u_i, and $y^{(i)}$ are never altered. Thus, the algorithm terminates with $\left|u_i\left(y^{(i)}\right)\right| \geq \xi_{\min}$ for $i = 0, \ldots, p$.

Finally, observe that all points in the final sample set Y are selected either from the original sample set, which is contained in $B_2(0;\Delta)$, or are obtained in Step 2 from within $B_2(0;1)$. Since $\Delta \geq 1$, it follows that $Y \subset B_2(0;\Delta)$.
□

Theorem 4.10 *Let $\Delta \geq 1$, and suppose that Algorithm 1 is called on a sample set $Y = \left\{y^{(0)}, y^{(1)}, \ldots, y^{(p)}\right\} \subset B_2(0;\Delta)$ resulting in a new sample set $\hat{Y}$, and let $\bar{\Phi}$ be the monomial basis* (11)*. Then,*

- $\left\| M(\bar{\Phi}, \hat{Y})^{-1} \right\|_2$ *as defined by* (7)*, is bounded by a constant depending only on p and ξ_{min}.*
- *The resulting sample set $\hat{Y}$ is Λ-poised over $B_2(0;\Delta)$ for quadratic interpolation, where $\Lambda > 0$ is a constant that depends only on Δ, ξ_{min} and n and is inversely proportional to ξ_{min}.*
- *The initial point $y^{(0)}$ is retained in the resulting sample set: $y^{(0)} \in \hat{Y}$.*

Proof:

Note that Algorithm 1 never swaps or replaces $y^{(0)}$. By Lemma 4.9, $\hat{Y} \subset B_2(0;\Delta)$ and the pivots in the LU-factorization of $M(\bar{\phi}, Y)$ are bounded below by $\xi_{\min}$; that is, $\left|u_i\left(y^{(i)}\right)\right| \geq \xi_{\min},\ \forall 0 \leq i \leq p$. Thus, by [8, Section 6.7, Exercise 3], we have that

$$\left\|M\left(\bar{\Phi}, Y\right)^{-1}\right\| \leq \frac{\sqrt{p+1}\epsilon_{growth}}{\xi_{\min}}$$

where ϵ_{growth} is a potentially large, but finite estimate of the growth factor in the LU factorizations. By Theorem 4.2, using the fact that $\Delta \geq 1$, we have that Y is Λ-poised in $B_2(0;\Delta)$, where

$$\Lambda = \left(\frac{\sqrt{p+1}\epsilon_{growth}}{\xi_{\min}}\right)\sqrt{p+1}\frac{\Delta^2}{2} = \frac{(p+1)\epsilon_{growth}}{2\xi_{\min}}\Delta^2.$$

□

4.7 Criticality Measure

A key component of any optimization algorithm is the *criticality measure*, which is used to define stopping criteria for the algorithm. The criticality measure $\chi(x^{(k)})$ at an iterate $x^{(k)}$ is a nonnegative quantity that is zero if and only if $x^{(k)}$ satisfies necessary optimality conditions. For unconstrained problems, a typical choice of criticality measure for classical (gradient-based) algorithms is given by $\chi(x) = \|\nabla f(x)\|$ since the first order optimality condition is $\nabla f(x) = 0$. The algorithm then terminates when $\chi(x) < \tau_\xi$, where $\tau_\xi > 0$ is a stopping tolerance.

For derivative-free optimization, $\nabla f(x)$ is not available, so the true criticality measure is replaced by the model criticality measure $\chi_m^{(k)}(x) = \left\|\nabla {m_f}^{(k)}(x)\right\|$. Note however that $\chi_m^{(k)}$ is only an accurate approximation of $\chi(x)$ if the gradient of the model function $\nabla {m_f}^{(k)}(x)$ is a close approximation to $\nabla f(x)$. For this reason, derivative-free algorithms require not only that $\chi_m^{(k)}(x)$ is small, but also that the model functions are accurate. To accomplish this, we require poised sample sets as discussed in Section 4.5 with a trust-region converging to zero. Once the criticality measure has reached a small enough threshold $\tau_\xi > 0$ and the trust region is small enough ($\Delta_k < \tau_\Delta$), we can terminate the algorithm.

If the feasible region $\mathcal{F}$ is convex, a classic criticality measure (see, for example [9] [74]) is given by

$$\chi(x) = \left\|x - \text{Proj}_{\mathcal{F}}\left(x - \nabla f\left(x\right)\right)\right\|.$$

If the feasible region is not convex, the projection operation is not well-defined, so this criticality measure is not helpful. Moreover, even with a convex feasible region, we need a way to determine the projection, so we prefer a criticality measure based on the functions defining the constraints. In particular, for a first order criticality measure at $x^{(k)}$, we define

$$F_c^{(k)} = \left\{x \in \mathbb{R}^n \middle| c_i\left(x^{(k)}\right) + \nabla c_i\left(x^{(k)}\right)^T\left(x - x^{(k)}\right) \leq 0\ \forall i \in [m]\right\} \tag{24}$$

and use

$$\chi_c^{(k)}(x) = \left\|x - \text{Proj}_{F_c^{(k)}}\left(x - \nabla f\left(x\right)\right)\right\|. \tag{25}$$

Note that $\chi_c^{(k)}(x^{(k)}) = 0$ if and only if $x^{(k)}$ satisfies the first order Karush-Kuhn-Tucker conditions. Unfortunately, without derivative information, we cannot calculate $\chi_c^{(k)}$ directly, so in our algorithms we

approximate it with

$$\chi_m^{(k)} = \left\| x^{(k)} - \text{Proj}_{\mathcal{F}_m^{(k)}} \left(x^{(k)} - \nabla {m_f}^{(k)} \left(x^{(k)} \right) \right) \right\|, \tag{26}$$

where $\mathcal{F}_m^{(k)}$ is the model feasible region defined by

$$\mathcal{F}_m^{(k)} = \left\{ x \in \mathbb{R}^n \middle| m_{c_i}^{(k)}(x) \leq 0 \quad \forall 1 \leq i \leq m \right\}. \tag{27}$$

This quantity measures how far the current iterate $x^{(k)}$ is from satisfying the first order optimality conditions for the model function ${m_f}^{(k)}$. In turn, as $\Delta_k \to 0$, the model ${m_f}^{(k)}$ better approximates f, $\mathcal{F}_m^{(k)}$ better approximates $\mathcal{F}$ and $x^{(k)}$ approaches a critical point for f.

4.8 Trust Regions

To define our algorithms, we distinguish between three types of trust regions: the *sample region* $T_{\text{sample}}^{(k)}$, the *search region* $T_{\text{search}}^{(k)}$, and the *outer trust region* $T_{\text{out}}^{(k)}$. For unconstrained optimization, these trust regions are typically identical and are chosen to be an L_2-ball of radius Δ_k. However, for constrained optimization, it is useful to distinguish between the two regions.

The sample region is where the algorithm chooses sample points, constraining them to ensure their feasibility. The search region defines the feasible region for the trust region subproblem. Both $T_{\text{sample}}^{(k)}$ and $T_{\text{search}}^{(k)}$ lie within the *outer trust region*, $T_{\text{out}}^{(k)} = B_\infty \left(x^{(k)}, \Delta_k \right)$, which is an L_∞-ball of radius Δ_k centered at the current iterate $x^{(k)}$:

$$T_{\text{out}}^{(k)} = B_\infty \left(x^{(k)}, \Delta_k \right) = \left\{ x \in \mathbb{R}^n \middle| \, x_i^{(k)} - \Delta_k \leq x_i \leq x_i^{(k)} + \Delta_k \quad \forall i \in [m] \right\}. \tag{28}$$

To allow for the best possible trial point, we would like the search region to be as large as possible within the outer trust region while remaining feasible. Ideally, we would set $T_{\text{search}}^{(k)} = T_{\text{out}}^{(k)} \cap \mathcal{F}$. However, this is only possible when the constraints are known. In **??**, we can only take $T_{\text{search}}^{(k)} = T_{\text{out}}^{(k)} \cap \mathcal{F}_m^{(k)}$. Observe that using an L_∞ ball instead of an L_2 ball results in linear constraints for the model problem, so that that $T_{\text{search}}^{(k)}$ is a polytope.

5 linear constraints

This chapter considers the case of linear constraints. Specifically, we present an algorithm for solving

$$\begin{array}{ll} \min_{x \in \mathbb{R}^n} & f(x) \\ \text{s.t.} & Ax \leq b \end{array} \tag{29}$$

where $f : \mathbb{R}^n \to \mathbb{R}$ is a black-box function, A is an $m \times n$ matrix and $b \in \mathbb{R}^m$. For convenience, we assume that $\|A_i\| = 1$ for each $i \in [m]$.

We consider two main strategies for defining $T_{\text{sample}}^{(k)}$. The first method, which is described in Section 8, defines $T_{\text{sample}}^{(k)}$ to be an ellipsoidal region

$$T_{\text{sample}}^{(k)} = \left\{ x \in \mathbb{R}^n \middle| \left(x - c^{(k)} \right)^T Q^{(k)} \left(x - c^{(k)} \right) \leq \frac{1}{2} \delta_k^2 \right\}. \tag{30}$$

We choose the positive definite, symmetric matrix $Q^{(k)}$, vector $c^{(k)} \in \mathbb{R}^n$, and constant $\delta_k > 0$ so that the sample region lies within the intersection of the feasible region and the outer trust region: $T^{(k)}_{\text{sample}} \subseteq \mathcal{F} \cap T^{(k)}_{\text{out}}$. This is referred to as the *ellipsoidal trust region approach*. For this approach, we present a method for selecting a well-poised set of sample points by a straightforward application of the model-improvement algorithm Algorithm 1 to a transformed problem in which the ellipsoidal region is mapped onto a ball.

In the second method, described in Section 9, we define

$$T^{(k)}_{\text{sample}} = T^{(k)}_{\text{search}} = T^{(k)}_{\text{out}} \cap \mathcal{F}.$$

Since this trust region is a bounded polyhedron, we call this the *polyhedral trust region approach*. While this method is perhaps more intuitive than the first approach, it is not immediately clear how to construct a well-poised sample set over the polyhedral region. We present one method with this aim, but we do not show equivalent error bounds as those found for ellipsoidal regions. To select sample points from this polyhedral region, we present Algorithm 4 as a modification of model-improvement algorithm (Algorithm 1). For this approach, the search trust region is given by $T^{(k)}_{\text{search}} = T^{(k)}_{\text{out}} \cap \mathcal{F}$.

In all cases, the trust regions are related by the inclusions

$$T^{(k)}_{\text{sample}} \subseteq T^{(k)}_{\text{search}} \subseteq T^{(k)}_{\text{out}} \cap \mathcal{F}.$$

Both methods fall into a general algorithmic framework, which we describe next.

6 Algorithm Template

All of the methods in this chapter follow an algorithmic template described in [9].

Algorithm 2: Linear Always-feasible Constrained Derivative-free Algorithm

Step 0 (Initialization)
Choose $\tau_\xi \geq 0$, $\tau_\Delta \geq 0$, $\omega_{\text{dec}} \in (0,1)$, $\omega_{\text{inc}} \geq 1$, $\gamma_{\text{suff}} \in (0,1]$, $\gamma_{\text{min}} \in (0, \gamma_{\text{suff}})$, and $\alpha > 0$.
Initialize $k \leftarrow 0$, $x^{(0)} \in \mathcal{F}$, and $0 < \Delta_0 \leq \Delta_{\text{max}}$.

Step 1 (Construct the Model)
Construct the trust region $T^{(k)}_{\text{sample}}$.
Construct ${m_f}^{(k)}$ by calling Algorithm 3 on $T^{(k)}_{\text{sample}}$.

Step 2 (Check Stopping Criteria)
Compute the criticality measure $\chi^{(k)}_m$ as in (26).
If $\chi^{(k)}_m < \tau_\xi$ and $\Delta_k < \tau_\Delta$, **stop**: return $x^{(k)}$ as the solution.
Otherwise, if $\Delta_k > \alpha \chi^{(k)}_m$, set $\Delta_{k+1} \leftarrow \omega_{\text{dec}} \Delta_k$, $x^{(k+1)} \leftarrow x^{(k)}$, $k \leftarrow k+1$ and **go to Step 1**.

Step 3 (Solve the Trust Region Subproblem)
Compute $s^{(k)} = \arg\min_{s \in T^{(k)}_{\text{search}} - x^{(k)}} {m_f}^{(k)}\left(x^{(k)} + s\right)$.

Step 4 (Test for Improvement)
Evaluate $f(x^{(k)} + s^{(k)})$ and ρ_k as in (4).
If $\rho_k < \gamma_{\text{min}}$, then $x^{(k+1)} \leftarrow x^{(k)}$, otherwise $x^{(k+1)} \leftarrow x^{(k)} + s^{(k)}$.
If $\rho_k < \gamma_{\text{suff}}$, then $\Delta_{k+1} \leftarrow \min\{\Delta_{\text{max}}, \omega_{\text{dec}}\Delta_k\}$.
If $\rho_k \geq \gamma_{\text{suff}}$, and $\|s^{(k)}\|_\infty = \Delta_k$, then $\Delta_{k+1} \leftarrow \omega_{\text{inc}}\Delta_k$.
If $\rho_k \geq \gamma_{\text{suff}}$, and $\|s^{(k)}\|_\infty < \Delta_k$, then $\Delta_{k+1} \leftarrow \Delta_k$.
Set $k \leftarrow k + 1$ and go to Step 1.

In Step 1, this algorithm relies on constructing and choosing sample points from $T^{(k)}_{\text{sample}}$. Different variations of the algorithm implement this step differently with their *ConstructTrustRegion* subroutine. We show convergence for the ellipsoidal approach, in which sample points are chosen using Algorithm 1 and the choice of $T^{(k)}_{\text{sample}}$ satisfies some additional restrictions. We discuss those restrictions here.

To simplify notation, define $G^{(k)} = \begin{pmatrix} A \\ I \\ -I \end{pmatrix}$ and $g^{(k)} = \begin{pmatrix} b \\ x_i^{(k)} + \Delta_k \\ -x_i^{(k)} + \Delta_k \end{pmatrix}$. We can then define

$$\begin{aligned} \mathcal{P}^{(k)} = \mathcal{F} \cap T^{(k)}_{\text{out}} &= \{x \in \mathbb{R}^n |\ Ax \leq b, \|x - x^{(k)}\|_\infty \leq \Delta_k\} \\ &= \{x \in \mathbb{R}^n | G^{(k)}x \leq g^{(k)}\}. \end{aligned} \tag{31}$$

Note that we will still have $\|G_i^{(k)}\| = 1$ for each $i \in [m]$.

Definition 6.1 *Let $\left\{Q^{(k)}\right\}$ be a sequence of positive definite symmetric matrices, $\left\{c^{(k)}\right\} \subset \mathbb{R}^n$ be a sequence of points, and $\{\delta_k\}$ be a sequence of positive scalars. For each $k \in \mathbb{N}$, define the ellipsoids*

$$E^{(k)} = \left\{ x \in \mathbb{R}^n \middle| \left(x - c^{(k)}\right)^T Q^{(k)} \left(x - c^{(k)}\right) \leq \frac{1}{2}\delta_k^2 \right\}, \text{and}$$
$$\hat{E}^{(k)} = \left\{ x \in \mathbb{R}^n \middle| \left(x - c^{(k)}\right)^T Q^{(k)} \left(x - c^{(k)}\right) \leq \delta_k^2 \right\}.$$

The sequence $\left\{E^{(k)}\right\}$ is said to be suitable if all of the following are satisfied:

1. *$E^{(k)} \subseteq \mathcal{P}^{(k)}$, where $\mathcal{P}^{(k)}$ is defined by* (31)*,*
2. *$x^{(k)} \in \hat{E}^{(k)}$,*
3. *The condition number $\kappa\left(Q^{(k)}\right)$ is bounded independently of k.*

7 Convergence Analysis

Here, we analyze the convergence behavior of Algorithm 2. For our analysis, we assume that Ω is some open set containing $\mathcal{F}$ and make the following assumptions about the problem and model functions:

Assumption 7.1 *The function f is twice continuously differentiable with Lipschitz continuous Hessian in Ω. That is, there exists constant $L_{\nabla^2} > 0$ such that*

$$\left\|\nabla^2 f(x) - \nabla^2 f(y)\right\| \leq L_{\nabla^2} \|x - y\| \quad \forall x, y \in \Omega. \tag{32}$$

Assumption 7.2 *The function f is bounded below over Ω. That is,*

$$f(x) \geq f_{min} \quad \forall x \in \Omega. \tag{33}$$

Assumption 7.3 *There exists a point $\bar{x}$ within the interior of the feasible region. Namely, using A and b from* (29)*:*

$$A\bar{x} < b. \tag{34}$$

Assumption 7.4 *The Hessians of f are uniformly bounded at each iterate. That is, there exists a constant $\beta \geq 1$ such that*

$$\left\| \nabla^2 f\left(x^{(k)}\right) \right\| \leq \beta - 1 \quad \forall k \in \mathbb{N}.$$

Our analysis closely follows that of [9], which presents a class of trust region algorithms for derivative-free optimization with convex constraints. In fact, if the tolerances τ_χ and τ_Δ are set to zero, then Algorithm 2 is a particular implementation of the algorithm studied within this reference. The only modification is in the trust region subproblem, in which we allow for more trial points with a L_∞ ball instead of an L_2 ball. It is therefore sufficient to show that the model functions generated by our method satisfy the hypotheses required for their analysis.

We have included a modification that $\Delta_k \leq \Delta_{\max}$; however, this does not alter any of the convergence results. In fact, the authors allow $\omega_{\text{inc}} = 1$, so that the trust region radius can never be increased.

The authors assume that the model of the objective is quadratic and satisfies an efficiency condition. Namely, there exists a constant κ_f such that for all $k \in \mathbb{N}$,

$$m_f{}^{(k)}\left(x^{(k)}\right) - m_f{}^{(k)}\left(x^{(k)} + s^{(k)}\right) \geq \kappa_f \chi_m^{(k)} \min\left\{ \frac{\chi_m^{(k)}}{1 + \left\| \nabla^2 m_f{}^{(k)}(x^{(k)}) \right\|}, \Delta_k, 1 \right\}, \tag{35}$$

where $\chi_m^{(k)}$ is defined by (26). The Generalized Cauchy Point is shown to satisfy (35) within [74]. This is a reasonable choice for $s^{(k)}$, but the exact solution as chosen within Algorithm 2 necessarily also satisfies (35) as it can only improve upon the Generalized Cauchy Point. Also, note that although [9] presents a criticality measure based on the true projection onto the constraints, for linear constraints, this precisely coincides with the models used in (26).

The authors of [9] also make four explicit assumptions. Their first assumption H_1 requires the function f to be differentiable and its gradient ∇f to be Lipschitz continuous with constant $L > 0$ in Ω. We have assumed Assumption 7.1 which is a strictly stronger assumption. As their second assumption H_2, they assume our Assumption 7.2. Their H_3 requires the Hessians of $m_f{}^{(k)}$ to be uniformly bounded. That is, there exists a constant $\beta \geq 1$ such that

$$\left\| \nabla^2 m_f{}^{(k)}\left(x^{(k)}\right) \right\| \leq \beta - 1 \quad \forall k \geq 0. \tag{36}$$

Showing that our construction of m_f satisfies (36) is the topic of Section 7.1. Their last assumption H_4 is an accuracy condition. Namely, they assume there exists a constant $\delta_g > 0$ such that for all $k \in \mathbb{N}$

$$\left\|\nabla m_k\left(x^{(k)}\right) - \nabla f\left(x^{(k)}\right)\right\| \leq \delta_g \Delta_k. \tag{37}$$

Showing that our construction of m_f satisfies (37) is the topic of Section 7.2.

7.1 Bounded Hessians

First, we show that Assumption 7.4 can be used instead of (36).

Lemma 7.1 *Suppose that Assumption 7.1 and Assumption 7.4 hold. Let ${m_f}^{(k)}$ be a quadratic model interpolating f on a Λ-poised sample set Y over $B_2\left(x^{(k)}, \Delta_k\right)$ for each $k \in \mathbb{N}$. Then* (36) *is also satisfied.*

Proof:

By Assumption 7.4, we can choose $\beta_1 \geq 1$ to be such that for all $k \in \mathbb{N}$:

$$\left\|\nabla^2 f\left(x^{(k)}\right)\right\| \leq \beta_1 - 1.$$

Because Y is Λ-poised, we can use Lemma 4.5 to bound $\left\|\hat{M}\left(\bar{\Phi}, Y\right)^{-1}\right\|$, and apply Theorem 4.7. Thus, we see that (16) is satisfied. Combining this with $\Delta_k \leq \Delta_{\text{max}}$, we see that

$$\left\|\nabla^2 f\left(x^{(k)}\right) - \nabla^2 m_f\left(x^{(k)}\right)\right\| \leq \kappa_h \Delta_k \leq \kappa_h \Delta_{\text{max}}$$

Defining $\beta_2 = \kappa_h \Delta_{\text{max}} + \beta_1 \geq 1$, we see that

$$\left\|\nabla^2 m_f\left(x^{(k)}\right)\right\| \leq \left\|\nabla^2 m_f\left(x^{(k)}\right) - \nabla^2 f\left(x^{(k)}\right)\right\| + \left\|\nabla^2 f\left(x^{(k)}\right)\right\| \leq \beta_2 - 1.$$

□

7.2 Satisfying the accuracy assumption

After an ellipsoidal $T^{(k)}_{\text{sample}}$ is constructed, sample points are then selected by applying Algorithm 1 to a transformed problem in which the ellipsoid $E^{(k)}$ is mapped onto a unit ball. This process is summarized here.

Given some symmetric matrix $Q^{(k)} \succ 0$, we can give $Q^{(k)}$ an eigen-decomposition $Q^{(k)} = VD^2V^T$, where $V^TV = I$ and D is a diagonal matrix with positive entries. For any $\delta > 0$ define $T : \mathbb{R}^n \times \mathbb{R}^n_+ \to \mathbb{R}^n$ by $T(x;\delta) = \frac{\delta}{\delta_k} DL^T\left(x - c^{(k)}\right)$. Note that T is invertible and can map $T^{(k)}_{\text{sample}}$ onto the δ-ball $B_2(0;\delta) = \{u \in \mathbb{R}^n \mid \|u\| \leq \delta\}$:

$$\delta^2 \geq \|T\left(x; \sqrt{2}\delta\right)\|^2 = \left\|\frac{\sqrt{2}\delta}{\delta_k} DV^T(x - c^{(k)})\right\|^2 = \frac{2\delta^2}{\delta_k^2}\left(x - c^{(k)}\right)^T Q\left(x - c^{(k)}\right)$$
$$\iff \left(x - c^{(k)}\right)^T Q\left(x - c^{(k)}\right) \leq \frac{1}{2}\delta_k^2.$$

Likewise, with $\hat{E}^{(k)}$ defined in Definition 6.1, we have that $T\left(\hat{E}^{(k)}; \delta\right) = B_2(0;\delta)$.

Algorithm 3: Model Construction Algorithm

Step 0 (Initialization)
Given a function $f : \mathbb{R}^n \to \mathbb{R}$; ellipsoid parameters $Q^{(k)} \succ 0$, $c^{(k)}$, and δ_k; and sample set Y.

Step 1 (Construct the Transformation)
Give $Q^{(k)}$ its eigen-decomposition $Q^{(k)} = VD^2V^T$, and define $T(x; \sqrt{2}) = \frac{\sqrt{2}}{\delta_k} DV^T \left(x - c^{(k)}\right)$.

Step 1 (Construct the Sample Set)
Construct the transformed sample set $\hat{Y}$ by computing $T\left(y^{(i)}; \sqrt{2}\right)$ for each $y^{(i)} \in Y$.
Call Algorithm 1 with $\Delta = 2$ on this sample to produce an improved sample set $\hat{Y}'$.
Construct the unshifted sample set Y' by computing $T^{-1}\left(\hat{y}^{(i)}; \sqrt{2}\right)$ for each $\hat{y}^{(i)} \in \hat{Y}'$.

Step 2 (Construct Model Functions)
Evaluate f at each $y^{(i)} \in Y'$, and construct m_f with (9)

This section analyzes the accuracy of the model functions constructed from the output of Algorithm 3. We first show the intermediate result Lemma 7.2 before satisfying (37). In particular, Theorem 7.3 establishes error bounds based on the condition number of $Q^{(k)}$. In subsequent sections, we will describe several methods for choosing $Q^{(k)}$ and $c^{(k)}$.

The following theorem allows us to translate error bounds derived for the ball $B_2(0; \delta)$ to the ellipsoid $E^{(k)}$.

Lemma 7.2 *Let $\delta_k > 0$, $c^{(k)} \in \mathbb{R}^n$ and a symmetric matrix $Q^{(k)} \succ 0$ be given. Let the eigen decomposition of $Q^{(k)}$ be $Q^{(k)} = VD^2V^T$, $V^TV = I$, where D is diagonal with positive entries. Let $\hat{E}^{(k)}$ be defined as in Definition 6.1:*

$$\hat{E}^{(k)} = \left\{ x \in \mathbb{R}^n \middle| \left(x - c^{(k)}\right)^T Q^{(k)} \left(x - c^{(k)}\right) \leq \delta_k^2 \right\}$$

For $\delta > 0$, define the transformation $T(x; \delta) = \frac{\delta}{\delta_k} DV^T \left(x - c^{(k)}\right)$, and let $\delta_r = \max_{x \in \hat{E}^{(k)}} \|x - c^{(k)}\|$. Let $\hat{m}_f(u)$ be a model of the shifted objective $\hat{f}(u) = f(T^{-1}(u; \delta_r))$ such that, for constants $\kappa_{ef}, \kappa_{eg}, \kappa_{eh} > 0$, the following error bounds hold for all $u \in B_2(0; \delta_r)$:

$$|\hat{m}_f(u) - \hat{f}(u)| \leq \kappa_{ef} \delta_r^3, \tag{38}$$

$$\|\nabla \hat{m}_f(u) - \nabla \hat{f}(u)\| \leq \kappa_{eg} \delta_r^2, \tag{39}$$

$$\|\nabla^2 \hat{m}_f(u) - \nabla^2 \hat{f}(u)\| \leq \kappa_{eh} \delta_r. \tag{40}$$

Then, with

$$\kappa'_{ef} = \kappa_{ef},$$
$$\kappa'_{eg} = \kappa_{eg} \sqrt{\kappa(Q^{(k)})},$$

$$\kappa'_{eh} = \kappa_{eh}\kappa(Q^{(k)}),$$

the model function $m_f(x) = \hat{m}_f(T(x;\delta_r))$ *satisfies the following error bounds for all* $x \in T^{-1}\left(B_2(0;\delta_r)\right) = \hat{E}^{(k)}$*:*

$$|m(x) - f(x)| \leq \kappa'_{ef}\delta_r^3, \tag{41}$$

$$\|\nabla m(x) - \nabla f(x)\| \leq \kappa'_{eg}\delta_r^2, \tag{42}$$

$$\|\nabla^2 m(x) - \nabla^2 f(x)\| \leq \kappa'_{eh}\delta_r. \tag{43}$$

Proof:

Noting that $D_{i,i} > 0$, observe that

$$\delta_r^2 = \frac{\delta_k^2}{\lambda_{\min}\left(Q^{(k)}\right)} = \frac{\delta_k^2}{\min_{i\in[n]} D_{i,i}^2},$$

so $\min_i \Delta_{i,i} = \frac{\delta_k^2}{\delta_r^2}$. Thus,

$$\kappa\left(Q^{(k)}\right) = \kappa\left(D^2\right) = \frac{\max_i D_{i,i}^2}{\min_i D_{i,i}^2} = \frac{\delta_r^2}{\delta_k^2}\max_i D_{i,i}^2 = \frac{\delta_r^2}{\delta_k^2}\|D\|^2$$
$$\implies \frac{\delta_r}{\delta_k}\|D\| = \sqrt{\kappa\left(Q^{(k)}\right)}.$$

Let $x \in E^{(k)}$ be arbitrary and let $u = T(x) \in B_2(0;\delta_r)$. Then

$$|m_f(x) - f(x)| = |\hat{m}(u) - \hat{f}(u)| \leq \kappa'_{ef}\delta_r^3.$$

Similarly, for the gradient we find:

$$\begin{aligned}\|\nabla m_f(x) - \nabla f(x)\| &= \left\|\frac{\delta_r}{\delta_k}DL^T\left(\nabla\hat{m}_f(u) - \nabla\hat{f}(u)\right)\right\| \\ &\leq \frac{\delta_r}{\delta_k}\|DL^T\|\|\nabla\hat{m}_f(u) - \nabla\hat{f}(u)\| \\ &\leq \sqrt{\kappa(Q^{(k)})}\kappa_{eg}\delta_r^2 = \kappa'_{eg}\delta_r^2.\end{aligned}$$

Finally, for the Hessian, we have

$$\begin{aligned}\|\nabla^2 m_f(x) - \nabla^2 f(x)\| &= \left\|\frac{\delta_r^2}{\delta_k^2}DL^T\left(\nabla\hat{m}_f(u) - \nabla\hat{f}(u)\right)LD^T\right\| \\ &\leq \frac{\delta_r^2}{\delta_k^2}\|D\|^2\|\nabla\hat{m}_f(u) - \nabla\hat{f}(u)\| \\ &\leq \kappa(Q^{(k)})\kappa_{eh}\delta_r = \kappa'_{eh}\delta_r.\end{aligned}$$

□

Lemma 7.2 shows that a uniform bound on the condition number of $Q^{(k)}$ will produce accurate model functions. Indeed, recall from Theorem 4.10 that Algorithm 1 produces a Λ-poised sample set, for $B_2(0;\delta)$. Then Lemma 4.5 and Theorem 4.6 ensure m_f will satisfy the provided error bounds over the ellipsoidal region as well. We can now satisfy the accuracy condition (37).

Theorem 7.3 *Suppose that Assumption 7.1, Assumption 7.4, and Assumption 7.3 hold. Suppose further that for each iterate k, the ellipsoids $E^{(k)}$ and $\hat{E}^{(k)}$ provided by the* ConstructTrustRegion *subroutine are suitable according to Definition 6.1, and the sample set is constructed by calling Algorithm 3.*

Then, $m_f\left(x^{(k)}\right) = f\left(x^{(k)}\right)$, and the accuracy condition (37) *is satisfied for each iterate k. Namely, there exists $\kappa_g > 0$ such that*

$$\left\|\nabla m_f\left(x^{(k)}\right) - \nabla f\left(x^{(k)}\right)\right\| \leq \kappa_g \Delta_k \quad \forall k \in \mathbb{N}.$$

Proof:

Fix an iterate k. The assumptions provide two ellipsoids:

- $E^{(k)} = \left\{ x \in \mathbb{R}^n \,\middle|\, \left(x - c^{(k)}\right)^T Q^{(k)} \left(x - c^{(k)}\right) \leq \frac{1}{2}{\delta_k}^2 \right\}$ with $E^{(k)} \subset \mathcal{P}^{(k)}$.
- $\hat{E}^{(k)} = \left\{ x \in \mathbb{R}^n \,\middle|\, \left(x - c^{(k)}\right)^T Q^{(k)} \left(x - c^{(k)}\right) \leq {\delta_k}^2 \right\}$ with $x^{(k)} \in \hat{E}^{(k)}$.

As in Lemma 7.2, we can give $Q^{(k)} = LD^2L^T$ its eigen-decomposition, and for any $\delta > 0$ define the transformation $T(x;\delta) = \frac{\delta}{\delta_k^2} DL^T\left(x - c^{(k)}\right)$. Notice that with $\delta = \delta_2 = \sqrt{2}$, the transformation $T\left(x;\delta_2\right)$ maps $T^{(k)}_{\text{sample}} = E^{(k)}$ to the unit ball. After using Algorithm 1 to choose sample points, we know by Theorem 4.10 that there is a bound $\Lambda > 0$ with $\left\|M\left(\bar{\Phi}, \hat{Y}\right)^{-1}\right\| \leq \Lambda$ depending only on p and $\xi_{\min}$. Next, we consider the shifted sample set $\bar{Y} = \sqrt{2}\frac{\delta_r}{\delta_2}\hat{Y}$, where $\delta_r = \max_{x \in \hat{E}^{(k)}} \|x - c^{(k)}\|$. Notice that the scaled matrix in (13) is $M\left(\bar{\Phi}, \hat{Y}\right) = \hat{M}\left(\bar{\Phi}, \bar{Y}\right)$, so we can use Theorem 4.7 to introduce constants $\kappa_f, \kappa_g, \kappa_h > 0$ such that

$$\begin{aligned}
\left|\hat{f}\left(u\right) - \hat{m}_f\left(u\right)\right| &\leq \kappa_f \Lambda \delta_r^3, \\
\left\|\nabla \hat{f}\left(u\right) - \nabla \hat{m}_f\left(u\right)\right\| &\leq \kappa_g \Lambda \delta_r^2, \\
\left\|\nabla^2 \hat{f}\left(u\right) - \nabla^2 \hat{m}_f\left(u\right)\right\| &\leq \kappa_h \Lambda \delta_r.
\end{aligned}$$

for all $u \in B_2\left(0, \delta_r\right)$. This means that the shifted functions $\hat{m}_f(u) = m_f(T^{-1}(u);\delta)$ and $\hat{f}(u) = f(T^{-1}(u;\delta))$ as described in Lemma 7.2, satisfy (38)—(40), and therefore (41)—(43).

Because $\kappa\left(Q^{(k)}\right)$ is bounded by some $\epsilon_\alpha > 0$ independently of k, and defining $\kappa_g' = \kappa_g\sqrt{\epsilon_\alpha}\Lambda\Delta_{\max}$, we see that we can use Lemma 7.2 to conclude that for all $x_0 \in \hat{E}^{(k)}$:

$$\left\|\nabla f\left(x_0\right) - \nabla m_f\left(x_0\right)\right\| \leq \kappa_g \Lambda \Delta_k^2 \sqrt{\kappa\left(\frac{2}{\delta_k} Q^{(k)}\right)} = \kappa_g\sqrt{\epsilon_\alpha}\Lambda\Delta_{\max}\Delta_k = \kappa_g'\Delta_k.$$

In particular, $x^{(k)} \in \hat{E}^{(k)}$. □

8 Ellipsoidal Trust Region Approach

The main idea of the ellipsoidal method is to define the sample trust region to be a feasible ellipsoid as in (30):

$$T^{(k)}_{\text{sample}} = E^{(k)} = \left\{ x \in \mathbb{R}^n \middle| (x - c^{(k)})^T Q^{(k)} (x - c^{(k)}) \leq \frac{1}{2}\delta_k^2 \right\}$$

where $Q^{(k)}$ is a positive definite matrix, $c^{(k)}$ is the center of the ellipsoid, and δ_k is a constant determining the ellipsoid's radius. $Q^{(k)}$ and $c^{(k)}$ are chosen so that the ellipsoid conforms roughly to the shape of the feasible region near the current iterate, while ensuring that it lies entirely within the intersection of the feasible region and the outer trust region.

8.1 A safe ellipsoid

8.1.1 Construction

This section shows how to construct one possible ellipsoid that satisfies Definition 6.1. We define the active constraints at a point $x \in \mathcal{F}$ by

$$\mathcal{A}(x) = \{ i \in [m] | A_i x = b_i \}. \tag{44}$$

Recall that A and b are defined by (29).

For any iterate k, if there are no active constraints, $\mathcal{A}\left(x^{(k)}\right) = \emptyset$, then we are free to choose

$$Q^{(k)} = I, \quad c^{(k)} = x^{(k)}, \quad \text{and} \quad \delta_k = \min \left\{ \Delta_k, \min_{i \in [m]} b_i - A_i x \right\}. \tag{45}$$

We show within Lemma 8.1 that this is a suitable ellipsoid. Otherwise, we begin by constructing a direction $\hat{u}^{(k)}$ that is maximally feasible with respect to the active constraints. To make this precise, let $\mathcal{S} \subseteq \{1, \ldots, m\}$ be arbitrary. We define the set of "most" feasible direction from x, and quantify how feasible they are with the definitions:

$$u_{\text{feasible}}(\mathcal{S}) = \begin{cases} \arg\max_{\|u\|=1} \min_{i \in \mathcal{S}} -u^T A_i & \text{if } \mathcal{S} \neq \emptyset \\ \emptyset & \text{if } \mathcal{S} = \emptyset \end{cases} \tag{46}$$

$$\pi_1(\mathcal{S}) = \begin{cases} \max_{\|u\|=1} \min_{i \in \mathcal{S}} -u^T A_i & \text{if } \mathcal{S} \neq \emptyset \\ 1 & \text{if } \mathcal{S} = \emptyset \end{cases} \tag{47}$$

$$\pi_2(x) = \pi_1\left(\mathcal{A}(x)\right). \tag{48}$$

Here, u_{feasible} is a set of directions that head away from each of the active constraints at a point. For each iterate, this produces the quantities

$$\pi^{(k)} = \pi_2\left(x^{(k)}\right), \tag{49}$$

$$\hat{u}^{(k)} \in u_{\text{feasible}}\left(\mathcal{A}\left(x^{(k)}\right)\right). \tag{50}$$

We then compute

$$\pi_3^{(k)} = \left(1 + \frac{1}{\sqrt{2}}\right)\sqrt{1 + \left(\pi^{(k)}\right)^2}, \tag{51}$$

$$\delta_f = \frac{1}{\pi_3^{(k)}} \min\left\{\Delta_k, \min_{i \in [m] \setminus \mathcal{A}\left(x^{(k)}\right)} \left[b_i - \left(A_i\right)^T x^{(k)}\right]\right\}. \tag{52}$$

Notice that the minimization in the above expression computes the minimum distance to a non-active constraint. It will be convenient to define the rotation matrix and affine mapping

$$R^{(k+1)} = 2\frac{(e_1 + \hat{u}^{(k)})(e_1 + \hat{u}^{(k)})^T}{(e_1 + \hat{u}^{(k)})^T(e_1 + \hat{u}^{(k)})} - \boldsymbol{I}, \tag{53}$$

$$T_k(x) = R^{(k+1)}\left(x - x^{(k)}\right). \tag{54}$$

Recall that $e_1 = (1, 0, \ldots, 0)^T$, and observe that $R^{(k+1)}e_1 = \hat{u}^{(k)}$, $R^{(k+1)}\hat{u}^{(k)} = e_1$, $\det(R^{(k+1)}) = 1$, and $R^{(k+1)}{R^{(k+1)}}^T = {R^{(k+1)}}^T R^{(k+1)} = I$.

We can then define the ellipsoid as

$$Q^{(k)} = {R^{(k+1)}}^T \begin{pmatrix} 1 & \mathbf{0}^T \\ \mathbf{0} & \left(\pi^{(k)}\right)^{-2}\boldsymbol{I} \end{pmatrix} R^{(k+1)}, \quad c^{(k)} = x^{(k)} + \delta_f \hat{u}^{(k)}, \quad \text{and} \quad \delta_k = \delta_f \tag{55}$$

whenever $\mathcal{A}\left(x^{(k)}\right) \neq \emptyset$.

8.1.2 Suitability

We show that this construction provides a suitable ellipsoid according to Definition 6.1.

Lemma 8.1 *Let $\mathcal{A}$ be defined by* (44).

If $\mathcal{A}\left(x^{(k)}\right) = \emptyset$ during iteration k, then (45) *defines a suitable ellipsoid for iteration k according to Definition 6.1.*

Proof:

If $\mathcal{A}\left(x^{(k)}\right) = \emptyset$, then we are free to use (45). This simplifies $\hat{E}^{(k)}$ to

$$\left(x - c^{(k)}\right)^2 Q^{(k)}\left(x - c^{(k)}\right)^2 \leq \delta_k^2 \Longrightarrow \left\|x - x^{(k)}\right\|^2 \leq \Delta_k^2.$$

Because $E^{(k)}$ is then a sphere within the outer trust region with radius less than the distance to the nearest constraint, $E^{(k)} \subseteq \mathcal{P}^{(k)}$. Because the sphere $\hat{E}^{(k)}$ is centered at $x^{(k)}$, $x^{(k)} \in \hat{E}^{(k)}$. Also, $\kappa\left(Q^{(k)}\right) = 1$.
□

Lemma 8.2 *Let π_2 be defined by* (48).

Suppose that Assumption 7.3 holds.

Then, $1 \geq \pi_2(y) > 0$ for any $y \in \mathcal{F}$.

Proof:

Let $y \in \mathcal{F}$, and let $\mathcal{A}(x)$ be defined by (44). If $\mathcal{A}(y) = \emptyset$, then $\pi_2(y) = 1 > 0$. Otherwise, let $i \in \mathcal{A}(y)$, so that by (29), $c_i(y)(Ay - b)_i = 0$. We know that if $\bar{x}$ is defined as in Assumption 7.3, then

$$c_i(\bar{x}) = c_i(y) + A_i(\bar{x} - y) \Longrightarrow A_i(\bar{x} - y) \leq c_i(\bar{x}) - c_i(y) = c_i(\bar{x}) < 0.$$

Using (31), we can write this as

$$-A_i \frac{\bar{x} - y}{\|\bar{x} - y\|} > 0 \Longrightarrow \min_{i \in \mathcal{A}(y)} -A_i \frac{\bar{x} - y}{\|\bar{x} - y\|} > 0.$$

Using this along with the definitions of u_{feasible}, π_1, and π_2 in (46), (47), and (48); we see

$$\pi_2(y) = \pi_1\left(\mathcal{A}(y)\right) = \max_{\|u\|=1} \min_{i \in \mathcal{A}(y)} -A_i^T u \geq \min_{i \in \mathcal{A}(y)} -A_i^T \frac{\bar{x} - y}{\|\bar{x} - y\|} > 0.$$

We know that $\pi_2(y) \leq 1$ because it is the dot product of two vectors of length one: if $\|u\| = 1$, then $\left|u^T A_i\right|^2 \leq \|u\|\|A_i\| = 1$ by Cauchy–Schwartz.
□

Lemma 8.3 *Let $\pi^{(k)}$ be defined by* (49).
Suppose that Assumption 7.3 holds.
There exists an $\epsilon_\alpha > 0$ such that $\pi^{(k)} \geq \epsilon_\alpha \; \forall \; k \in \mathbb{N}$.

Proof:

Let $\mathcal{A}$ be defined by (44). Because there are m constraints, each $\mathcal{A}(x)$ is one of the 2^m subsets of $\{1, 2, 3, \ldots, m\}$. This means that u_{feasible}, π_1, and $\pi^{(k)}$ as defined by (46), (47), and (49) can only take on at most $1 + 2^m$ values. By Lemma 8.2, we know that each of these values must be positive. Thus, we are free to choose ϵ_α to be the smallest of these values. □

It will be useful to define some intermediate cones. Namely, for any scalar π, direction u, and point c, we define the cone

$$\mathcal{C}\left(\pi, u, c\right) = \left\{c + tu + s \in \mathbb{R}^n \middle| \; s^T u = 0, t \geq 0, \|s\| \leq \pi t\right\}. \tag{56}$$

We can use this to define shifted and unshifted cones about the point $x^{(k)}$ along a direction $\hat{u}^{(k)}$:

$$C_{\text{sh}}^{(k)} = \mathcal{C}\left(\pi^{(k)}, e_1, 0\right), \tag{57}$$

$$C_{\text{unsh}}^{(k)} = \mathcal{C}\left(\pi^{(k)}, \hat{u}^{(k)}, x^{(k)}\right). \tag{58}$$

Observe that, by construction, $C_{\text{unsh}}^{(k)}$ is feasible with respect to the active constraints. That is, $A_i x \leq b_i$ for all $x \in C_{\text{unsh}}^{(k)}$ and $i \in \mathcal{A}\left(x^{(k)}\right)$. Figure 1 contains a depiction of these cones.

Lemma 8.4 *Let $C_{unsh}^{(k)}$ and $\mathcal{P}^{(k)}$ be defined by* (58) *and* (31).
The set $C_{unsh}^{(k)}$ is feasible with respect to the active constraints of $\mathcal{P}^{(k)}$ at $x^{(k)}$.

Proof:

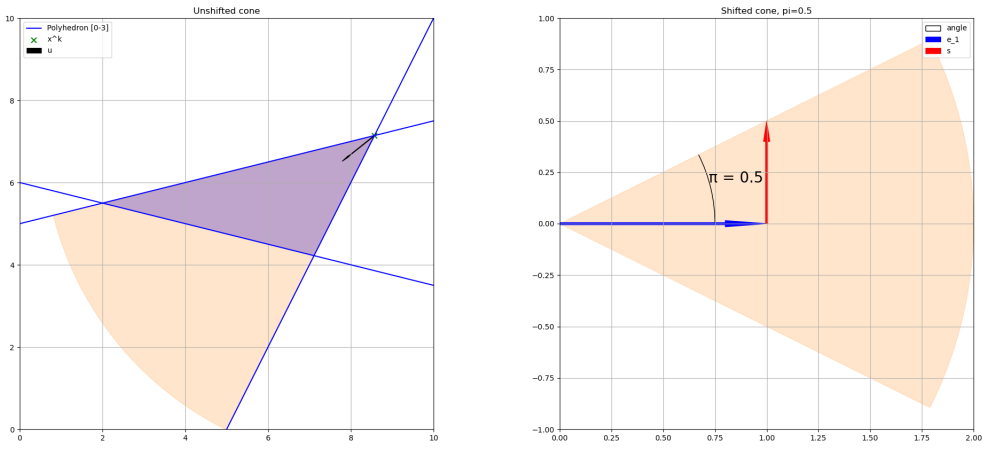

Figure 1: On the left is an unshifted cone $C^{(k)}_{\text{unsh}}$. The zeros of the polyhedron constraint functions are in blue, and the current iterate is in green. A feasible direction u from the current iterate is calculated, and the width of the cone is determined to lie within the active constraints of the polyhedra. On the right is a shifted cone $C^{(k)}_{\text{unsh}}$. It is centered at the origin, and points along e_1, opening at a rate of $\pi^{(k)}$.

Note that the trust region boundary cannot be active at $x^{(k)}$ as $\Delta_k > 0$. Let $\mathcal{A}$, $\pi^{(k)}$, and $\hat{u}^{(k)}$ be defined by (44), (49), (50) and A and b be defined by (29). Let $y = x^{(k)} + t\hat{u}^{(k)} + s \in C^{(k)}_{\text{unsh}}$ and $i \in \mathcal{A}\left(x^{(k)}\right)$ be arbitrary. Then,

$$A_i^T y - b_i = A_i^T(t\hat{u}^{(k)} + s) = A_i^T s + tA_i^T\hat{u}^{(k)} \leq \|s\| - \pi^{(k)}t \leq 0.$$

□

The following function is useful for showing results about our ellipsoids:

$$f_e(\pi, \delta, r; x) = (x - \delta e_1)^T \begin{pmatrix} 1 & \mathbf{0}^T \\ \mathbf{0} & \pi^{-2}\boldsymbol{I} \end{pmatrix} (x - \delta e_1) - r. \tag{59}$$

Lemma 8.5 *Let $C^{(k)}_{sh}, f_e, \pi^{(k)}$ be defined as in* (57), (59), (49). *Then, for all $\delta > 0$, the ellipsoid*

$$\left\{ x \in \mathbb{R}^n \middle| f_e\left(\pi^{(k)}, \delta, \frac{1}{2}\delta^2; x\right) \leq 0 \right\} \subseteq C^{(k)}_{sh}. \tag{60}$$

Proof:

Suppose that $x \in \left\{ x \in \mathbb{R}^n \middle| f_e\left(\pi^{(k)}, \delta, \frac{1}{2}\delta^2; x\right) \leq 0 \right\}$, and let $t = x^T e_1$, $s = x - te_1$. Then

$$f_e(x) \leq 0 \implies (x - \delta e_1)^T \begin{pmatrix} 1 & \mathbf{0}^T \\ \mathbf{0} & \left(\pi^{(k)}\right)^{-2}\boldsymbol{I} \end{pmatrix} (x - \delta e_1) \leq \frac{1}{2}\delta^2$$

$$\implies (t-\delta)^2 + \frac{1}{\left(\pi^{(k)}\right)^2}\|s\|^2 \leq \frac{1}{2}\delta^2$$
$$\implies \|s\|^2 \leq \left(\pi^{(k)}\right)^2\left[\frac{1}{2}\delta^2 - (t-\delta)^2\right]$$
$$= \left(\pi^{(k)}\right)^2\left[t^2 - 2(t-\frac{1}{2}\delta)^2\right] \leq \left(\pi^{(k)}\right)^2 t^2$$
$$\implies \|s\| \leq \pi^{(k)}t.$$

Thus, $x \in C_{\text{sh}}^{(k)}$. □

Lemma 8.6 *Let $R^{(k+1)}$, T_k, $C_{unsh}^{(k)}$, $C_{sh}^{(k)}$ be defined as in* (53)*,* (54)*,* (58)*,* (57)*. Then $T_k(C_{unsh}^{(k)}) = C_{sh}^{(k)}$.*

Proof:

Observe that $R^{(k+1)}e_1 = \hat{u}^{(k)}$, $R^{(k+1)}\hat{u}^{(k)} = e_1$, $\det(R^{(k+1)}) = 1$, and $R^{(k+1)}{R^{(k+1)}}^T = {R^{(k+1)}}^T R^{(k+1)} = I$.

Suppose that $x \in C_{\text{unsh}}^{(k)}$. Then there exists $t \geq 0$ and $s \in \mathbb{R}^n$ such that $x = x^{(k)} + t\hat{u}^{(k)} + s$ where $s^T\hat{u}^{(k)} = 0$ and $\|s\| \leq \pi^{(k)}t$. Then $T_k(x) = tR^{(k+1)}\hat{u}^{(k)} + R^{(k+1)}s = te_1 + R^{(k+1)}s$. Observe that $(Rs)_1 = (R^{(k+1)}s)^T e_1 = s^T{R^{(k+1)}}^T(R^{(k+1)}\hat{u}^{(k)}) = s^T\hat{u}^{(k)} = 0$. Hence, $T_k(x) = \begin{pmatrix} t \\ \sigma \end{pmatrix}$ where $\sigma \in \mathbb{R}^{n-1}$ satisfies $\|\sigma\| = \|s\| \leq \pi^{(k)}t$. Thus, $T_k(x) \in C_{\text{sh}}^{(k)}$. Conversely, if $\begin{pmatrix} t \\ \sigma \end{pmatrix} \in C_{\text{sh}}^{(k)}$, then let $s = {R^{(k+1)}}^T\begin{pmatrix} 0 \\ \sigma \end{pmatrix}$ to see that $x = T_k^{-1}\left(\begin{pmatrix} t \\ \sigma \end{pmatrix}\right) = {R^{(k+1)}}^T\left(te_1 + \begin{pmatrix} 0 \\ \sigma \end{pmatrix}\right) = t\hat{u}^{(k)} + s$, where $\|s\| = \|\sigma\| \leq \pi^{(k)}t$. Hence $T_k^{-1}\left(\begin{pmatrix} t \\ \sigma \end{pmatrix}\right) \in C_{\text{unsh}}^{(k)}$. □

Lemma 8.7 *Let $\mathcal{A}$, f_e, δ_f $\pi^{(k)}$, $R^{(k+1)}$, T_k, and $\mathcal{P}^{(k)}$ be defined by* (44)*,* (59)*,* (52)*,* (49)*,* (53)*,* (57)*, and* (31)*, respectively.*

For each iteration k, if $\mathcal{A}\left(x^{(k)}\right) \neq \emptyset$, the ellipsoid

$$E^{(k)} = \left\{ x \in \mathbb{R}^n \middle| f_e\left(\pi^{(k)}, \delta_f, \frac{1}{2}\delta_f^2; T_k(x)\right) \leq 0 \right\} \tag{61}$$

satisfies $E^{(k)} \subseteq \mathcal{P}^{(k)}$.

Proof:

Let $\pi_3^{(k)}$, $R^{(k+1)}$, $C_{\text{unsh}}^{(k)}$, and $C_{\text{sh}}^{(k)}$ be defined by (51), (53), (58), and (57), respectively. We see that if $x \in E^{(k)}$, then by Lemma 8.5 we have that $T_k(x) = \begin{pmatrix} t \\ \sigma \end{pmatrix} \in C_{\text{sh}}^{(k)}$ for some $\sigma \in \mathbb{R}^{n-1}$ with $\|\sigma\| \leq \pi^{(k)}$, and

$$(x - \delta_f e_1)^T \begin{pmatrix} 1 & \mathbf{0}^T \\ \mathbf{0} & \left(\pi^{(k)}\right)^{-2}\boldsymbol{I} \end{pmatrix}(x - \delta_f e_1) \leq \frac{1}{2}\delta_f^2$$

$$\implies (t-\delta_f)^2 \leq (t-\delta_f)^2 + \frac{1}{\left(\pi^{(k)}\right)^2}\|\sigma\|^2 \leq \frac{1}{2}\delta_f^2$$
$$\implies t \leq \left(1+\frac{1}{\sqrt{2}}\right)\delta_f$$

so that

$$\|x\|^2 = t^2 + \|\sigma\|^2 \leq \left(1+\left(\pi^{(k)}\right)^2\right)t^2 \leq \left(1+\left(\pi^{(k)}\right)^2\right)\left(1+\frac{1}{\sqrt{2}}\right)^2\delta_f^2 = \left(\pi_3^{(k)}\right)^2\delta_f^2$$
$$\implies \|x\| \leq \pi_3^{(k)}\delta_f \leq \min_{i\in[m]\setminus\mathcal{A}\left(x^{(k)}\right)}\left[b_i - \left(A_i\right)^T x^{(k)}\right].$$

Thus, all points within $E^{(k)}$ are closer than the nearest point of a non-active constraint. Combine this with Lemma 8.4 to see that $E^{(k)} \subseteq \mathcal{P}^{(k)}$. □

Lemma 8.8 *Let δ_f, $R^{(k+1)}$, T_k, $C_{unsh}^{(k)}$, $C_{sh}^{(k)}$, $\mathcal{P}^{(k)}$ be defined as in* (52), (53), (54), (58), (57), (31).
For any iteration k, the ellipsoid

$$\hat{E}^{(k)} = \left\{x \in \mathbb{R}^n | f_e\left(\pi^{(k)}, \delta_k, \delta_k^2; T_k(x)\right) \leq 0\right\} \tag{62}$$

satisfies $x^{(k)} \in \hat{E}^{(k)}$.

Proof:

We have that

$$f_e\left(\pi^{(k)}, \delta_f, \delta_f^2; T_k\left(x^{(k)}\right)\right) = f_e\left(\pi^{(k)}, \delta_f, \delta_f^2; 0\right) =$$
$$\left(0-\delta_f e_1\right)^T \begin{pmatrix} 1 & \mathbf{0}^T \\ \mathbf{0} & \left(\pi^{(k)}\right)^{-2}\boldsymbol{I} \end{pmatrix}\left(0-\delta_f e_1\right) = \delta_f^2 \leq \delta_f^2.$$

□

Lemma 8.9 *Let $\mathcal{A}$ and $\pi^{(k)}$ be defined by* (44) *and* (49)*, respectively.*
Suppose that Assumption 7.3 holds.
For some iteration k, if $\mathcal{A} \neq \emptyset$, then the ellipsoid defined by (55) *is suitable according to Definition 6.1.*

Proof:

Let f_e, T_k, $R^{(k+1)}$, δ_f, and $\hat{\mathcal{E}}^k_{\text{feasible}}$ be defined as in (59), (54), (53), (61), and (62), respectively. Observe that Definition 6.1 with (55) defines $E^{(k)}$ and $\hat{E}^{(k)}$ to be

$$E^{(k)} = \left\{x \in \mathbb{R}^n \middle| f_e\left(\pi^{(k)}, \delta_k, \tfrac{1}{2}\delta_k^2; T_k(x)\right) \leq 0\right\}, \qquad \text{and}$$
$$\hat{E}^{(k)} = \left\{x \in \mathbb{R}^n \middle| f_e\left(\pi^{(k)}, \delta_k, \delta_k^2; T_k(x)\right) \leq 0\right\}.$$

By Lemma 8.7, we know that $E^{(k)} \subseteq \mathcal{P}^{(k)}$. By Lemma 8.8, we know that $x^{(k)} \in \hat{E}^{(k)}$. By Lemma 8.3, there exists $\epsilon_\alpha > 0$, such that the condition number $\kappa\left(Q^{(k)}\right) = \frac{\max\{1,\pi^{(k)-2}\}}{\min\{1,\pi^{(k)-2}\}} = \pi^{(k)-2} > 0$. This is because $\det\left(R^{(k+1)}\right) = 1$ means the condition number of $Q^{(k)}$ is not affected $R^{(k+1)}$. □

8.2 Ellipsoid searches

Within Section 8.1.1, we showed how to construct one possible ellipsoid that satisfies Definition 6.1. In practice, this ellipsoid could be less than desirable. Within this section, we discuss the requirements of other variants of the *ConstructTrustRegion* subroutine that we explored for practicality. The key requirement is these ellipsoids must still satisfy Definition 6.1 to share the same convergence results.

8.2.1 Ensuring suitability

To retain the convergence results, these algorithms maintain $E^{(k)} \subseteq \mathcal{P}^{(k)}$, $x^{(k)} \in \hat{E}^{(k)}$, and a bound on $\kappa\left(Q^{(k)}\right)$.

To ensure the condition number is bounded, the algorithm can compute the condition number of the safe ellipsoid, and only consider ellipsoids better conditioned. Alternatively, it could introduce its own bound. We found it simplest to parameterize ellipsoids by their Cholesky factorization $Q^{(k)} = LL^T$. Namely, we parameterized the search space in terms of a lower triangular matrix L, and required the diagonal entries to be positive. To ensure a bounded condition number, we chose a $\epsilon_\alpha > 0$, and constrain $\max_{i\in[n]} \leq \epsilon_\alpha \min_{i\in[n]}$.

One potential difficulty created by moving the ellipsoid center $c^{(k)}$ is that the current iterate $x^{(k)}$ may not lie within near the resulting ellipsoid. The pitfall is that the model function may lose accuracy near the current iterate. Thus, we have implemented a few ways of ensuring the current iterate is within the search trust region. This can be done by either of the following two options:

- Adding a constraint to the ellipsoid problem to include the original point.
- Expanding the size of the ellipsoid.

Adding a constraint. In order to include the original point as a constraint, we add a constraint of the following form to the definition of the ellipsoid.

$$\frac{1}{2}\left(x^{(k)} - c^{(k)}\right)^T Q^{(k)} \left(x^{(k)} - c^{(k)}\right) \leq \frac{1}{2}\delta_k^2.$$

Constraints of this nature make finding the ellipsoid much more expensive: the optimization problem we construct uses ${Q^{(k)}}^{-1}$ as decision variables, so that constraints in terms of $Q^{(k)}$ must model matrix inversion.

Increasing the radius. An alternative is to scale $Q^{(k)}$ by a constant. We use a scaling factor $\delta_k > 0$ defined by

$$\delta_k = \max\left\{1, \sqrt{\left(x^{(k)} - c^{(k)}\right)^T Q^{(k)} \left(x^{(k)} - c^{(k)}\right)}\right\}$$

and let the ellipsoid be:

$$E^{(k)} = \left\{x \in \mathbb{R}^n \middle| \frac{1}{2}\left(x - c^{(k)}\right)^T Q^{(k)} \left(x - c^{(k)}\right) \leq \frac{1}{2}\delta_k^2\right\}.$$

However, this means that in general $E^{(k)} \not\subset \mathcal{F}$. so that the sample points must be contained to also lie within the feasible region: $E^{(k)} \cap \mathcal{F}_m^{(k)}$. For details on how to choose sample points with additional constraints, see Algorithm 1.

8.2.2 Maximal volume ellipsoids

The error bounds given in Lemma 7.2 suggest that we can obtain more accurate model functions by minimizing the condition number of the matrix $Q^{(k)}$. However, we also desire a large ellipsoid so that our model will satisfy the error bounds over more of $T^{(k)}_{\text{search}}$. Thus, one choice for $T^{(k)}_{\text{sample}}$ is to choose the maximum volume ellipsoid that is both feasible and lies within the outer trust region. We can accomplish this for various ellipsoid centers, by finding the maximal volume ellipsoid that is constrained to lie within the polytope

$$\mathcal{P}^{(k)} := \left\{ x \in \mathbb{R}^n | Ax \leq b, x_i^{(k)} - \Delta_k \leq x_i \leq x_i^{(k)} + \Delta_k \right\}.$$

This is not the only reasonable approach: maximizing ensures a larger region for which the models will be accurate, but we are most interested in descent directions. Namely, we may wish to use previously evaluated points to provide a hint of where the next ellipsoid should be. Another consideration is where points have been evaluated: it may be more economical to choose a smaller ellipsoid that can reuse existing points. However, in this section, we consider the problem of choosing $Q^{(k)}$ and δ_k to maximize the volume of $E^{(k)} \subseteq \mathcal{P}^{(k)}$ given a fixed center $c^{(k)}$. Later, in Section 10.1.2, we will explore strategies for moving the center of the ellipsoid to improve the performance of our trust region algorithm.

We adopt a method similar to that described in [76], which presents an algorithm for finding the maximum volume ellipsoid inscribed in a given polytope.

Let $\bar{g} = g^{(k)} - G^{(k)}c^{(k)}$ and $d = x - c^{(k)}$ so that the polytope becomes

$$\mathcal{P}^{(k)} = \left\{ c^{(k)} + d \in \mathbb{R}^n \mid G^{(k)}d \leq \bar{g} \right\}.$$

Using this transformation, the ellipsoid can then be centered at zero, and defined by a symmetric positive definite matrix $Q \succ 0$:

$$E = \left\{ d \in \mathbb{R}^n \;\middle|\; \frac{1}{2} d^T Q d \leq 1 \right\}.$$

Our goal is to determine the matrix Q that maximizes the volume of E such that $\mu + E \subset \mathcal{P}$. This is accomplished by defining $\bar{b} = b - Ac$ and solving the following problem for Q for a given center:

$$\begin{array}{cc} \sup_{Q \succeq 0} & \det(Q^{-1}) \\ \text{s.t.} & A_i^T Q^{-1} A_i \leq \frac{1}{2} \bar{b_i}^2. \end{array} \tag{63}$$

Theorem 8.10 *Let $\mathcal{P} = \{x \in \mathbb{R}^n | Ax \leq b\}$, where A is an $m \times n$ matrix, and $b \in \mathbb{R}^m$. Let $c \in \operatorname{int} \mathcal{P}$.*

Suppose that some positive definite, symmetric matrix Q solves (63)*, where $\bar{b} = b - Ac$. Then the ellipsoid $E = \{x \in \mathbb{R}^n | (x-c)^T Q (x-c) \leq 1\}$ has the maximum volume over all ellipsoids centered at c and contained in $\mathcal{P}$.*

Proof:

Define the auxiliary function $f(d) = \frac{1}{2}d^T Q d$ so that $E = \{d \in \mathbb{R}^n \mid f(d) \leq 1\}$. Because Q is positive definite, f has a unique minimum on each hyperplane $\{d \in \mathbb{R}^n | \ A_i d = b_i\}$. Let this minimum be $d^{(i)} = \arg\min_{A_i d = \bar{b}_i} f(d)$ for $i \in [m]$. By the first-order optimality conditions, there exists a $\lambda \in \mathbb{R}^m$ such that for each $i \in [m]$,

$$\nabla f(d^{(i)}) = Q d^{(i)} = \lambda_i A_i.$$

Since Q is invertible, we have $d^{(i)} = \lambda_i Q^{-1} A_i$. We also know that $A_i^T d^{(i)} = \bar{b_i}$, so

$$A_i^T \lambda_i Q^{-1} A_i = \bar{b_i} \Longrightarrow \lambda_i = \frac{\bar{b_i}}{A_i^T Q^{-1} A_i},$$

so that

$$d^{(i)} = \lambda_i Q^{-1} A_i = \frac{\bar{b_i}}{A_i^T Q^{-1} A_i} Q^{-1} A_i \quad \forall 1 \leq i \leq m.$$

Because $E \subset \mathcal{P}$, we also know that $f(d^{(i)}) \geq 1$ for each i. Thus,

$$\begin{aligned}
&\frac{1}{2}(d^{(i)})^T Q d^{(i)} \geq 1 \\
&\qquad \Longrightarrow \frac{1}{2}\left(\frac{\bar{b_i}}{A_i^T Q^{-1} A_i} Q^{-1} A_i\right)^T Q \frac{\bar{b_i}}{A_i^T Q^{-1} A_i} Q^{-1} A_i \geq 1 \\
&\qquad \Longrightarrow \frac{1}{2}\frac{1}{A_i^T Q^{-1} A_i} \bar{b_i} A_i^T Q^{-1} Q \frac{\bar{b_i}}{A_i^T Q^{-1} A_i} Q^{-1} A_i \geq 1 \\
&\qquad \Longrightarrow \frac{1}{2}\frac{1}{A_i^T Q^{-1} A_i} \frac{\bar{b_i}^2}{A_i^T Q^{-1} A_i} A_i^T Q^{-1} A_i \geq 1 \\
&\qquad \Longrightarrow \frac{1}{2}\frac{\bar{b_i}^2}{A_i^T Q^{-1} A_i} \geq 1 \\
&\qquad \Longrightarrow \frac{1}{2}\bar{b_i}^2 \geq A_i^T Q^{-1} A_i \\
&\qquad \Longrightarrow A_i^T Q^{-1} A_i \leq \frac{1}{2}\bar{b_i}^2.
\end{aligned}$$

Because the volume of the ellipsoid is proportional to the determinant of Q^{-1}, the maximal ellipsoid is defined by (63). □

Notice that the constraints for (63) can be simplified for the outer trust region's constraints. These take the form:

$$e_i^T \left(\frac{Q^{(k)}}{\frac{1}{2}\delta_k^2}\right)^{-1} e_i \leq \left[e_i^T \left(x^{(k)} - c^{(k)}\right) \pm \Delta_k\right]^2 \quad \forall i \in [n]$$

or

$$\left(\frac{Q^{(k)}}{\frac{1}{2}\delta_k^2}\right)^{-1}_{i,i} \leq \left[x_i^{(k)} - c_i^{(k)} \pm \Delta_k\right]^2 \quad \forall i \in [n]. \tag{64}$$

9 Polyhedral Trust Region Approach

One simple approach to ensuring that all sample points are feasible is to restrict them to lie within the intersection of the feasible region and the outer trust region. In particular, we define the sample and search trust regions by

$$T^{(k)}_{\text{sample}} = T^{(k)}_{\text{search}} = \mathcal{F} \cap T^{(k)}_{\text{out}}.$$

Note that by using an L_∞-ball for the outer trust region, both $T^{(k)}_{\text{sample}}$ and $T^{(k)}_{\text{search}}$ are polytopes.

The main challenge in implementing this approach is ensuring that the sample points chosen from within $T^{(k)}_{\text{sample}}$ are well-poised. To accomplish this, we modify the model improvement algorithm given by Algorithm 1. Recall that Algorithm 1 works on a shifted and scaled problem in which sample points are selected to lie within a unit ball. Thus, in Step 2 of that algorithm, each new sample point $\hat{y}$ is selected by $\hat{y} \in \arg\max_{t:\|t\|\leq 1} |u_i(t)|$, where the pivot polynomials u_i are constructed during the algorithm. We modify this step by choosing $\hat{y} \in \arg\max_{t \in T^{(k)}_{\text{sample}}} |u_i(t)|$.

The challenge lies in finding sufficiently poised sample points. Note that Algorithm 1 uses a parameter ξ_{min} as a lower bound of the pivot values of the Vandermonde matrix. For unconstrained problems, this approach could always find a pivot value for any $\xi_{\text{min}} \in (0, 1)$ because it optimizes over a sphere. However, when requiring points to live within $\mathcal{F} \cap T^{(k)}_{\text{out}}$, it can happen that even after replacing a point, we still have not satisfied this bound. In Figure 2, for some values of ξ_{min}, there is no point in $\mathcal{F} \cap T^{(k)}_{\text{out}}$ that will leave a sufficiently large pivot.

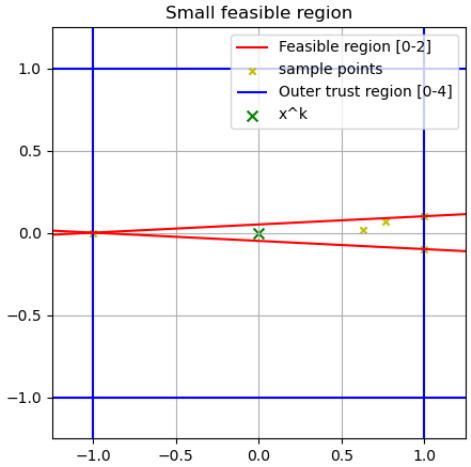

Figure 2: An example of constraints limiting sample point choices. If the constraints remove a large region of the trust region, there may be no feasible Λ-poised set for a given $\Lambda > 0$.

One way to handle this is to introduce a ξ_{cur} which is allowed to decrease. (Possibly, until a threshold is reached for maintaining a fixed Λ.) If the new point does not improve the geometry of the set significantly, then there is no other point that would do better. To test this, we introduce a constant $\delta_{\text{improv}} > 0$ and require

a new point to increase the current pivot by a factor greater than δ_{improv}. If the new point does not satisfy this test, we proceed with our current point and possibly decrease ξ_{cur}. Conceptually,

- ξ_{min} is a tolerance that ensures Step 3 does not perform division by zero,
- ξ_{cur} measures the most poised set possible within $T^{(k)}_{\text{sample}}$,
- and replacement points are ignored if they do not improve the poisedness by more than δ_{improv}.

The new modified improvement algorithm is described in Algorithm 4. The *ConstructTrustRegion* subroutine for this approach follows the prototype with $T^{(k)}_{\text{sample}} = T^{(k)}_{\text{search}} = \mathcal{F} \cap T^{(k)}_{\text{out}}$. As usual, we may also wish to remove points larger than a certain radius from the current model center.

Algorithm 4: Modified Model Improvement Algorithm

Step 0 (Initialization)
Given an interpolation set $T_{\text{sample}}^{(k)}$, and set Y of $p+1$ points, initialize $i = 1$, $0 < \xi_{\text{min}} < \xi_{\text{desired}}$, $0 < \delta_{\text{improv}} < 1$, $\xi_{\text{cur}} = \xi_{\text{desired}}$.

Step 1 (Pivot)
Compute the next pivot index $j_i^{\max} \in \arg\max_{i \le j \le |Y|-1} \left|u_i\left(y^j\right)\right|$, and swap points i and $j_i^{\max}$ within Y.

Step 2 (Check threshold) If $\left|u_i\left(y^i\right)\right| \ge \xi_{\text{cur}}$ then go to Step 3.
Compute $\hat{y} = \arg\max_{t \in T_{\text{sample}}^{(k)} \cap \mathcal{F}} |u_i(x)|$
If $|u_i(\hat{y})| < \xi_{\text{min}}$ then **Stop**: the algorithm failed
If $|u_i(\hat{y})| - \xi_{\text{cur}} \ge \delta_{\text{improv}} \xi_{\text{cur}}$ then replace point y^i with $\hat{y}$ and set $\xi_{\text{cur}} \leftarrow |\phi_i(\hat{y})|$

Step 3 (Gaussian elimination)
For $j = i+1, \ldots, p$:
Set $u_j(x) \leftarrow u_j(x) - \frac{u_j\left(y^i\right)}{u_i\left(y^i\right)} u_i(x)$
If $i = p$ then **Stop**, otherwise Set $i \leftarrow i+1$ and go to Step 1

When the algorithm fails, it is likely because $T_{\text{sample}}^{(k)}$ is nearly contained in a subspace for which some basis function can be written as a linear combination of other basis functions. Because our feasible region contains an interior point, we would expect that sufficiently small values of $\xi_{\text{min}} > 0$ allow the algorithm to run to completion. However, for non-linear constraints, or even equality-based constraints, there may not be such an ξ_{min}.

Intuitively, higher order monomials would not improve the accuracy of the model over $T_{\text{sample}}^{(k)}$ if their function values do not significantly differ from that of a linear combination of lower order monomials. This insight inspires a model improving algorithm that uses Gaussian elimination with *full pivoting* to select only those monomials *useful* for approximating functions over $T_{\text{sample}}^{(k)}$. When there is no replacement point or corresponding monomial in the basis that can be added to the existing set of Lagrange polynomials with a pivot larger than ξ_{min}, the algorithm would simply stop the Gaussian elimination. Note that the maximization over $T_{\text{sample}}^{(k)}$ to find a replacement point would need to maximize $|u_j|$ for several $i \le j \le p$ rather than simply i.

10 Results

10.1 Algorithm variants

Here, we present numerical results for our implementations of Algorithm 2. We begin by describing the different ways we implemented *ConstructTrustRegion*.

10.1.1 Circular trust region

The simplest approach to maintaining a feasible trust region is to set the inner trust region radius sufficiently small. Within the *ConstructTrustRegion* subroutine, this method sets the trust region radius to the distance

to the closest constraint: $T_{\text{out}}^{(k)} = B_2\left(x^{(k)}, \min\left\{\Delta_k, \min_i \frac{|A_i x^{(k)} - b_i|}{\|A_i\|}\right\}\right)$. In practice, this does not work well as the radius can become too small to allow adequate progress.

Two general strategies were considered for addressing this issue as illustrated in Figure 3. One option is to shift the center of the inner trust region as long as it remains within the outer trust region. The second option is to elongate the trust region along the nearest constraint as discussed in the next section. Of course, both of these can be done at the same time.

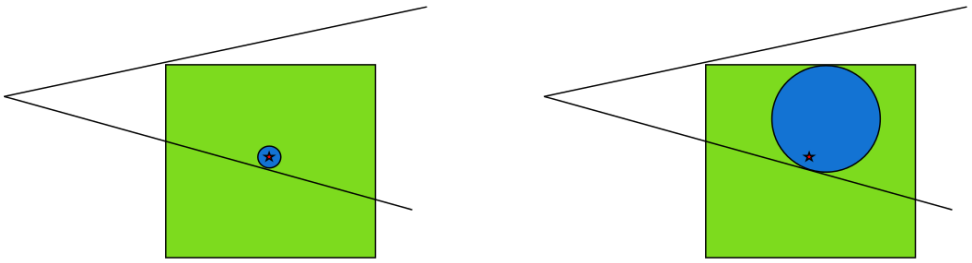

Figure 3: When the current iterate is too close to a constraint, the circular trust region becomes too small. Shifting the trust region center can help remedy this. The star is the current iterate, the outer trust region is in green, and the inner trust region is in blue.

In order to address this issue, we considered using ellipsoidal trust regions. Whereas the circle does not allow improvement when the current iterate lies along a constraint, an ellipsoid elongates along this constraint. In figure Figure 4, we have this type of iterate, but by using an ellipsoid we are still able to search towards the vertex of the feasible region.

More specifically, at iteration k, we choose a scaling factor δ_k and solve for an ellipsoid center $c^{(k)} \in \mathbb{R}^n$ and positive definite matrix $Q^{(k)}$ to define an ellipsoid

$$E^{(k)} = \left\{ x \in \mathbb{R}^n \,\middle|\, \frac{1}{2}\left(x - c^{(k)}\right)^T Q^{(k)} \left(x - c^{(k)}\right) \leq \frac{1}{2}\delta_k^2 \right\}.$$

The simplest approach is to simply let the center of the ellipsoid be the current iterate: $c^{(k)} = x^{(k)}$.

10.1.2 Choosing the ellipsoid center

The most obvious choice for the center of the ellipsoid is to choose $c^{(k)} = x^{(k)}$ (i.e., the current iterate). However, if $x^{(k)}$ is too close to a boundary of the feasible region, this can result in a badly shaped or very small ellipsoid. We, therefore, explore strategies where the *ConstructTrustRegion* subroutine moves the center of the ellipsoid away from the boundary. This is depicted in Figure 4.

Outer trust region search. One approach is to search all possible centers within $\mathcal{F} \cap T_{\text{out}}^{(k)}$.

This has the advantage that it allows for the largest volume. However, one problem with this search is that it can force the trust region away from the descent direction. Notice that in Figure 5, although the

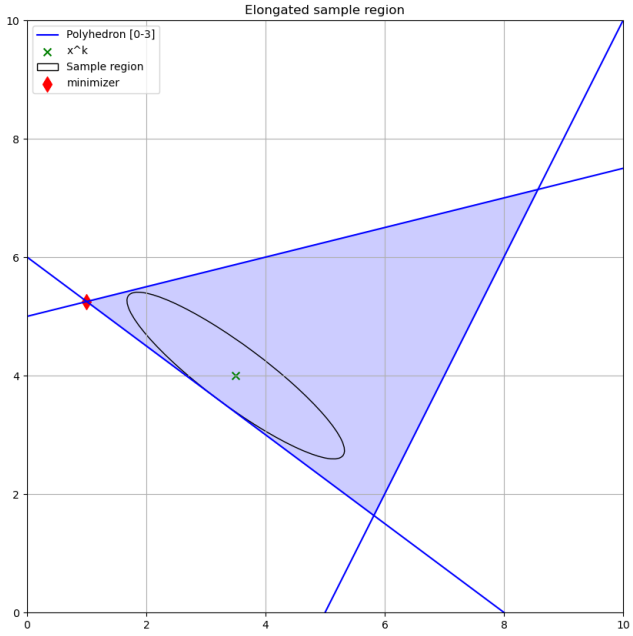

Figure 4: Although the center of the ellipsoid in green is close to the boundary of the blue constraints, it can elongate to allow progress.

ellipsoid found has larger volume than before being shifted, this ellipsoid contains points farther from the corner that happens to contain the trust region's minimizer. Within the numerical results, these algorithms are described as "ellipse everywhere".

The following section addresses this problem by proposing a path search method for choosing the ellipsoid center.

Path searches. Rather than searching over all possible centers, it may be more efficient to only move away from the boundary. This can be done using one-dimensional search along an appropriate direction. For example, our first attempt was to simply search a line directed orthogonally away from the closest constraint. This has obvious problems as shown in Figure 6: we should avoid letting the new center get closer to another constraint.

To fix this, we search along a piecewise linear path leading away from the closest constraints. The algorithm works by choosing a set of breakpoints $s_0, s_1, s_2, \ldots, s_{n_{\text{points}}}$ that are each equidistant to a subset of the constraint's faces. Intuitively, the search moves away from the nearest constraint until it reaches a point equidistant to the second nearest constraint, and so on. The center search then considers points along the line segments between these points.

More precisely, the first point is chosen to be the current iterate: $s_0 = x^{(k)}$. The algorithm then repeats

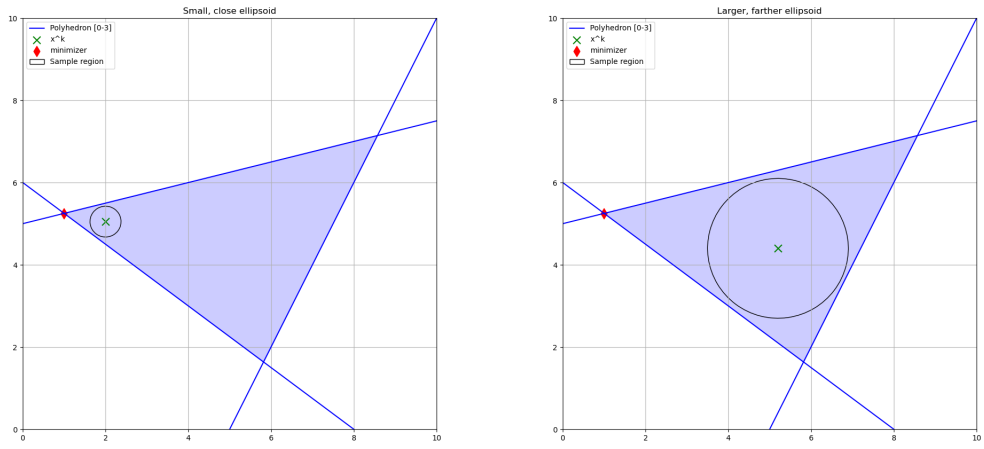

Figure 5: An example of how the search for the feasible region center can go wrong. On the left, a very small sample region is selected; however its proximity to the minimizer makes the model more accurate at the minimizer. On the right, a sample region with larger volume is chosen, but it is further from the trust region minimizer.

the following process for i from 0 to $n_{\text{points}} - 1$. First, compute the set of nearest constraints, where the distance from the current point s_i to each constraint A_j for $j \in [m]$: is given by $d_j = b - A_j x$. Recall that the rows of A are normalized: $\|A_j\| = 1 \quad \forall j \in [m]$. While finding the next point s_{i+1}, let E be the indices of A whose constraints are equidistant and nearest to s_i: $\{j \in [m] | d_j = \min_{l \in [m]} d_l\}$. Let the remaining indices be $R = [m] \setminus E$. The algorithm chooses a search direction $p = {A_E}^T r$ as a linear combination of the normal vectors of the nearest constraints. This search ray r can be found by computing a point s_i whose distance to each equidistant constraint is twice its current value:

$$b_E - A_E(s_i + {A_E}^T r) = 2\left[b_E - A_E s_i\right] \Longrightarrow r = \left[A_E {A_E}^T\right]^{-1} \left(b_E + A_E s_i\right).$$

We can travel along this ray until we reach a point that is the same distance to a remaining constraint. By travelling a distance t, we see that the j-th constraint becomes active when $A_j(s_i + tp) = b_j \Longrightarrow t = \frac{b_j - A_j s_i}{A_j p}$ so that we can travel by $t = \min_{j \in R} \left\{ \frac{b_j - A_j s_i}{A_j p} \right\}$. We set $s_{i+1} = s_i + tp$. This process is described in Algorithm 5. Of course, n_{points} must be less than or equal to $n + 1$ in order for this to be defined.

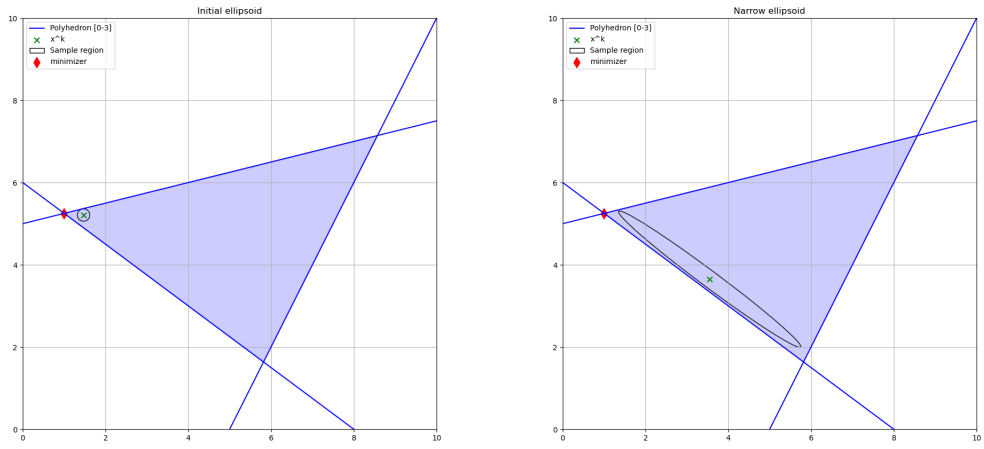

Figure 6: Why only considering the nearest constraint is not sufficient. On the left is the starting ellipsoid. By choosing centers further away from only the nearest constraint, the ellipsoid becomes narrow as another constraint is limiting the length of the second axis.

Algorithm 5: Path segment construction

Step 0 **(Initialization)**
Choose a number of segments $n_{\text{points}} \leq n$, and set $s_0 = x^{(k)}$, $i = 1$.

Step 1 **(Compute nearest constraints)**
Compute $d_j = b - A_j s_{i-1}$ for each $j \in [m]$, and partition

$$E = \left\{ j \in [m] \middle| d_j = \min_{l \in [m]} d_l \right\}, \text{ and } R = [m] \setminus E.$$

If $|E| \geq n$, then **Stop**.

Step 2 **(Compute search segment)**
Compute the search direction $p = {A_E}^T \left[A_E {A_E}^T \right]^{-1} (b_E + A_E s_{i-1})$ and distance $t = \min_{j \in R} \left\{ \frac{b_j - A_j s_{i-1}}{A_j p} \right\}$.
Set $s_i \leftarrow s_{i-1} + tp$.

Step 4 **(Repeat)**
if $i = n_{\text{points}}$ **stop**, otherwise set $i \leftarrow i + 1$ and go to Step 1.

Once each end point s_i is computed, the algorithm searches along each line segment $[s_{i-1}, s_i]$ for $i \in [n_{\text{points}}]$. This means that we can define a class of searches that each limit the number of line segments to search n_{points}. Within the results, these algorithms are described as "ellipse segment n_{points}".

10.2 Buffered segments

Even with these modifications, the iterates may approach the boundary of the feasible region. Another way of dealing with this is to shift the constraints closer to the current iterate. Namely, we introduce a parameter υ to determine how far to scale the constraints. Then, within the trust region subproblem, we add constraints of $Ax \leq b\upsilon + (1-\upsilon)Ax^{(k)}$. This produces the buffered segment searches within the results.

Circumscribed ellipse. We also implemented a variant of the algorithm in which the sample region is a smallest volume ellipsoid to contain the feasible region. Notice the ellipsoid necessarily includes points that are outside the current trust region and may include infeasible points. Thus, both the trust region and linear constraints have to be added to the optimization problem defining the replacement point while computing the Lagrange polynomials. For more details about this modified model improvement algorithm, see Section 9.

10.3 Sample problem

The first test was on a problem with simple constraints and a pathological objective. We let $f(x) = \epsilon x + (1-\epsilon)(y - \alpha x \sin(\gamma x))^2$ for a fixed constant ϵ, and set the constraints to be $x_2 \leq ax_1$, $x_2 \geq -ax_1$ for a fixed constant a. We summarize the number of function evaluations and iterations taken within **??**.

In general, we notice the linear models use fewer evaluations than quadratic models. We see that the method with the fewest iterations and function evaluations is the linear polyhedral shape. This is likely because the polyhedral shape is allowed to search the entire outer trust region. This also explains why the circumscribed ellipse and maximum volume simplex also perform well. Also, the scaled ellipsoid performs comparably to the unscaled version.

10.4 Hock-Schittkowksi test problems

We tested these algorithms on several problems from the Hock-Schittkowski problem set [77] and [78]. We selected the problems that have linear constraints: 21, 24, 25, 35, 36, 44, 45, 76, 224, 231, 232, 250, 251. We summarize the results within **??**.

Performance profile. In order to better evaluate the algorithms on the problems across in this test set, we use a performance profile developed in [79]. Given a set of Solvers $\mathcal{S}$ that solved a set of problems $\mathcal{P}$ with the number of evaluations of solver s on problem p being $N(s,p)$, the performance ratio is defined to be $r(s,p) = \frac{N(s,p)}{\min_{s\in\mathcal{S}} N(s,p)}$. If the algorithm does not complete, then the number of evaluations is set to ∞. The performance profile of a solver s and parameter $\alpha \in [0,\infty)$ is then the number of problems for which the performance ratio is less than or equal to α:

$$\rho(s,\alpha) = \frac{1}{\|\mathcal{P}\|} \|p \in \mathcal{P} | r(s,p) \leq \alpha\| . \tag{65}$$

The y axis of a performance plot is the performance profile, and the x axis is the parameter α. Note that algorithms with high performance profiles for small values of α solved a large number of problems the most with the fewest evaluations, while algorithms that eventually reach high performance profiles with larger values of α solve a large set of problems. The performance profile for the Hock-Schittkowski problem set is given in figure Figure 7.

The line segment search with 5 segments does not solve many problems, this is because several of the problems have dimension less than 5, so that it was not even run on these. Notice that the polyhedral search does very well. We conjecture that this may not hold with modeled, nonlinear constraints.

References

[1] N. Neveu, J. Larson, J. G. Power, and L. Spentzouris, "Photoinjector optimization using a derivative-free, model-based trust-region algorithm for the Argonne Wakefield Accelerator," *Journal of Physics: Conference Series*, vol. 874, no. 1, p. 012062, 2017.

[2] N. Ploskas, C. Laughman, A. U. Raghunathan, and N. V. Sahinidis, "Optimization of circuitry arrangements for heat exchangers using derivative-free optimization," *Chemical Engineering Research and Design*, vol. 131, pp. 16–28, 2018. Energy Systems Engineering.

[3] N. B. Kovachki and A. M. Stuart, "Ensemble Kalman inversion: a derivative-free technique for machine learning tasks," *Inverse Problems*, vol. 35, p. 095005, Aug 2019.

[4] Z. Cheng, E. Shaffer, R. Yeh, G. Zagaris, and L. Olson, "Efficient parallel optimization of volume meshes on heterogeneous computing systems," *Engineering with Computers*, vol. 33, no. 4, pp. 717–726, 2017.

[5] T. Gao and J. Li, "A derivative-free trust-region algorithm for reliability-based optimization," *Structural and Multidisciplinary Optimization*, vol. 55, no. 4, pp. 1535–1539, 2017.

[6] J. S. Eldred, J. Larson, M. Padidar, E. Stern, and S. M. Wild, "Derivative-free optimization of a rapid-cycling synchrotron," Tech. Rep. 2108.04774, ArXiv, 2021.

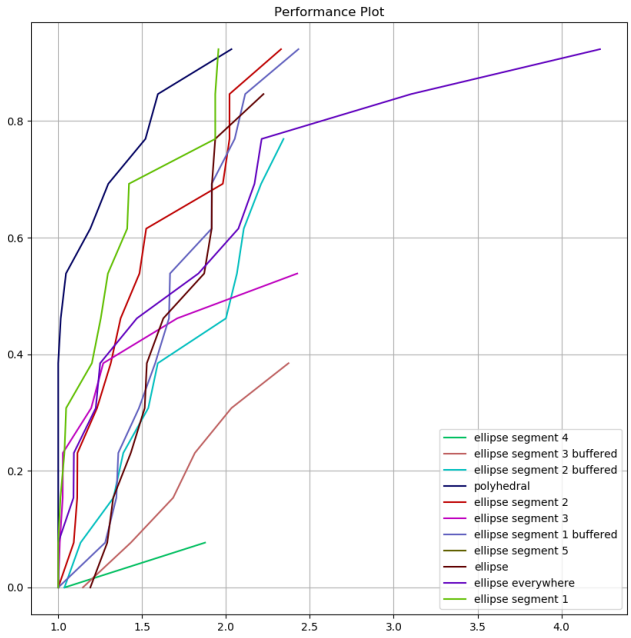

Figure 7: A performance profile comparing different variants of the algorithm for linear constraints. We see that the polyhedral algorithm is both efficient and robust.

[7] S. Le Digabel and S. M. Wild, "A taxonomy of constraints in simulation-based optimization," Tech. Rep. 1505.07881, ArXiv, 2015.

[8] A. R. Conn, K. Scheinberg, and L. N. Vicente, *Introduction to Derivative-Free Optimization*. Society for Industrial and Applied Mathematics, 2009.

[9] P. D. Conejo, E. W. Karas, L. G. Pedroso, A. A. Ribeiro, and M. Sachine, "Global convergence of trust-region algorithms for convex constrained minimization without derivatives," *Appl. Math. Comput.*, vol. 220, pp. 324–330, 2013.

[10] L. M. Rios and N. V. Sahinidis, "Derivative-free optimization: a review of algorithms and comparison of software implementations," *Journal of Global Optimization*, vol. 56, no. 3, pp. 1247–1293, 2013.

[11] A. Custodio, K. Scheinberg, and L. Vicente, "Methodologies and software for derivative-free optimization," 2017.

[12] J. Larson, M. Menickelly, and S. M. Wild, "Derivative-free optimization methods," *Acta Numerica*, vol. 28, pp. 287–404, 2019.

[13] E. Fermi and N. Metropolis, "Numerical solution of a minimum problem," Tech. Rep. LA-1492, Los Alamos Scientific Laboratory of the University of California, 1952.

[14] J. A. Nelder and R. Mead, "A simplex method for function minimization," *The Computer Journal*, vol. 7, pp. 308–313, 01 1965.

[15] R. Hooke and T. A. Jeeves, ""Direct search" solution of numerical and statistical problems," *J. ACM*, vol. 8, p. 212–229, April 1961.

[16] T. G. Kolda, R. M. Lewis, and V. Torczon, "Optimization by direct search: New perspectives on some classical and modern methods," *SIAM Rev.*, vol. 45, pp. 385–482, 2003.

[17] V. Torczon, "On the convergence of pattern search algorithms," *SIAM J. Optim.*, vol. 7, pp. 1–25, 1997.

[18] C. Audet and J. E. Dennis, "Analysis of generalized pattern searches," *SIAM Journal on Optimimization*, vol. 13, pp. 889–903, 2002.

[19] C. Audet and J. E. Dennis, "Mesh adaptive direct search algorithms for constrained optimization," *SIAM J. Optim.*, vol. 17, pp. 188–217, 2006.

[20] M. Abramson and C. Audet, "Convergence of mesh adaptive direct search to second-order stationary points," *SIAM Journal on Optimization*, vol. 17, pp. 606–619, 01 2006.

[21] A. R. Conn, K. Scheinberg, and P. L. Toint, "Recent progress in unconstrained nonlinear optimization without derivatives," *Mathematical Programming*, vol. 79, pp. 397–414, 1997.

[22] M. J. D. Powell, "UOBYQA: unconstrained optimization by quadratic approximation," *Mathematical Programming*, vol. 92, pp. 555–582, 2002.

[23] M. J. D. Powell, "The NEWUOA software for unconstrained optimization without derivatives," in *Large-Scale Nonlinear Optimization* (G. Di Pillo and M. Roma, eds.), pp. 255–297, Boston, MA: Springer US, 2006.

[24] R. Oeuvray and M. Bierlaire, "Boosters: A derivative-free algorithm based on radial basis functions," *International Journal of Modelling and Simulation*, vol. 29, no. 1, pp. 26–36, 2009.

[25] S. Wild, R. Regis, and C. Shoemaker, "ORBIT: Optimization by radial basis function interpolation in trust-regions," *SIAM J. Scientific Computing*, vol. 30, pp. 3197–3219, 01 2008.

[26] S. Wild and C. Shoemaker, "Global convergence of radial basis function trust region derivative-free algorithms," *SIAM Journal on Optimization*, vol. 21, pp. 761–781, 07 2011.

[27] A. Booker, J. Dennis, P. Frank, D. Serafini, V. Torczon, and M. Trosset, "A rigorous framework for optimization of expensive functions by surrogates," *Structural Optimization*, vol. 17, 09 1998.

[28] H. A. Le Thi, I. Vaz, and L. Vicente, "Optimizing radial basis functions by d.c. programming and its use in direct search for global derivative-free optimization," *Top*, vol. 20, pp. 190–214, 04 2012.

[29] A. Custódio and L. Vicente, "Using sampling and simplex derivatives in pattern search methods," *SIAM Journal on Optimization*, vol. 18, pp. 537–555, 01 2007.

[30] R. M. Lewis and V. Torczon, "A globally convergent augmented Lagrangian pattern search algorithm for optimization with general constraints and simple bounds," *SIAM Journal on Optimization*, vol. 12, no. 4, pp. 1075–1089, 2002.

[31] R. M. Lewis and V. Torczon, "A direct search approach to nonlinear programming problems using an augmented Lagrangian method with explicit treatment of linear constraints," Tech. Rep. WM-CS-2010-01, Department of Computer Science, College of William and Mary, 2010.

[32] L. F. Bueno, A. Friedlander, J. M. Martínez, and F. N. C. Sobral, "Inexact restoration method for derivative-free optimization with smooth constraints," *SIAM J. Optim.*, vol. 23, pp. 1189–1213, 2013.

[33] R. Brekelmans, L. Driessen, H. Hamers, and D. den Hertog, "Constrained optimization involving expensive function evaluations: A sequential approach," *Eur. J. Oper. Res.*, vol. 160, pp. 121–138, 2005.

[34] P. S. Ferreira, E. W. Karas, M. Sachine, and F. N. C. Sobral, "Global convergence of a derivative-free inexact restoration filter algorithm for nonlinear programming," *Optimization*, vol. 66, pp. 271 – 292, 2017.

[35] C. Audet and J. E. Dennis, "A progressive barrier for derivative-free nonlinear programming," *SIAM Journal on Optimization*, vol. 20, no. 1, pp. 445–472, 2009.

[36] G. Liuzzi and S. Lucidi, "A derivative-free algorithm for inequality constrained nonlinear programming via smoothing of an ℓ_∞ penalty function," *SIAM Journal on Optimization*, vol. 20, no. 1, pp. 1–29, 2009.

[37] G. Liuzzi, S. Lucidi, and M. Sciandrone, "Sequential penalty derivative-free methods for nonlinear constrained optimization," *SIAM Journal on Optimization*, vol. 20, no. 5, pp. 2614–2635, 2010.

[38] G. Fasano, G. Liuzzi, S. Lucidi, and F. Rinaldi, "A linesearch-based derivative-free approach for nonsmooth constrained optimization," *SIAM Journal on Optimization*, vol. 24, no. 3, pp. 959–992, 2014.

[39] M. Diniz-Ehrhardt, J. M. Martínez, and L. Pedroso, "Derivative-free methods for nonlinear programming with general lower-level constraints," *Computational & Applied Mathematics*, vol. 30, pp. 19–52, 07 2016.

[40] V. Picheny, R. B. Gramacy, S. M. Wild, and S. L. Digabel, "Bayesian optimization under mixed constraints with a slack-variable augmented Lagrangian," in *Advances in Neural Information Processing Systems 29*, pp. 1435–1443, Curran Associates, Inc., 2016.

[41] C. Audet and J. E. Dennis, "Pattern search algorithms for mixed variable programming," *SIAM Journal on Optimization*, vol. 11, no. 3, pp. 573–594, 2001.

[42] T. Pourmohamad, *Combining Multivariate Stochastic Process Models with Filter Methods for Constrained Optimization*. PhD thesis, UC Santa Cruz: Statistics and Applied Mathematics, 2016.

[43] N. Echebest, M. L. Schuverdt, and R. P. Vignau, "An inexact restoration derivative-free filter method for nonlinear programming," *Computational and Applied Mathematics*, vol. 36, pp. 693–718, Mar 2017.

[44] P. R. Sampaio and P. L. Toint, "A derivative-free trust-funnel method for equality-constrained nonlinear optimization," *Computational Optimization and Applications*, vol. 61, pp. 25–49, May 2015.

[45] P. R. Sampaio and P. L. Toint, "Numerical experience with a derivative-free trust-funnel method for nonlinear optimization problems with general nonlinear constraints," *Optimization Methods and Software*, vol. 31, no. 3, pp. 511–534, 2016.

[46] H. Glass and L. Cooper, "Sequential search: A method for solving constrained optimization problems," *J. ACM*, vol. 12, p. 71–82, jan 1965.

[47] M. J. D. Powell, "A direct search optimization method that models the objective and constraint functions by linear interpolation," in *Advances in Optimization and Numerical Analysis* (S. Gomez and J.-P. Hennart, eds.), pp. 51–67, Dordrecht: Springer Netherlands, 1994.

[48] Á. Bűrmen, J. Olenšek, and T. Tuma, "Mesh adaptive direct search with second directional derivative-based Hessian update," *Computational Optimization and Applications*, vol. 62, pp. 693–715, Dec 2015.

[49] A. Tröltzsch, "A sequential quadratic programming algorithm for equality-constrained optimization without derivatives," *Optimization Letters*, vol. 10, no. 2, pp. 383–399, 2016.

[50] M. J. Box, "A new method of constrained optimization and a comparison with other methods," *The Computer Journal*, vol. 8, pp. 42–52, 04 1965.

[51] W. Spendley, G. R. Hext, and F. R. Himsworth, "Sequential application of simplex designs in optimisation and evolutionary operation," *Technometrics*, vol. 4, no. 4, pp. 441–461, 1962.

[52] J. H. May, *Linearly constrained nonlinear programming: a solution method that does not require analytic derivatives*. PhD thesis, Yale University, 1974.

[53] J. H. May, "Solving nonlinear programs without using analytic derivatives," *Operations Research*, vol. 27, no. 3, pp. 457–484, 1979.

[54] F. V. Berghen, *CONDOR: A constrained, non-linear, derivative-free parallel optimizer for continuous, high computing load, noisy objective functions*. PhD thesis, Université Libre de Bruxelles, 2004.

[55] R. M. Lewis and V. Torczon, "Pattern search methods for linearly constrained minimization," *SIAM J. Optim.*, vol. 10, pp. 917–941, 2000.

[56] S. Lucidi and M. Sciandrone, "A derivative-free algorithm for bound constrained optimization," *Computational Optimization and Applications*, vol. 21, pp. 119–142, 2002.

[57] G. Chandramouli and V. Narayanan, "A scaled conjugate gradient based direct search algorithm for high dimensional box constrained derivative free optimization," 2019.

[58] T. G. Kolda, R. M. Lewis, and V. Torczon, "Stationarity results for generating set search for linearly constrained optimization," *SIAM Journal on Optimization*, vol. 17, no. 4, pp. 943–968, 2007.

[59] S. Lucidi and P. Tseng, "Objective-derivative-free methods for constrained optimization," *Mathematical Programming, Series B*, vol. 92, pp. 37–59, 03 2002.

[60] C. Audet, S. Le Digabel, and M. Peyrega, "Linear equalities in blackbox optimization," *Computational Optimization and Applications*, vol. 61, pp. 1–23, 2015.

[61] S. Gratton, C. W. Royer, L. N. Vicente, and Z. Zhang, "Direct search based on probabilistic feasible descent for bound and linearly constrained problems," *Computational Optimization and Applications*, vol. 72, pp. 525–559, Apr 2019.

[62] S. Gratton, C. W. Royer, L. N. Vicente, and Z. Zhang, "Direct search based on probabilistic descent," *SIAM Journal on Optimization*, vol. 25, no. 3, pp. 1515–1541, 2015.

[63] M. Powell, "The BOBYQA algorithm for bound constrained optimization without derivatives," DAMTP 2009/NA06, University of Cambridge, 2009.

[64] B. Arouxet, N. Echebest, and E. Pilotta, "Active-set strategy in Powell's method for optimization without derivatives," *Computational and Applied Mathematics*, vol. 30, pp. 171–196, 07 2016.

[65] S. M. Wild, *Derivative-free optimization algorithms for computationally expensive functions*. PhD thesis, Cornell University, 2008.

[66] S. Gratton, P. L. Toint, and A. Tröltzsch, "An active-set trust-region method for derivative-free nonlinear bound-constrained optimization," *Optimization Methods and Software*, vol. 26, no. 4-5, pp. 873–894, 2011.

[67] E. A. E. Gumma, M. H. A. Hashim, and M. M. Ali, "A derivative-free algorithm for linearly constrained optimization problems," *Computational Optimization and Applications*, vol. 57, pp. 599–621, Apr 2014.

[68] G. Galvan, M. Sciandrone, and S. Lucidi, "A parameter-free unconstrained reformulation for nonsmooth problems with convex constraints," *Computational Optimization and Applications*, 2021. To appear.

[69] F. Augustin and Y. M. Marzouk, "NOWPAC: a provably convergent derivative-free nonlinear optimizer with path-augmented constraints," Tech. Rep. 1403.1931, ArXiv, 2014.

[70] R. Bollapragada, M. Menickelly, W. Nazarewicz, J. O'Neal, P.-G. Reinhard, and S. M. Wild, "Optimization and supervised machine learning methods for fitting numerical physics models without derivatives," *Journal of Physics G: Nuclear and Particle Physics*, vol. 48, no. 2, p. 024001, 2021.

[71] R. G. Regis and S. M. Wild, "CONORBIT: constrained optimization by radial basis function interpolation in trust regions," *Optimization Methods and Software*, vol. 32, no. 3, pp. 552–580, 2017.

[72] P. Conejo, E. Karas, and L. Pedroso, "A trust-region derivative-free algorithm for constrained optimization," *Optimization Methods and Software*, vol. 30, no. 6, pp. 1126–1145, 2015.

[73] A. Brilli, G. Liuzzi, and S. Lucidi, "An interior point method for nonlinear constrained derivative-free optimization," Tech. Rep. 2108.05157, ArXiv, 2021.

[74] A. R. Conn, N. I. M. Gould, and P. L. Toint, *Trust-region Methods*. Society for Industrial and Applied Mathematics, 2000.

[75] J. E. Dennis and R. B. Schnabel, *Numerical methods for unconstrained optimization and nonlinear equations*. Englewood Cliffs, N.J: Prentice-Hall, 1983.

[76] L. G. Khachiyan and M. J. Todd, "On the complexity of approximating the maximal inscribed ellipsoid for a polytope," *Mathematical Programming*, vol. 61, no. 1, pp. 137–159, 1993.

[77] K. Schittkowski, *More Test Examples for Nonlinear Programming Codes*. No. 282 in Lecture Notes in Economics and Mathematical Systems, Berlin, Heidelberg: Springer-Verlag, 1987.

[78] W. Hock and K. Schittkowski, "Test examples for nonlinear programming codes," *Journal of Optimization Theory and Applications*, vol. 30, no. 1, pp. 127–129, 1980.

[79] J. J. Moré and S. M. Wild, "Benchmarking derivative-free optimization algorithms," *SIAM Journal on Optimization*, vol. 20, no. 1, pp. 172–191, 2009.