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

Trever Hallock and Stephen C. Billups

December 7, 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 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], these constraints are called *unrelaxable*. For problems involving unrelaxable constraints, numerical function values at infeasible points (if obtained) are likely to be unreliable. We, therefore, require all sample points to be feasible, and we seek to avoid infeasible function evaluations. 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 10.3, 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 $b \in \mathbb{R}^m$.

The central idea of the method described in **??** 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.

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. 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.

Here, we consider the case of a black-box objective with linear constraints. Because the constraints are known, we present an algorithm that constructs sample points from the feasible region intersected with the current trust region. This region is called the *sample region*. The algorithm we present is an implementation of the algorithmic template described in [9]. As such, we can prove its convergence by showing that it satisfies the required assumptions presented in [9]. We then discuss several variants to improve on the algorithm, as well as provide numerical results.

In future work, we will consider the case of nonlinear black-box constraints. In contrast to the case of linear constraints, we can no longer be assured that feasible points for the model sub-problems are feasible for the original problem. We, therefore, propose a methodology that constructs second-order cones for each nearly-active constraint and then constructs an ellipsoidal sample trust region within the intersection of these cones. We show that under reasonable assumptions, this sample trust region is guaranteed to be feasible when the trust region is sufficiently small. Using this result, we prove that the criticality measure for the iterates generated by the algorithm converges to zero. Thus, any accumulation point of the iterates satisfies the first-order optimality conditions. Finally, numerical results are presented comparing the performance of our algorithm to several other algorithms for constrained derivative-fee optimization.

2 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

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

3 Background

3.1 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 8 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.

3.2 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 3.3.

3.3 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 $\rho_k>\gamma_{\text{suff}}$.

3.4 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 (67) 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.

3.5 Sample Set Geometry

The term *geometry* describes how the distribution of points in the sample set Y affects the model's accuracy.

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*.

There is a close connection between the Λ-poisedness of Y and the condition number of the Vandermonde matrix associated with the monomial basis for $\mathcal{P}_n^2$

$$\bar{\Phi} = \{\bar{\phi}_0, \ldots, \bar{\phi}_p\} = \{1, x_1, \ldots, x_n, x_1^2/2, \ldots x_n^2/2, x_1 x_2, \ldots, x_{n-1} x_n\}. \tag{11}$$

In particular, let $\hat{Y}$ be the shifted and scaled sample set $\left\{ \frac{y^{(i)} - y^{(0)}}{\Delta} | y^{(i)} \in Y \right\}$. Then we have the following result:

Theorem 3.1 *([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.*

A limitation of Theorem 3.1 is that it assumes $\hat{Y}$ is contained in the unit ball $B_2(0;1)$. The following theorem does not make this assumption.

Theorem 3.2 *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 3.3 *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 3.4 *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 3.5 *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* (68)*, 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 3.3 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 3.4 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 3.1 is its requirement to have a point evaluated at the origin. We remove this assumption in Theorem 3.7 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 3.6 *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 3.7 *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 (68) *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 3.6 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),

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

□

3.6 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 6.

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 10.3 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 6 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 6

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 6 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 3.9 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 3.8 *([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 3.9 *For any given $\xi_{\min} \in (0, 1/4]$, Algorithm 6 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 3.8,

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

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 3.10 *Let $\Delta \geq 1$, and suppose that Algorithm 6 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* (68)*, 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 6 never swaps or replaces $y^{(0)}$. By Lemma 3.9, $\hat{Y} \subset B_2(0;\Delta)$ 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 3.2, using the fact that $\Delta \geq 1$, we have that Y is Λ-poised in $B_2(0;\Delta)$, where

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

□

3.7 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 10.3 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.

3.8 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.

4 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 7, 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 6 to a transformed problem in which the ellipsoidal region is mapped onto a ball.

In the second method, described in Section 8, 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 8 as a modification of model-improvement algorithm (Algorithm 6). 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.

5 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 6 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 5.1 *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* (92)*,*
2. *$x^{(k)} \in \hat{E}^{(k)}$,*
3. *The condition number $\kappa\left(Q^{(k)}\right)$ is bounded independently of k.*

6 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 6.1 *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 6.2 *The function f is bounded below over Ω. That is,*

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

Assumption 6.3 *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 6.4 *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 (86) within [74]. This is a reasonable choice for $s^{(k)}$, but the exact solution as chosen within Algorithm 2 necessarily also satisfies (86) 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 10.2 which is a strictly stronger assumption. As their second assumption H_2, they assume our Assumption 10.3. 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 Assumption 10.4 is the topic of Section 10.18.1. 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 (75) is the topic of Section 10.18.2.

6.1 Bounded Hessians

First, we show that Assumption 10.7 can be used instead of Assumption 10.4.

Lemma 6.1 *Suppose that Assumption 10.2 and Assumption 10.7 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 Assumption 10.4 is also satisfied.*

Proof:

By Assumption 10.7, 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 3.5 to bound $\left\|\hat{M}\left(\bar{\Phi}, Y\right)^{-1}\right\|$, and apply Theorem 3.7. Thus, we see that (72) is satisfied. Combining this with $\Delta_k \leq \Delta_{\max}$, we see that

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

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

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

□

6.2 Satisfying the accuracy assumption

After an ellipsoidal $T^{(k)}_{\text{sample}}$ is constructed, sample points are then selected by applying Algorithm 6 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 5.1, 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 6 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 (70)

This section analyzes the accuracy of the model functions constructed from the output of Algorithm 3. We first show the intermediate result Theorem 10.4 before satisfying (75). In particular, Theorem 6.3 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 6.2 *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 5.1:*

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

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

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

□

Theorem 10.4 shows that a uniform bound on the condition number of $Q^{(k)}$ will produce accurate model functions. Indeed, recall from Theorem 10.3 that Algorithm 6 produces a Λ-poised sample set, for $B_2(0;\delta)$. Then Lemma 3.5 and Theorem 3.6 ensure m_f will satisfy the provided error bounds over the ellipsoidal region as well. We can now satisfy the accuracy condition (75).

Theorem 6.3 *Suppose that Assumption 10.2, Assumption 10.7, and Assumption 10.5 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 5.1, 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 (75) *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 Theorem 10.4, 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 6 to choose sample points, we know by Theorem 10.3 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 3.7 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 Theorem 10.4, 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 Theorem 10.4 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, $x^{(k)} \in \hat{E}^{(k)}$. □

7 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.

7.1 A safe ellipsoid

7.1.1 Construction

This section shows how to construct one possible ellipsoid that satisfies Definition 5.1. 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 Theorem 10.10 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 $\mathcal{A}\left(x^{(k)}\right) \neq \emptyset$.

7.1.2 Suitability

We show that this construction provides a suitable ellipsoid according to Definition 5.1.

Lemma 7.1 *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 5.1.*

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 7.2 *Let π_2 be defined by* (96).
Suppose that Assumption 10.5 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 10.5, 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 (92), 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 (94), (95), and (96); 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 7.3 *Let $\pi^{(k)}$ be defined by* (97)*.*
Suppose that Assumption 10.5 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 (94), (95), and (97) can only take on at most $1 + 2^m$ values. By Theorem 10.8, 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 7.4 *Let $C_{unsh}^{(k)}$ and $\mathcal{P}^{(k)}$ be defined by* (100) *and* (92)*.*
The set $C_{unsh}^{(k)}$ is feasible with respect to the active constraints of $\mathcal{P}^{(k)}$ at $x^{(k)}$.

Proof:

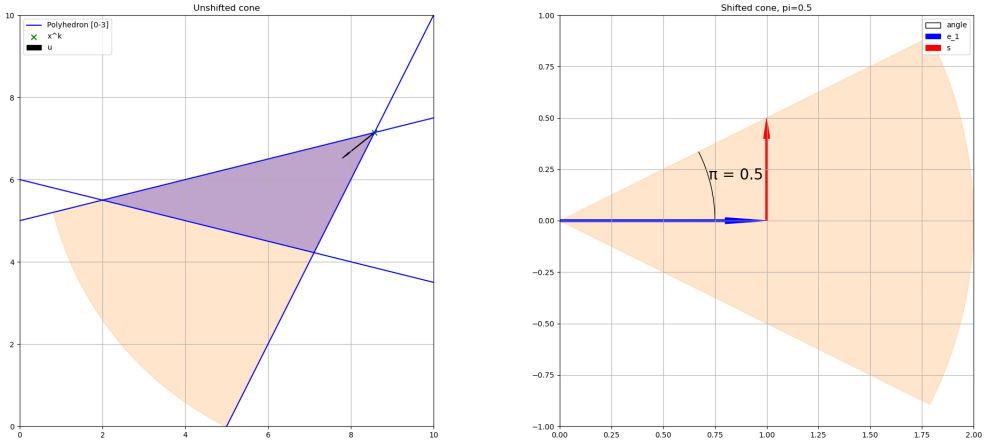

Figure 1: On the left is an unshifted cone $C_{\text{unsh}}^{(k)}$. The zeros of the polyhedron constraint functions are in blue, and the current iterate is in green. A feasible direction u from the current iterate is calculated, and the width of the cone is determined to lie within the active constraints of the polyhedra. On the right is a shifted cone $C_{\text{unsh}}^{(k)}$. 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), (97), (98) and A and b be defined by (29). Let $y = x^{(k)} + t\hat{u}^{(k)} + s \in C_{\text{unsh}}^{(k)}$ 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 7.5 *Let $C_{sh}^{(k)}, f_e, \pi^{(k)}$ be defined as in* (101), (104), (97). *Then, for all $\delta > 0$, the ellipsoid*

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

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

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

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

Thus, $x \in C_{\text{sh}}^{(k)}$. □

Lemma 7.6 *Let $R^{(k+1)}$, T_k, $C_{unsh}^{(k)}$, $C_{sh}^{(k)}$ be defined as in* (102), (103), (100), (101). *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 $t \ge 0$ and $s \in \mathbb{R}^n$ such that $x = x^{(k)} + t\hat{u}^{(k)} + s$ where $s^T\hat{u}^{(k)} = 0$ and $\|s\| \le \pi^{(k)}t$. 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 $\|\sigma\| = \|s\| \le \pi^{(k)}t$. 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 $\|s\| = \|\sigma\| \le \pi^{(k)}t$. Hence $T_k^{-1}\left(\begin{pmatrix} t \\ \sigma \end{pmatrix}\right) \in C_{\text{unsh}}^{(k)}$. □

Lemma 7.7 *Let $\mathcal{A}$, f_e, δ_f $\pi^{(k)}$, $R^{(k+1)}$, T_k, and $\mathcal{P}^{(k)}$ be defined by* (44), (104), (52), (97), (102), (101), *and* (92), *respectively.*

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

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

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), (102), (100), and (101), respectively. We see that if $x \in E^{(k)}$, then by Theorem 10.12 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 $\|\sigma\| \le \pi^{(k)}$, and

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

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

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 Theorem 10.11 to see that $E^{(k)} \subseteq \mathcal{P}^{(k)}$. □

Lemma 7.8 *Let δ_f, $R^{(k+1)}$, T_k, $C_{unsh}^{(k)}$, $C_{sh}^{(k)}$, $\mathcal{P}^{(k)}$ be defined as in* (52), (102), (103), (100), (101), (92).
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 7.9 *Let $\mathcal{A}$ and $\pi^{(k)}$ be defined by* (44) *and* (97)*, respectively.*
Suppose that Assumption 10.5 holds.
For some iteration k, if $\mathcal{A} \neq \emptyset$, then the ellipsoid defined by (55) *is suitable according to Definition 5.1.*

Proof:

Let f_e, T_k, $R^{(k+1)}$, δ_f, and $\hat{\mathcal{E}}_{\text{feasible}}^k$ be defined as in (104), (103), (102), (106), and (107), respectively. Observe that Definition 5.1 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 Theorem 10.14, we know that $E^{(k)} \subseteq \mathcal{P}^{(k)}$. By Theorem 10.15, we know that $x^{(k)} \in \hat{E}^{(k)}$. By Theorem 10.9, 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 $R^{(k+1)}$. □

7.2 Ellipsoid searches

Within Section 7.1.1, we showed how to construct one possible ellipsoid that satisfies Definition 5.1. 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 5.1 to share the same convergence results.

7.2.1 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 6.

7.2.2 Maximal volume ellipsoids

The error bounds given in Theorem 10.4 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 9.1.2, 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 7.10 *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 (77)*, 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 (77). □

Notice that the constraints for (77) 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

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

8 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 6. Recall that Algorithm 6 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 6 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 13, 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.

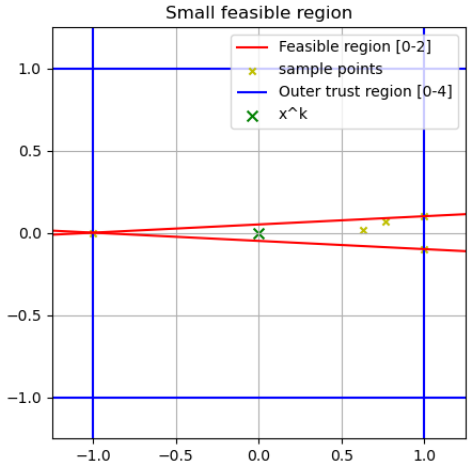

Figure 2: An example of constraints limiting sample point choices. If the constraints remove a large region of the trust region, there may be no feasible Λ-poised set for a given $\Lambda > 0$.

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 8. 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.

9 Results

9.1 Algorithm variants

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

9.1.1 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 8. 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.

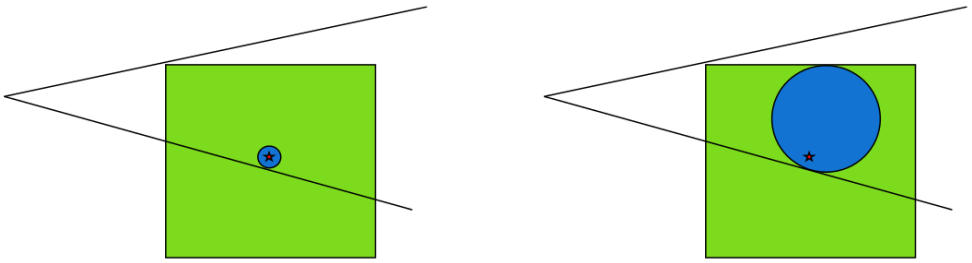

Figure 3: When the current iterate is too close to a constraint, the circular trust region becomes too small. Shifting the trust region center can help remedy this. The star is the current iterate, the outer trust region is in green, and the inner trust region is in blue.

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 9, 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: $c^{(k)} = x^{(k)}$.

9.1.2 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 9.

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 10, although the

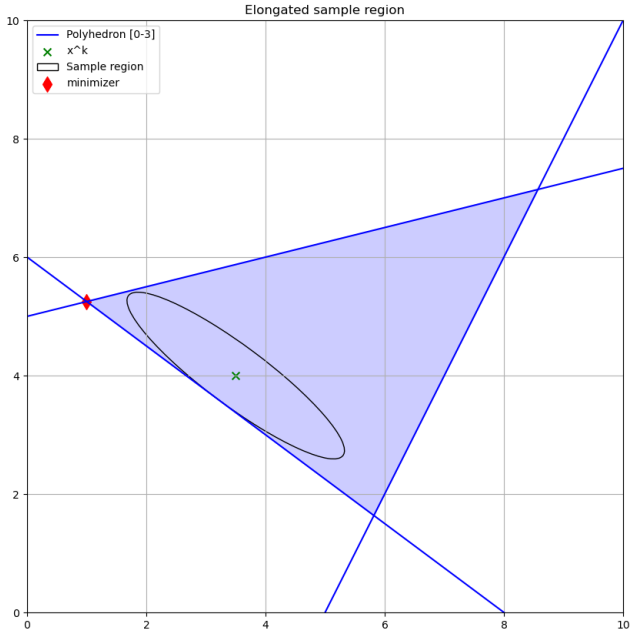

Figure 4: Although the center of the ellipsoid in green is close to the boundary of the blue constraints, it can elongate to allow progress.

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 11: 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

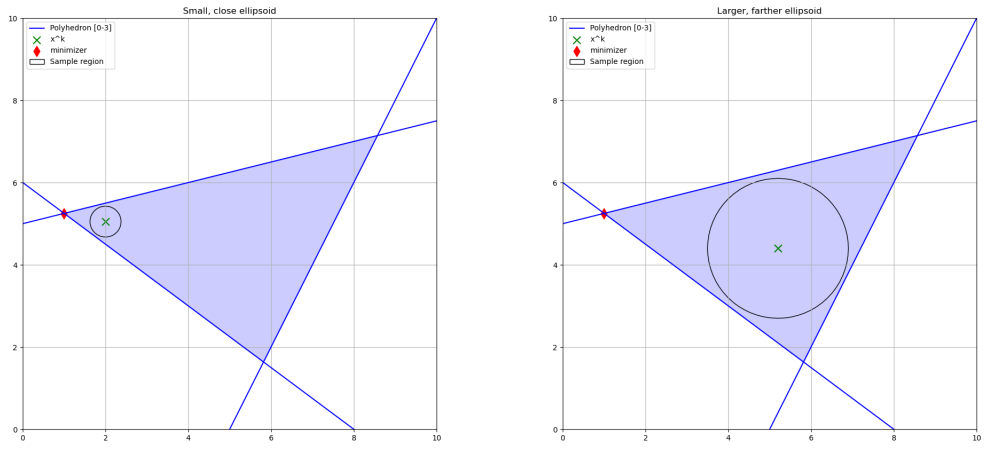

Figure 5: An example of how the search for the feasible region center can go wrong. On the left, a very small sample region is selected; however its proximity to the minimizer makes the model more accurate at the minimizer. On the right, a sample region with larger volume is chosen, but it is further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

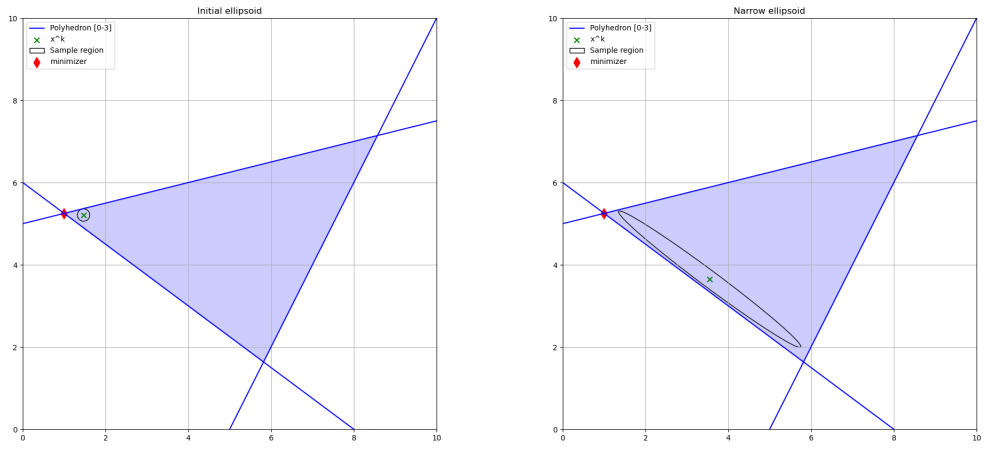

Figure 6: Why only considering the nearest constraint is not sufficient. On the left is the starting ellipsoid. By choosing centers further away from only the nearest constraint, the ellipsoid becomes narrow as another constraint is limiting the length of the second axis.

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}".

9.2 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 8.

9.3 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.

9.4 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.

10 Parts from old paper

10.1 Background

An excellent introduction to model-based trust region methods for derivative-free optimization is provided in [**?**]. At the heart of these methods is the idea of constructing model functions that approximate the objective function $f(x)$ over a trust-region.

10.2 Interpolation

Derivative free trust region methods construct models of the objective 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 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*

$$m_f(y^i) = \sum_{j=0}^{p} \alpha_j \phi_j(y^i) = f(y^i) \quad \forall \quad 0 \leq i \leq p. \tag{66}$$

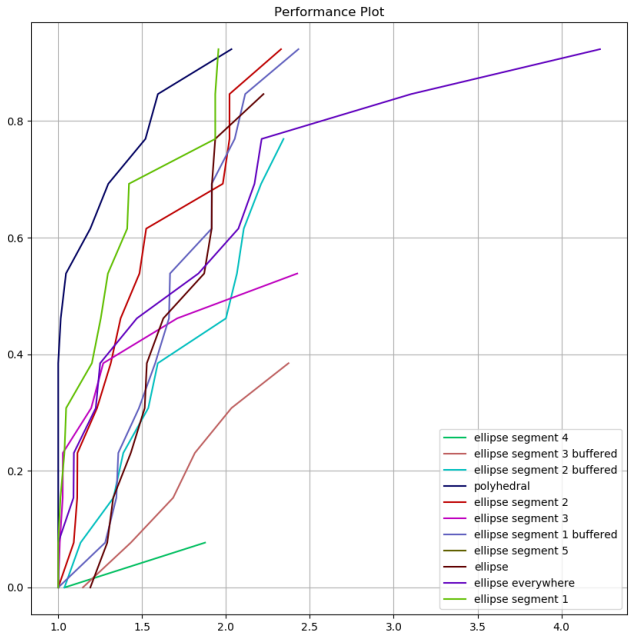

Figure 7: A performance profile comparing different variants of the algorithm for linear constraints. We see that the polyhedral algorithm is both efficient and robust.

This equation can be written more compactly in the form

$$V\alpha = \bar{f}, \tag{67}$$

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

$$V = M(\Phi, Y) := \begin{bmatrix} \phi_0(y^0) & \phi_1(y^0) & \ldots & \phi_p(y^0) \\ \phi_0(y^1) & \phi_1(y^1) & \ldots & \phi_p(y^1) \\ & & \vdots & \\ \phi_0(y^p) & \phi_1(y^p) & \ldots & \phi_p(y^p) \end{bmatrix}, \tag{68}$$

The interpolation equation (67) has a unique solution if and only if V is nonsingular. In this case, we say that the sample set Y is *poised* for interpolation with respect to the basis functions ϕ_i. However, even when V is nonsingular but "close" to singular, as measured by its condition number, the model's approximation may become inaccurate.

¡

10.3 Sample set geometry

The term *geometry* describes how the distribution of points in the sample set Y affects the model's accuracy. In the case of polynomial model functions, a careful analysis of model accuracy can be performed using *Lagrange polynomials*. Let the space of polynomials 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

$$l_i(y^j) = \delta_{i,j} \tag{69}$$

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

$$m_f(x) = \sum_{j=0}^{p} f(y^i) l_i(x). \tag{70}$$

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

$$\Lambda \geq \max_{0 \leq i \leq p} \max_{x \in B} |l_i(x)|. \tag{71}$$

In the case of interpolation over the quadratic polynomials, $\mathcal{P}_n^2$, we say that Y is Λ-*poised for quadratic interpolation*. The concept of Λ-poisedness allows us to establish the following error bounds, as shown in [**?**, Theorem]:

Theorem 10.1 *Let $Y = \{y^0, y^1, \ldots, y^p\} \subset \mathbb{R}^n$ be a set of $p+1 = \frac{(n+1)(n+2)}{2}$ sample points and $\Delta = \max_{1 \leq j \leq p} \|y^j - y^0\|$. Suppose that Y is Λ-poised for quadratic interpolation on $B(y^0; \Delta)$. Then, for any constant $L > 0$, there exist constants κ_h, κ_g, and κ_f such that the following error bounds hold for any function $f : \mathbb{R}^n \to \mathbb{R}$ that is LC^2 with Lipschitz constant L on an open set containing $B(y^0; \Delta)$:*

$$\|\nabla^2 f(y) - \nabla^2 m_f(y)\| \leq \kappa_h \Delta \quad \forall y \in B_2(y^0; \Delta) \tag{72}$$

$$\|\nabla f(y) - \nabla m_f(y)\| \leq \kappa_g \Delta^2 \quad \forall y \in B_2(y^0; \Delta) \tag{73}$$

$$|f(y) - m_f(y)| \leq \kappa_f \Delta^3 \quad \forall y \in B_2(y^0; \Delta). \tag{74}$$

where m_f is the quadratic model function interpolating f on Y.

There is a close connection between the Λ-poisedness of Y and the condition number of the Vandermonde matrix associated with the monomial basis $\bar{\Phi} = \{\bar{\phi}_0, \ldots, \bar{\phi}_p\} = \{1, x_1, \ldots, x_n, x_1^2/2, \ldots x_n^2/2, x_1 x_2, \ldots, x_{n-1} x_n\}$. In particular, let $\hat{Y}$ be the shifted and scaled sample set $\left\{ \frac{(y^i - y^0)}{\Delta} | y^i \in Y \right\}$. Then we have the following result

Theorem 10.2 *([?, Theorem 3.14]) Let $\hat{M} = M(\bar{\Phi}, \hat{Y})$. If $\hat{M}$ is nonsingular and $\left\|\hat{M}\right\|^{-1} \leq \Lambda$, then the set $\hat{Y}$ is $\Lambda\sqrt{p+1}$-poised in the unit ball $B(0;1)$. Conversely, if the set $\hat{Y}$ is Λ-poised for quadratic interpolation on the unit ball, then $\left\|\hat{M}^{-1}\right\| \leq \theta\Lambda\sqrt{p+1}$, where $\theta > 0$ is a constant dependent on n but independent of $\hat{Y}$ and Λ.*

TO DO — Move the following to later in the paper —

In particular, these bounds ensure that the following accuracy condition

$$\|\nabla m_f{}^{(k)}(x^{(k)}) - \nabla f(x^{(k)})\| \leq \kappa_g \Delta_k \tag{75}$$

for some fixed constant κ_g independent of k. We will extend these results for ellipsoidal trust regions in

A more detailed discussion can be found in [80], but a step to ensure good geometry is required for convergence analysis although it may come at the expense of adding more function evaluations.

10.4 Geometry 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 [11] and stated here in Algorithm 6.

The model improvement algorithm starts with a set of $p + 1$ sample points and 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 in the sample set 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, $j \geq i$ can be found such that $u_i(y^j)$ has sufficiently large magnitude, 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 which is obtained by maximizing $|u_i(x)|$ over the trust region.

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 6: Model Improvement Algorithm–REPLACE THIS

Step 0 (Initialization)

Initialize $i = 1$. Given a non-empty set Y of $p + 1$ points. Construct the Vandermonde matrix $V_{i,j} = \bar{\phi}_j(\frac{1}{\Delta}(y^i - y^0))$. Initialize constant $\xi_{\text{min}} > 0$.

Step 1 (Pivot)

Swap row i with row $i_{\max} = \arg\max_{j|j\geq i} V_{j,i}$

Step 2 (Check threshold)

If $|V_{i,i}| < \xi_{\text{min}}$ then select $\hat{y} \in \arg\max_{t|\|t\|\leq 1} |\phi_i(t)|$

Replace row i with $V_{i,j} \leftarrow \phi_j(\hat{y})$.

$Y \leftarrow Y \cup \{\hat{y}\} - \{y^i\}$

Step 3 (LU)

Set $V_{\bullet j} \leftarrow V_{\bullet j} - \frac{V_{i,j}}{V_{i,i}} V_{\bullet j} \forall j = i \ldots p$

If $i = p$ then **Stop**, otherwise Set $i \leftarrow i + 1$ and go to Step 1

At the completion of the algorithm, we obtain a Λ-poised sample set $Y = \{y^0, \ldots, y^p\}$ that is Λ-poised, where Λ is inversely proportional to $\xi_{\min}$ as given by the following result, which is shown in [**?**] through Theorems 6.5 and 3.14, and Section 6.7, Exercise 3:

Theorem 10.3 *The sample set Y obtained from Algorithm 6 is Λ-poised for quadratic interpolation, where $\Lambda > 0$ is a constant that depends only on ξ_{min} and n and is inversely proportional the ξ_{min}.*

We will show in Section 10.6 that this algorithm can be used to create a poised set over an ellipsoidal region.

10.5 Constructing feasible well-poised sample sets

The central idea behind our method is to choose sample points at the kth iteration from within a feasible ellipsoid defined by

$$E^{(k)} = \{x \in \mathbb{R}^n | (x - \mu^k)^T Q^{(k)} (x - \mu^k) \leq 1\}$$

where $Q^{(k)}$ is a positive definite matrix and μ^k is the center of the ellipsoid. $Q^{(k)}$ and μ^k are chosen so that the ellipsoid conforms roughly to the shape of the feasible region near the current iterate. Sample points are then selected by applying the model improvement algorithm to a transformed problem in which $E^{(k)}$ is mapped onto a ball.

In addition to requiring the ellipsoid to be feasible, we also require that it lie within the current trust region. To achieve this, we constrain the ellipsoid to lie within the polytope $P^k := \{x | Gx \leq b, x_i^{(k)} - \Delta_k \leq x_i \leq x_i^{(k)} + \Delta_k\}$. In essence, we have replaced the usual trust region $B_2(x^{(k)}, \Delta_k)$ with an L_1 ball, called the *outer trust region*, which is defined by

$$T_{\text{out}}^{(k)} = B_\infty(x^{(k)}, \Delta_k) = \{x \in \mathbb{R}^n |\ {x^{(k)}}_i - \Delta_k \leq x_i \leq {x^{(k)}}_i + \Delta_k \quad \forall 1 \leq i \leq n\}. \tag{76}$$

Defining $A = \begin{bmatrix} G \\ I \\ -I \end{bmatrix}$ and $b = \begin{bmatrix} g \\ {x^{(k)}}_i + \Delta_k \\ -{x^{(k)}}_i + \Delta_k \end{bmatrix}$, we can then write

$$P^k = \{x | Ax \leq b\}.$$

The following section analyzes the accuracy of the resulting model. We will then describe methods for choosing $Q^{(k)}$ and μ^k

10.6 Poisedness over Ellipsoidal Trust Regions

It is possible to show Λ-poisedness for an ellipsoidal region with a change of variables to the ball centered around the origin. We wish to construct a model for $f(x)$ in the ellipsoidal region $E^{(k)} = \{x \in \mathbb{R}^n | (x - c)^T Q^{(k)} (x - c) \leq 1\}$ for some symmetric, positive definite $Q^{(k)} \in \mathbb{R}^{n \times n}$ and some center $c \in \mathbb{R}^n$. We can give $Q^{(k)}$ its eigen-decomposition $Q^{(k)} = LD^2L^T$, where $L^TL = I$ and D is a diagonal matrix with nonnegative entries. Let $\delta = \max_{x \in E^{(k)}} \|x - c\|$. Then, the transformation $T(x) = \delta DL^T(x - c)$ maps $E^{(k)}$ to the δ ball $\{u = T(x) \in \mathbb{R}^n \mid \|u\| \leq \delta\}$. Conversely, $T^{-1}(u) = \frac{1}{\delta} LD^{-1}u + c$ maps the δ ball to the ellipsoidal region $E^{(k)}$.

Theorem 10.4 *Let T and δ be as defined above, and let $\hat{m}_f(u)$ be a model of the shifted objective $\hat{f}(u) = f(T^{-1}(u))$ in the δ ball such that there exist constants $\kappa_{ef}, \kappa_{eg}, \kappa_{eh} > 0$ such that for all $\{u \in R^n | \ \|u\| \leq \delta\}$, we have*

$$|\hat{m}_f(u) - \hat{f}(u)| \leq \kappa_{ef}\delta^3$$
$$\|\nabla\hat{m}_f(u) - \nabla\hat{f}(u)\| \leq \kappa_{eg}\delta^2$$
$$\|\nabla^2\hat{m}_f(u) - \nabla^2\hat{f}(u)\| \leq \kappa_{eh}\delta.$$

Then, with

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

we have that for all $x \in E^{(k)}$, the model function $m_f(x) = \hat{m}_f(T(x))$ will satisfy

$$|m(x) - f(x)| \leq \kappa'_{ef}\delta^3$$
$$\|\nabla m(x) - \nabla f(x)\| \leq \kappa'_{eg}\delta^2$$
$$\|\nabla^2 m(x) - \nabla^2 f(x)\| \leq \kappa'_{eh}\delta.$$

Proof:

We know that $\delta = \frac{1}{\sqrt{\lambda_{\min}(Q^{(k)})}} = \frac{1}{\min_i D_{i,i}}$. This means,

$$\kappa(Q^{(k)}) = \kappa(D^2) = \frac{\max_i D_{i,i}^2}{\min_i D_{i,i}^2} = \delta^2 \max_i D_{i,i}^2 = \delta^2\|D\|^2$$
$$\|D\| = \frac{1}{\delta}\sqrt{\kappa(Q^{(k)})} \leq \frac{\kappa_\lambda}{\delta}.$$

Also, $\delta \leq \Delta_k$ as the ellipse is constructed within the outer trust region.
Then, we have for all $\{u = T(x) \mid \|u\| \leq \delta\} \Leftrightarrow x \in E^{(k)}$

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

Similarily, for the gradient we find:

$$\|\nabla m_f(x) - \nabla f(x)\| = \delta\left\|DL^T\left(\nabla\hat{m}_f(u) - \nabla\hat{f}(u)\right)\right\| \leq \delta\|DL^T\|\|\nabla\hat{m}_f(u) - \nabla\hat{f}(u)\| \leq \sqrt{\kappa(Q^{(k)})}\kappa_{eg}\delta^2$$

Finally, we show that for the Hessian:

$$\|\nabla^2 m_f(x) - \nabla^2 f(x)\| = \delta^2\left\|DL^T\left(\nabla\hat{m}_f(u) - \nabla\hat{f}(u)\right)LD^T\right\| \leq \delta^2\|D\|^2\|\nabla\hat{m}_f(u) - \nabla\hat{f}(u)\| \leq \kappa(Q^{(k)})\kappa_{eh}\delta$$

□

This shows that in order to have strongly quadratic model functions, we need only bound the condition number of $Q^{(k)}$.

10.7 Finding the maximum volume feasible ellipsoid for a fixed center

The error bounds given in Theorem 10.4 suggest that we can obtain more accurate model functions by minimizing the condition number of the matrix $Q^{(k)}$. Here, we first solve the problem of finding the maximum-volume feasible ellipsoid given a fixed center. Later we will explore strategies for moving the center of the ellipsoid in order to improve performance.

We require the ellipsoid to be both feasible and also lie within the current trust region. To simplify the problem formulation, we use an L_1 ball rather than an L_2 ball for our trust region. In particular, define

We adopt a method similar to that described in [76], which presents an algorithm for finding the maximum volume inscribed ellipsoid for a polytope. Let P be a polytope defined by an $m \times n$ matrix A, $P = \{x \in \mathbb{R}^n \mid Ax \leq b\}$. In our algorithm P is defined as the intersection of the feasible region with an L_1 ball. We wish to find the maximum-volume ellipsoid $E \subset P$ centered at a point $\mu \in P$.

Let $\bar{b} = b - A\mu$ and $d = x - \mu$ so that the polytope becomes

$$P = \{\mu + d \in \mathbb{R}^n \mid Ad \leq \bar{b}\}$$

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

$$E = \{d \in \mathbb{R}^n \mid \frac{1}{2} d^T Q d \leq 1\}.$$

Our goal is to determine the matrix Q that maximizes the volume of E such that $\mu + E \subset P$. This is accomplished by solving the following problem for Q:

$$\begin{aligned} Q = V(\mu) = \sup_{Q \succeq 0} \det(Q^{-1}) \\ s.t. \quad A_i^T Q^{-1} A_i \leq \frac{1}{2} \bar{b_i}^2. \end{aligned} \tag{77}$$

Theorem 10.5 *Let $P = \{x \in \mathbb{R}^n | Ax \leq b\}$, where A is an $m \times n$ matrix, and $b \in \mathbb{R}^m$. Let $\mu \in \operatorname{int} P$. Suppose that Q solves* (77)*, where $\bar{b} = b - A\mu$. Then the ellipsoid $E = \{x \in \mathbb{R}^n | (x-\mu)^T Q (x-\mu) \leq 1\}$ has the maximum volume over all ellipsoids centered at μ and contained in 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 $A_i d = b_i$. Let this minimum be $d^{(i)} = \arg\min_{A_i d = \bar{b}_i} f(d)$ for $i = 1, \ldots, m$. By the first order optimality conditions, there exists a $\lambda \in \mathbb{R}^m$ such that

$$\nabla f(d^{(i)}) = Q d^{(i)} = \lambda_i A_i \Longrightarrow d^{(i)} = \lambda_i Q^{-1} A_i \quad \forall 1 \leq i \leq m$$

We also know that

$$A_i^T d^{(i)} = \bar{b_i} \Longrightarrow A_i^T \lambda_i Q^{-1} A_i = \bar{b_i} \Longrightarrow \lambda_i = \frac{\bar{b_i}}{A_i^T Q^{-1} A_i}$$

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

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

Because the volume of the ellipsoid is proportional to the determinant of Q^{-1}, the maximal ellipsoid is defined by (77). □

10.8 Choosing the ellipsoid center

The most obvious choice for the center of the ellipsoid is to choose $\mu^k = x^{(k)}$ (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 ellipsoid. We therefore, in this section, explore strategies for moving the center of the ellipsoid away from the boundary.

10.8.1 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: $T_{\text{out}}^{(k)} = B_2(x^{(k)}, \min\{\Delta_k, \min_i \frac{|A_i x^{(k)} - b_i|}{\|A_i\|}\})$. 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 8. 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.

10.8.2 Ellipsoids

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

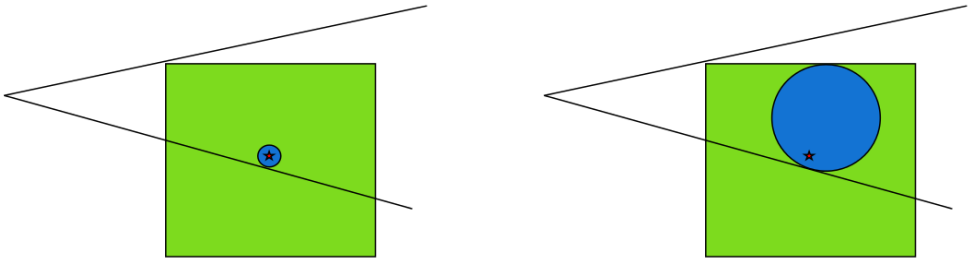

Figure 8: When the current iterate is too close to a constraint, the circular trust region becomes too small. Shifting the trust region center helps remedy this. The star is the current iterate, the green is the outer trust region, and blue the inner.

figure Figure 9, we have this type of iterate, but by using an ellipsoid we are still able to search towards the vertex of the feasible region.

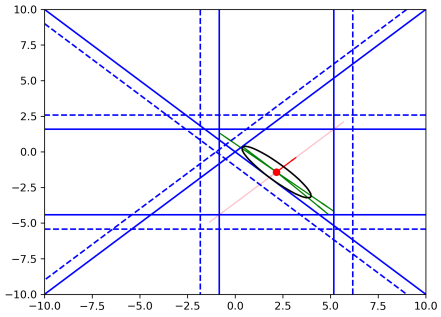

Figure 9: A nicer trust region

More specifically, at iteration k, we choose a scaling factor π^k and solve for an ellipsoid center μ^k and positive definite matrix $Q^{(k)}$ to define an ellipsoid $E^{(k)} = \{x \in \mathbb{R}^n \| \pi^k - \frac{1}{2}(x-\mu^k)^T Q^{(k)}(x-\mu^k) \geq 0\}$. Of course, the simplest approach is to not change the center of the ellipsoid, but instead let $\mu^k = x^k$.

10.8.3 Search Everything

One approach is to search all possible centers within $\mathcal{F} \cap T_{\text{out}}^{(k)}$. That is, we solve:

$$\mu^k = \sup_{\mu \in \mathcal{F} \cap T_{\text{out}}^{(k)}} V(\mu)$$

where $V(\mu)$ is the volume of the ellipsoid defined in (77). This has the advantage that it captures much of the feasible region. However, one problem with this search is that it can force the trust region away from the

desired direction. Notice that in Figure 10, although the ellipsoid found has larger volume than before being shifted, this ellipsoid contains points farther from the corner containing the minimizer.

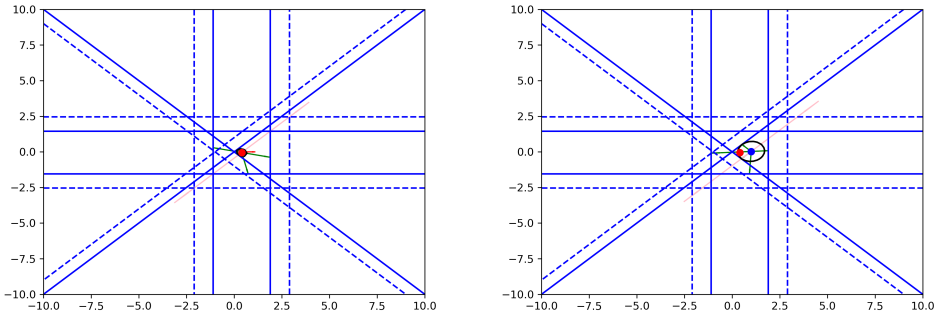

Figure 10: Searching $\mathcal{F}$

One attempt to fix this problem is by limiting the search direction for the center of the ellipsoid.

10.8.4 Line Searches

Although ${m_f}^{(k)}$'s minimizer over $T_{\text{out}}^{(k)}$ can appear anywhere, there are some reasons for expecting it to be at a "vertex." If it lies in the interior, there is little need for using constrained approaches once near the solution.

One way of trying to ensure a feasible direction towards a vertex, while still allowing a larger volume ellipsoid, is by limiting the search for the new center to lie on line segments starting at the current iterate $x^{(k)}$.

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 11, as we should avoid letting the new center get closer to another constraint:

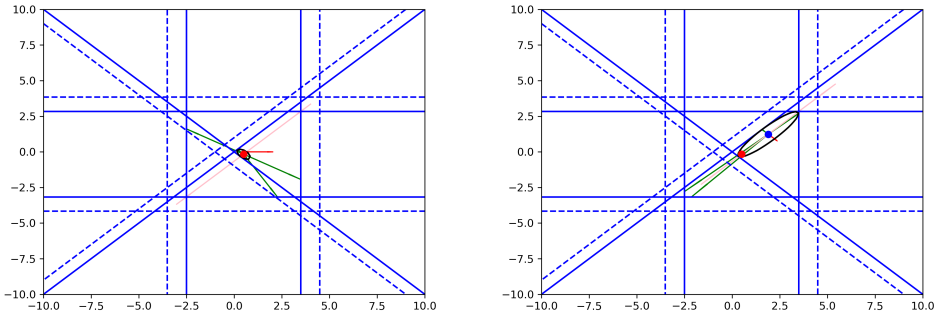

Figure 11: Line searches

For a given distance d, let the indices i for which $\frac{|A_i x - b|}{\|A_i\|} \leq d$

To fix this, we break the search space within the *ConstructtrustRegion* subroutine into segments based on the nearest constraints. The algorithm works by choosing a set of up to n_{points} points $s_1, s_2, \ldots, s_{n_{\text{points}}}$ that are each equidistant to a subset of the constraint's faces. 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: $s_1 = x^{(k)}$. The algorithm then repeats the following process for i from 1 to n_{points}. First, compute the set of nearest constraints, where the distance from a point x to a constraint A_i is given by $d(A_i, x) = \frac{|A_i x - b|}{\|A_i\|}$. While finding the next point s_{i+1}, let A_E be a normalized array of the equidistant faces $\{\frac{A_i}{\|A_i\|} | d(A_i, s_i) = \min_j d(A_j, s_i), i = 1, 2, \ldots, m\}$ and b_E be the rows' corresponding values of b. All other faces are called the remaining faces, and construct the matrix A_R and vector b_R. It then finds a search direction $p = r{A_E}^T$ as a linear combination of the normal vectors to the equidistant faces. This search ray can be found by setting the slack to each equidistant face to a vector of all ones: $A_E(s_i + r{A_E}^T) - b_E = 1$. We can travel along this ray until we reach a point that is the same distance to a remaining face. Specifically, we can travel by

$$t = \arg\min_j \frac{d(A_{E0}, s_i) - d(A_{Rj}, s_i)}{A_{Rj} - d(A_{E0})p} | (A_{Rj} - d(A_{E0})p > 0. \tag{78}$$

We can then set $s_{i+1} = s_i + tp$.

Of course, n_{points} must be less than or equal to $n + 1$ in order for this to be defined. Also, the algorithm must stop early if A_E contains parallel faces.

This means that we can define a class of searches that each limit the number of line segments to search n_{points}.

In figure Figure 12, the red line shows the line segments equidistant their closest constraints. Notice that with two line segments, the algorithm can already choose new centers further from the vertex.

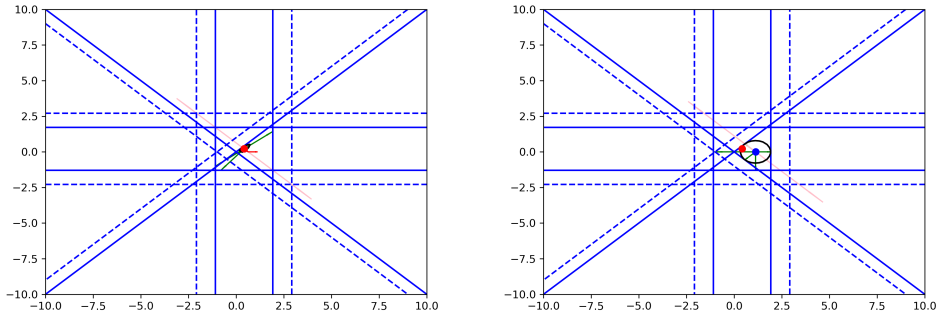

Figure 12: Ellipse runs away from the optimizer

==========

This section describes strategies for choosing the center μ^k for the ellipsoid E^k. At issue is the fact that if the center of the ellipsoid is too close to the boundary of the feasible region, then the condition number of Q^k may be unacceptably large. We there There are several issues at play:

- The ellipsoid should include the current iterate (or maybe not).
- We want the ellipsoid

10.8.5 Ellipsoid Choices

There are a number of issues to be solved to define this ellipsoid:

- How do we ensure that $x^{(k)} \in E^{(k)}$?
- How do we choose $E^{(k)}$ in such a way that it does not limit travel along a decent direction?
- How do we choose the center of the ellipsoid μ^k?

If $x^{(k)} \notin T^{(k)}_{\text{search}}$, the ellipse may not even contain a point with any reduction. 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.
- Expand the size of the ellipsoid.

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

$$\pi^k - \frac{1}{2}(x^k - \mu^k)^T Q^{(k)}(x^k - \mu^k) \geq 0.$$

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

Increase the size. An alternative is to scale $Q^{(k)}$ by a constant. We use the scaling factor π^k defined by

$$\pi^{(k)} = \max\{1, \frac{1}{2}(x^k - \mu^k)^T Q^{(k)}(x^k - \mu^k)^T\}$$

and let the ellipsoid be:

$$E^{(k)} = \{x \in \mathbb{R}^n | 1 - \frac{1}{2\pi^k}(x - \mu^k)^T Q^{(k)}(x - \mu^k) \geq 0\}$$

However, this means that in general $E^{(k)} \not\subset \mathcal{F}$ so that the trust region subproblem must contain constraints for both the ellipsoid and the feasible region: $T^{(k)}_{\text{search}} = E^{(k)} \cap \Omega$.

To help mitigate the second issue, we maximize the volume of the ellipsoid. However, the choice of ellipsoid center can still limit travel along a decent direction. Choosing the best center is the topic of the next section.

10.9 Algorithm Components

Before describing the algorithm, we discuss several components referenced within an algorithm template.

10.9.1 Criticality Measure

In order to define stopping criteria for the algorithm, we introduce a criticality measure χ which goes to zero as the iterates approach a first order critical point. When the criticality measure is small, we must also decrease the trust region radius. Once this has reached a small enough threshold τ_χ and the trust region is small enough ($\Delta_k < \tau_\Delta$), we can terminate the algorithm. For now, our algorithm is designed to work with convex constraints, so we employ a classic criticality measure discussed in [**?**] of

$$\chi_m^{(k)} = \|x^{(k)} - \text{Proj}_{\mathcal{F}}(x^{(k)} - \nabla {m_f}^{(k)}(x^{(k)}))\| \tag{79}$$

The first order optimality conditions for $x^\star \in \mathbb{R}^n$ to by a local optimum of f is that $x^\star$ satisfies

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

For linear constraints, this condition is necessary and sufficient. Thus, our criticality measure measures how far the current iterate is from satisfying the first order optimality conditions for $x^{(k)}$ to be a optimum of ${m_f}^{(k)}$. In turn, as $\Delta_k \to 0$, the model ${m_f}^{(k)}$ better approximates f and $x^{(k)}$ approaches an optimum of f.

10.9.2 Assessing Model Accuracy and Radius Management

Each iteration that evaluates a trial point must also test the accuracy of the model functions. To test the accuracy, we calculate a quantity

$$\rho_k = \frac{f(x^{(k)}) - f(x^{(k)} + s^{(k)})}{{m_f}^{(k)}(x^{(k)}) - {m_f}^{(k)}(x^{(k)} + s^{(k)})} \tag{80}$$

which measures the actual improvement over the predicted improvement. A small ρ_k implies the model functions are not sufficiently accurate. Values of ρ_k close to 1 imply that the model accurately predicted the new objective value. A large ρ_k implies progress minimizing the objective although the model was not accurate. This has been widely used within trust region frameworks such as [74] and within a derivative free context [**?**]. The user supplies fixed constants $0 < \gamma_{\text{min}} \leq \gamma_{\text{suff}} \leq 1$ as thresholds on ρ_k and $0 < \omega_{\text{dec}} < 1 \leq \omega_{\text{inc}}$ as decrement or increment factors to determine the trust region update policy.

10.10 Trust Regions

Our algorithm maintains up to three trust regions. The outer trust region is an L_1 ball of radius Δ_k defined by

$$T_{\text{out}}^{(k)} = B_\infty(x^{(k)}, \Delta_k) = \{x \in \mathbb{R}^n |\ x_i^{(k)} - \Delta_k \leq x_i \leq x_i^{(k)} + \Delta_k \quad \forall 1 \leq i \leq n\}. \tag{81}$$

Note that the outer trust region may include infeasible points. To ensure feasibility of all sample points, we construct an inner trust region for sample points $T_{\text{sample}}^{(k)}$ satisfying $T_{\text{sample}}^{(k)} \subset T_{\text{out}}^{(k)} \cap \mathcal{F}$ and $x^{(k)} \in T_{\text{sample}}^{(k)}$. However, we do not want to limit the search for a new iterate to the same trust region we use to construct the model. This means we introduce another trust region $T_{\text{search}}^{(k)}$ that also satisfies $T_{\text{search}}^{(k)} \subset T_{\text{out}}^{(k)} \cap \mathcal{F}$ and $x^{(k)} \in T_{\text{search}}^{(k)}$ for the trust region subproblem.

The Sample Region We consider general strategies for constructing $T_{\text{sample}}^{(k)}$ and $T_{\text{search}}^{(k)}$.

1. Take

$$T_{\text{sample}}^{(k)} = T_{\text{search}}^{(k)} = T_{\text{out}}^{(k)} \cap \mathcal{F}. \tag{82}$$

   This results in a polyhedral trust region, so we refer to this approach as the *Polyhedral Trust Region Approach*.

2. Force

$$T_{\text{sample}}^{(k)} \subseteq T_{\text{search}}^{(k)} \subseteq T_{\text{out}}^{(k)} \cap \mathcal{F} \tag{83}$$

   where $T_{\text{sample}}^{(k)}$ has an ellipsoidal shape. This is referred to as the *Ellipsoidal Trust Region Approach*

The advantage of the ellipsoidal trust region approach (83) is that we can reuse classical methods for ensuring good geometry. We can construct $T_{\text{sample}}^{(k)}$ to be ellipsoidal and use efficient algorithms within [**?**] to satisfy (75). However, we must be careful while choosing $T_{\text{search}}^{(k)}$ to allow sufficient reduction when we solve the trust region subproblem using the inner trust region. The search trust region is used while selecting the next iterate:

$$s^{(k)} = \underset{s^{(k)} \in T_{\text{search}}^{(k)}}{\arg\min}\ {m_f}^{(k)}(x^{(k)} + s^{(k)}).$$

The Search Region When using the ellipsoidal trust region approach, we have two choices for this trust region. We can take

$$T_{\text{search}}^{(k)} = T_{\text{sample}}^{(k)} \tag{84}$$

or

$$T_{\text{search}}^{(k)} = T_{\text{out}}^{(k)} \cap \mathcal{F}. \tag{85}$$

Namely, in (84) with an ellipsoidal inner trust region, we still have an option to select our trial point from the entire $s^{(k)} \in T_{\text{out}}^{(k)} \cap \mathcal{F}$.

To complete the polyhedral trust region approach (82), we need some redefinition of poisedness for polyhedral shapes. However, since the trust region is larger, it is easier to ensure sufficient reduction. This strategy has the drawback that it will yield sample points close to the boundary of the feasible region. This may cause more infeasible evaluation attempts when we use models to approximate black-box constraints this may.

The classical methods for ensuring good geometry require an optimization call to the model functions over a sphere. This is no longer possible in the polyhedral trust region approach. However, the bounds produced over the entire trust region may also be stronger than required as the models will only be used on the feasible region. This may mean the geometric requirements can be reduced.

Within our algorithm, if $T_{\text{out}}^{(k)} \subseteq \mathcal{F}$ we can set $T_{\text{sample}}^{(k)}$ to be a sphere. This saves the computation of $T_{\text{sample}}^{(k)}$ when it is not needed, as there are no nearby constraints.

10.11 Algorithms

10.11.1 Sufficient Model Reduction

To ensure sufficient reduction of the objective's model function during each iteration, we impose the following efficiency condition:

$$m_f{}^{(k)}(x^{(k)}) - m_f{}^{(k)}(x^{(k)} + s^{(k)}) \geq \kappa_f \chi_k \min\left\{ \frac{\chi_k}{1 + \|\nabla^2 m_f{}^{(k)}(x^{(k)})\|}, \Delta_k, 1 \right\} \tag{86}$$

where κ_f is a constant independent of k. This is widely used within trust region frameworks such as [**?**] and [74]. It can be shown that the *generalized Cauchy point* satisfies this condition [74].

10.12 Algorithm Template

We follow an algorithm template described in [**?**], where variations of the algorithm have different choices of $T^{(k)}_{\text{sample}}$ implemented in a *ConstructTrustRegion* subroutine. The different versions are described in the remainder of this section.

Algorithm 7: Always-feasible Constrained Derivative Free Algorithm

Step 0 (Initialization)
Initialize tolerance constants $\tau_\xi \geq 0$, $\tau_\Delta \geq 0$, starting point $x^{(0)} \in \mathcal{F}$, initial radius $\Delta_0 > 0$, iteration counter $k = 0$, $0 < \omega_{\text{dec}} < 1 \leq \omega_{\text{inc}}$, $0 < \gamma_{\text{min}} < \gamma_{\text{suff}} \leq 1$, $\alpha > 0$, $k \leftarrow 1$, $0 < \omega_{\text{dec}} < 1 \leq \omega_{\text{inc}}$, $0 < \gamma_{\text{min}} < \gamma_{\text{suff}} < 1$.

Step 1 (Construct the model)
$T^{(k)}_{\text{sample}} \leftarrow$ CONSTRUCTTRUSTREGION($\Delta_k, x^{(k)}$). Ensure that the sample points are poised with respect to $T^{(k)}_{\text{sample}}$ for (75) by calling Algorithm 6. Construct $m_f{}^{(k)}$ as described in (70) to construct $m_f{}^{(k)}(x)$.

Step 2 (Check stopping criteria)
Compute χ_k as in (79).

If $\chi^{(k)}_m < \tau_\xi$ and $\Delta_k < \tau_\Delta$ then return $x^{(k)}$ as the solution.

Otherwise, if $\Delta_k > \alpha \chi^{(k)}_m$ then $\Delta_{k+1} \leftarrow \omega_{\text{dec}} \Delta_k$, $x^{(k+1)} \leftarrow x^{(k)}$, $k \leftarrow k + 1$ and go to Step 1.

Step 3 (Solve the trust region subproblem)
Compute $s^{(k)} = \min_{s \in T^{(k)}_{\text{search}}} m_f{}^{(k)}(x^{(k)} + s^{(k)})$.

Step 4 (Test for improvement)
Evaluate $f(x^{(k)} + s^{(k)})$ and evaluate ρ_k as in (80)

If $\rho_k < \gamma_{\text{min}}$ then $x^{(k+1)} = x^{(k)}$ (reject) and $\Delta_{k+1} = \omega_{\text{dec}} \Delta_k$

If $\rho_k \geq \gamma_{\text{min}}$ and $\rho < \gamma_{\text{suff}}$ then $x^{(k+1)} = x^{(k)} + s^{(k)}$ (accept), $\Delta_{k+1} = \omega_{\text{dec}} \Delta_k$

If $\rho_k > \gamma_{\text{suff}}$ then $x^{(k+1)} = x^{(k)} + s^{(k)}$ (accept), $\Delta_{k+1} = \omega_{\text{inc}} \Delta_k$

$k \leftarrow k + 1$ and go to Step 1.

Much of the work is deferred to the *ConstructTrustRegion* subroutine. We will describe several different approaches for this subroutine.

10.13 Polyhedral Trust Region Approach

One simple approach to handle partially-quantifiable constraints is to maintain the same trust region as the classical algorithm, but avoid letting points fall outside the feasible region within the model improvement algorithm Algorithm 6. That is, we add the constraints $m_{c_i}^{(k)}(x) \leq 0 \forall i \in \mathcal{I}$ and $m_{c_i}^{(k)}(x) = 0 \forall i \in \mathcal{E}$ to the model improvement algorithm while selecting new points. This constrains the new points to also lie within the current model of the trust region in Algorithm 6, Step 2. The search space for this optimization problem will be the feasible region intersect the trust region: $\mathcal{F} \cap T_{\text{out}}^{(k)}$.

The challenge lies in finding sufficiently poised sample points. Note that Algorithm 6 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 optimized over a sphere. However, when requiring points to live within $\mathcal{F} \cap T_{\text{out}}^{(k)}$, it can happen that even after replacing a point, we still have not satisfied this bound. In Figure 13, for some values of ξ_{min}, there is no point in $\mathcal{F} \cap T_{\text{out}}^{(k)}$ that will leave a sufficiently large pivot.

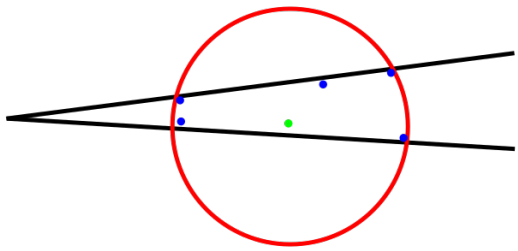

Figure 13: Limited sample point choice

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}. The new modified improvement algorithm is described in Algorithm 8:

Algorithm 8: Modified Model Improvement Algorithm

Step 0 (Initialization)
Initialize $i = 1$. If the current sample set does not have p points, repeat one of the current points. Construct the Vandermonde matrix $V_{i,j} = \phi_j(\frac{1}{\Delta}(y^i - y^0))$. Initialize $0 < \xi_{\text{min}} < \xi_{\text{desired}}$, $0 < \delta_{\text{improv}} < 1$, $\xi_{\text{cur}} = \xi_{\text{desired}}$.

Step 1 (Pivot)
Swap row i with row $i_{\max} = \arg\max_{j|j \geq i} V_{j,i}$

Step 2 (Check threshold)

If $|V_{i,i}| \geq \xi_{\text{cur}}$ then go to Step 3

$\hat{y} = \arg\max_{t \in T^{(k)}_{\text{sample}} \cap X} |\phi_i(t)|$

If $|\phi_i(\hat{y})| < \xi_{\text{min}}$ then **Stop**: the algorithm failed

If $\xi_{\text{cur}} - |\phi_i(\hat{y})| > \delta_{\text{improv}} \xi_{\text{cur}}$ then replace $V_{i,j}$ with $\phi_j(\hat{y})$ and $\xi_{\text{cur}} \leftarrow |\phi_i(\hat{y})|$

Step 3 (LU)

Set $V_i \leftarrow \frac{1}{V_{i,i}} V_i$

Set $V_{,j} \leftarrow V_{,j} - V_{i,j} V_{\bullet,i} \forall j = i \ldots p$

$i \leftarrow i + 1$ Go to step 1 unless $i > p$

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 is usual, we may also wish to remove points larger than a certain radius from the current model center.

10.14 Ellipsoidal Trust Region Approach

If we adopt the ellipsoidal trust region approach to maintain a feasible inner trust region with a "nice" shape we ensure of a stronger version of (75). Namely, we know from Section 10.6 that

$$\|m_f{}^{(k)}(x) - \nabla f(x)\| \leq \kappa_g \Delta_k \quad \forall x \in T^{(k)}_{\text{search}}.$$

If we also choose our trial point with (84), we have no guarantee of satisfying the efficiency condition (86) because Δ_k is the outer trust region radius. However, the model will likely be more accurate over this region.

10.15 Convergence Discussion

10.16 Algorithm Assumptions

Here, we show convergence for one version of Algorithm 7. Namely, we choose to satisfy (83) and let $T^{(k)}_{\text{sample}}$ have an ellipsoidal shape as described in Section 10.10. Also, we will select (85) as discussed in Section 10.10: namely, $T^{(k)}_{\text{search}} = T^{(k)}_{\text{out}} \cap \mathcal{F}$. To do this, we require the following assumptions.

Here, we will let Ω be some open set containing the feasible region: $\mathcal{F} \subset \Omega$.

Assumption 10.1 *The function f is differentiable and its gradient ∇f is Lipchitz continuous with constant $L_g > 0$ in Ω. That is,*

$$\|\nabla f(x) - \nabla f(y)\| \leq L\|x - y\| \quad \forall x, y \in \Omega. \tag{87}$$

Assumption 10.2 *The function f has Lipschitz continuous hessian with constant $L_h > 0$ in Ω. That is,*

$$\|\nabla^2 f(x) - \nabla^2 f(y)\| \leq L_h\|x - y\| \quad \forall x, y \in \Omega. \tag{88}$$

Assumption 10.3 *The function f is bounded below over Ω. That is,*

$$f(x) \geq f_{min} \quad \forall x \in \Omega. \tag{89}$$

Assumption 10.4 *The Hessian's of ${m_f}^{(k)}$ are uniformly bounded at each iterate. That is, there exists a constant $\beta \geq 1$ such that*

$$\|\nabla^2 {m_f}^{(k)}(x^{(k)})\| \leq \beta - 1 \quad \forall k \geq 0. \tag{90}$$

Assumption 10.5 *There exists a point $\bar{x}$ within the interior of the feasible region:*

$$G\bar{x} < g. \tag{91}$$

10.17 Required Assumptions

If the tolerances $\tau_\chi = 0$ and $\tau_\Delta = 0$ are set to zero, Algorithm 7 is a particular implementation of the algorithm presented in [**?**]. This means we only need to satisfy the requirements detailed in their convergence analysis. For your convenience, we duplicate these assumptions.

H_0 The efficiency condition (86) is satisfied.

H_1 The function f is differentiable and its gradient ∇f is Lipchitz continuous with constant $L > 0$ in Ω.

H_2 The function f is bounded below over Ω.

H_3 The matrices H_k are uniformly bounded. That is, there exists a constant $\beta \geq 1$ such that $\|\nabla^2 {m_f}^{(k)}\| \leq \beta - 1$ for all $k \geq 0$.

H_4 There exists a constant δ_g such that $\|\nabla m_k(x^{(k)}) - \nabla f(x^{(k)})\| \leq \delta_g \Delta_k$ for all $k \geq 0$.

H_0 can be satisfied within our algorithm by selecting the Generalized Cauchy Point [74] to solve the trust region subproblem. Notice that H_1, H_2, and H_3 are kept as hypothesis within our algorithm. This leaves H_4, which is the topic of Section 10.18.2. However, first we show a way of using a more straightforward version of H_3 in Section 10.18.1.

10.18 Satisfying these assumptions

10.18.1 More simple H_3

First, we show that Assumption 10.7 can be used instead of Assumption 10.4 under some restrictions. Namely, for this to be true, we also first assume:

Assumption 10.6 *There exists a $\Delta_{max} > 0$ such that $\Delta_k \leq \Delta_{max}$ for all $k \geq 0$.*

With this modest assumption, we can make the assumptions more straightforward by replacing Assumption 10.4 with Assumption 10.7:

Assumption 10.7 *The Hessian's of f are uniformly bounded at each iterate. That is, there exists a constant $\beta \geq 1$ such that $\|\nabla^2 f\| \leq \beta - 1$ for all $k \geq 0$*

This is the result of the following lemma:

Lemma 10.6 *Assume that Assumption 10.7, Assumption 10.2,* **??***, and Assumption 10.6 are satisfied and that each $k \in \mathbb{N}$, ${m_f}^{(k)}$ is a quadratic model of f over $B_\infty(x^{(k)}, \Delta_k)$ as in Theorem 10.1. Then Assumption 10.4 is also satisfied.*

Proof:

Let $\beta_1 \geq 1$ be such that for all $k \geq 0$:

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

Because ${m_f}^{(k)}$ are fully quadratic, we know that (72) is satisfied. Combining this with Assumption 10.6 we see that

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

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

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

□

10.18.2 Satisfying the Accuracy Assumption

The only remaining assumption is the accuracy condition H_4.

As discussed in Section 10.6, we know that if the condition number of $Q^{(k)}$ is bounded, we can map a poised set over the unit ball to $T^{(k)}_{\text{sample}}$. However, we must also ensure that we are always able to find a feasible ellipsoid. Although it is not always possible to find a feasible ellipsoid that contains the current iterate, we can find a feasible ellipsoid that only needs to be scaled by a constant to do so. This ellipsoid must satisfy the following conditions:

Definition 10.7 *An ellipsoid E_k determined by a positive definite, symmetric matrix $Q^{(k)}$, a center $c^{(k)} \in \mathbb{R}^n$, and a radius ϵ_k is said to be a* ***suitable ellipsoid*** *for iteration k if all of the following are satisfied:*

1. *The ellipsoid $E_k = \{x | (x-c)^T Q (x-c) \leq \frac{1}{2}\epsilon^2\}$ satisfies $E_k \subseteq P_k$ as defined in* (92)

2. *The ellipsoid* $\hat{E}_k = \{x|(x-c)^T Q(x-c) \leq \epsilon^2\}$ *satisfies* $x^{(k)} \in \hat{E}_k$.

3. $\sigma(Q^{(k)})$ *is bounded independently of* k.

Because the ellipsoid we construct must be feasible with respect to both the trust region and the constraints, we simplify notation by naming

$$P_k = \{x \in \mathbb{R}^n| \quad Gx \leq g, \|x - x^{(k)}\|_\infty \leq \Delta_k\} = \{x \in \mathbb{R}^n | \nabla m_c^{(k)}\left(x^{(k)}\right)x \leq m_c^{(k)}\left(x^{(k)}\right), \|\nabla m_c^{(k)}\left(x^{(k)}\right)_i\| = 1\}. \tag{92}$$

This is the normalized polyhedron formed by adding the trust region constraints for iteration k to the problem constraints. The active constraints at any point will be denoted by

$$\mathcal{A}(x) = \{1 \leq i \leq m | c_i(x) = 0 \Leftrightarrow G_i x = g_i\} \tag{93}$$

It will be convenient to use a feasible direction with respect to the active constraints at a given point. To this end, let $\mathcal{S} \subseteq \{1, \ldots, m\}$ be arbitrary, and define

$$u_{\text{feasible}}(\mathcal{S}) = \begin{cases} \arg\max_{\|u\|=1} \min_{i \in \mathcal{S}} -u^T G_i & \text{if } \mathcal{S} \neq \emptyset \\ \emptyset & \text{if } \mathcal{S} = \emptyset \end{cases} \tag{94}$$

$$\pi_1(\mathcal{S}) = \begin{cases} \max_{\|u\|=1} \min_{i \in \mathcal{S}} -u^T G_i & \text{if } \mathcal{S} \neq \emptyset \\ 1 & \text{if } \mathcal{S} = \emptyset \end{cases} \tag{95}$$

$$\pi_2(x) = \pi_1\left(\mathcal{A}(x)\right) \tag{96}$$

$$\alpha_k = \pi_2\left(x^{(k)}\right) \tag{97}$$

$$\hat{u}^{(k)} \in u_{\text{feasible}}\left(\mathcal{A}(x^{(k)})\right) \tag{98}$$

Here, π_2 is a set of directions that hopefully head away from each of the active constraints at a point.

Throughout, we will use a couple different cones, which can be described here

$$\mathcal{C}(\alpha, d, c) = \left\{x \in \mathbb{R}^n| \quad x = c + td + s, s^T d = 0, t \geq 0, \|s\| \leq \alpha t\right\} \tag{99}$$

$$C_{\text{unsh}}^{(k)} = \mathcal{C}(\alpha_k, \hat{u}^{(k)}, x^{(k)}) = \{x \in \mathbb{R}^n| \quad x = x^{(k)} + t\hat{u}^{(k)} + s, s^T u^\star = 0, t \geq 0, \|s\| \leq \alpha_k t\} \tag{100}$$

$$C_{\text{sh}}^{(k)} = \mathcal{C}(\alpha_k, e_1, 0) = \{x = (t, s)^T \in \mathbb{R}^n, t \in \mathbb{R}_{\geq 0}, s \in \mathbb{R}^{n-1}| \quad \|s\| \leq \alpha_k t\} \tag{101}$$

We will use the following mapping to go between these cones:

$$R^{(k+1)} = 2\frac{(e_1 + \hat{u}^{(k)})(e_1 + \hat{u}^{(k)})^T}{(e_1 + \hat{u}^{(k)})^T(e_1 + \hat{u}^{(k)})} - \boldsymbol{I} \tag{102}$$

$$T_k(x) = R^{(k+1)}\left(x - x^{(k)}\right) \tag{103}$$

Finally, the following is a useful function for defining ellipsoids:

$$f_e(\alpha, \delta, r; x) = (x - \delta e_1)^T \begin{bmatrix} 1 & \mathbf{0}^T \\ \mathbf{0} & \alpha^{-2}\boldsymbol{I} \end{bmatrix} (x - \delta e_1) - r \tag{104}$$

Lemma 10.8 *Assume that Assumption 10.5 is satisfied and $x_0 \in \mathcal{F}$. Then $1 \geq \pi_2(x_0) > 0$.*

Proof:

Let $x_0 \in \mathcal{F}$. If $\mathcal{A}(x_0) = \emptyset$, then $\alpha_k = 1 > 0$. Otherwise, let $i \in \mathcal{A}(x_0)$, so that $c_i(x_0) = 0$. Because c_i is convex, we know

$$c_i(\bar{x}) \geq c_i(x_0) + \nabla c_i(x_0)^T(\bar{x} - x_0) \Longrightarrow \nabla c_i(x_0)^T(\bar{x} - x_0) \leq c_i(\bar{x}) - c_i(x_0) = c_i(\bar{x}) < 0$$

Using **??**, we can write this as

$$-G_i^T \frac{\bar{x} - x_0}{\|\bar{x} - x_0\|} > 0 \Longrightarrow \min_{i \in \mathcal{A}(x_0)} -G_i^T \frac{\bar{x} - x_0}{\|\bar{x} - x_0\|} > 0.$$

Using this along with definitions (94), (95), and (96) we see

$$\pi_2(x_0) = \pi_1\left(\mathcal{A}(x_0)\right) = \max_{\|u\|=1} \min_{i \in \mathcal{A}(x_0)} -G_i^T u \geq \min_{i \in \mathcal{A}(x_0)} -G_i^T \frac{\bar{x} - x_0}{\|\bar{x} - x_0\|} > 0.$$

We know that $\pi_2(x_0) \leq 1$ because it is the dot product of two vectors of length one: if $\|u\| = 1$, then $\left|u^T G_i\right|^2 \leq \|u\|\|G_i\| = 1$ by Cauchy–Schwarz.
□

Lemma 10.9 *If Assumption 10.5 is satisfied, then there exists an $\epsilon_\alpha > 0$ such that $\alpha_k \geq \epsilon_\alpha \; \forall \; k \in \mathbb{N}$.*

Proof:

Because there are only m constraints, each $\mathcal{A}(x)$ is one of the only 2^m subsets of $\{1, 2, 3, \ldots, m\}$. This means that u_{feasible}, π_1, and α_k can only take on at most $1 + 2^m$ values. By Theorem 10.8, 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. □

Lemma 10.10 *If $\mathcal{A} = \emptyset$ during iteration k, then there exists a suitable ellipsoid for iteration k.*

Proof:

If $\mathcal{A} = \emptyset$, then we are free to select $c = x_0, Q = I$, and ϵ smaller than the distance to the nearest constraint: $\epsilon \leq \min_{1 \leq i \leq m} b_i^{(k)} - \left(\nabla m_c^{(k)}\left(x^{(k)}\right)_i\right)^T x_0$. Because E_k is then a sphere with radius less than the distance to the nearest constraint, $E \subseteq P$. Because the sphere $\hat{E}_k$ is centered at x_0, $x_0 \in \hat{E}_k$. Also, $\sigma(Q) = 1$. □

Lemma 10.11 *The set $C_{unsh}^{(k)}$ defined in (100) is feasible with respect to the active constraints of P_k at $x^{(k)}$.*

Proof:

Note that the trust region boundary cannot be active at $x^{(k)}$ as $\Delta_k > 0$. Let $y = x^{(k)} + t\hat{u}^{(k)} + s \in C_{\text{unsh}}^{(k)}$ and $i \in \mathcal{A}(x^{(k)})$ be arbitrary. Then,

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

□

Lemma 10.12 *Let $C_{sh}^{(k)}, f_e, \alpha_k$ be defined as in* (101)*,* (104)*,* (97)*. Then the ellipsoid*

$$\left\{x \in \mathbb{R}^n | f_e(\alpha_k, \delta, \frac{1}{2}\delta^2; x) \leq 0\right\} \subseteq C_{sh}^{(k)} \quad \forall \delta > 0 \tag{105}$$

Proof:

Suppose that $x \in \left\{x \in \mathbb{R}^n | f_e(\alpha_k, \delta, \frac{1}{2}\delta^2; x) \leq 0\right\}$, then

$$f_e(x) \leq 0 \Longrightarrow (x - \delta e_1)^T \begin{bmatrix} 1 & \mathbf{0}^T \\ \mathbf{0} & \alpha_k^{-2}\boldsymbol{I} \end{bmatrix} (x - \delta e_1) \leq \frac{1}{2}\delta^2 \Longrightarrow (t-\delta)^2 + \frac{1}{\alpha_k^2}\|s\|^2 \leq \frac{1}{2}\delta^2$$
$$\Longrightarrow \|s\|^2 \leq \alpha_k^2 \left[\frac{1}{2}\delta^2 - (t-\delta)^2\right] = \alpha_k^2 \left[t^2 - 2(t - \frac{1}{2}\delta)^2\right] \leq \alpha_k^2 t^2 \Longrightarrow \|s\| \leq \alpha_k t.$$

Thus, $x \in C_{\text{sh}}^{(k)}$. □

Lemma 10.13 *Let $R^{(k+1)}$, T_k, $C_{unsh}^{(k)}$, $C_{sh}^{(k)}$ be defined as in* (102)*,* (103)*,* (100)*,* (101)*. Then $T_k(C_{unsh}^{(k)}) = C_{sh}^{(k)}$.*

Proof:

Observe that

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

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 $\|s\| \leq \alpha_k t$. 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, $T_k(x) = \begin{bmatrix} t \\ \sigma \end{bmatrix}$ where $\sigma \in \mathbb{R}^{n-1}$ satisfies $\|\sigma\| = \|s\| \leq \alpha t$. Thus, $T_k(x) \in C_{\text{sh}}^{(k)}$. Conversely, if $\begin{bmatrix} t \\ \sigma \end{bmatrix} \in C_{\text{sh}}^{(k)}$, then let $s = {R^{(k+1)}}^T \begin{bmatrix} 0 \\ \sigma \end{bmatrix}$ to see that $x = T_k^{-1}\left(\begin{bmatrix} t \\ \sigma \end{bmatrix}\right) = {R^{(k+1)}}^T \left(te_1 + \begin{bmatrix} 0 \\ \sigma \end{bmatrix}\right) = t\hat{u}^{(k)} + s$ where $\|s\| = \|\sigma\| \leq \alpha t$. Hence $T_k^{-1}\left(\begin{bmatrix} t \\ \sigma \end{bmatrix}\right) \in C_{\text{unsh}}^{(k)}$. □

Lemma 10.14 *Let $R^{(k+1)}$, T_k, $C_{unsh}^{(k)}$, $C_{sh}^{(k)}$, P_k be defined as in* (102)*,* (103)*,* (100)*,* (101)*,* (92)*.*
For each iteration k, there exists a $\delta_f > 0$ such that the ellipsoid

$$\mathcal{E}_{feasible}^k = \left\{x \in \mathbb{R}^n | f_e(\delta_f, \frac{1}{2}\delta_f^2, \alpha_k, T_k(x)) \leq 0\right\} \tag{106}$$

satisfies $\mathcal{E}_{feasible}^k \subseteq P_k$.

Proof:

Let L be the shortest distance from $x^{(k)}$ to any point on a non-active constraint. Define $\alpha' = \sqrt{(1+\alpha_k^2)\left(1+\frac{1}{\sqrt{2}}\right)}$, and let $\delta_f = \frac{1}{\alpha'}L$. We see that if $x \in \mathcal{E}^k_{\text{feasible}}$, then by Theorem 10.12 we have that $T_k(x) = \begin{bmatrix} t \\ \sigma \end{bmatrix} \in C^{(k)}_{\text{sh}}$ for some $\sigma \in \mathbb{R}^{n-1}$, and

$$(x-\delta_f e_1)^T \begin{bmatrix} 1 & \mathbf{0}^T \\ \mathbf{0} & \alpha_k^{-2}\boldsymbol{I} \end{bmatrix} (x-\delta_f e_1) \leq \frac{1}{2}\delta_f^2$$

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

so that

$$\|x\|^2 = t^2 + \|\sigma\|^2 \leq \left(1+\alpha_k^2\right)t^2 \leq \left(1+\alpha_k^2\right)\left(1+\frac{1}{\sqrt{2}}\right)\delta_f^2 = \alpha'^2\delta_f^2$$

$$\implies \|x\| \leq \alpha'\delta_f \leq L$$

Thus, all points within $\mathcal{E}^k_{\text{feasible}}$ are closer than the nearest point of a non-active constraint. Combine this with Theorem 10.11 to see that $\mathcal{E}^k_{\text{feasible}} \subseteq P_k$. □

Lemma 10.15 *For some iteration k, let $R^{(k+1)}$, T_k, $C^{(k)}_{unsh}$, $C^{(k)}_{sh}$, P_k be defined as in* (102)*,* (103)*,* (100)*,* (101)*,* (92)*. Also, let $\delta_f > 0$ be defined as in Theorem 10.14.*

$$\hat{\mathcal{E}}^k_{feasible} = \left\{x \in \mathbb{R}^n | f_e(\delta_f, \delta_f^2, \alpha_k, T_k(x)) \leq 0\right\} \tag{107}$$

satisfies $x^{(k)} \in \mathcal{E}^k_{feasible}$.

Proof:

We have that

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

□

Lemma 10.16 *For iteration k, if $\alpha_k > 0$, then there exists a suitable ellipsoid for iteration k.*

Proof:

Let $R^{(k+1)}$, be defined as in (102) . Also, let δ_f, $\hat{\mathcal{E}}^k_{\text{feasible}}$ be defined as in (106) (107) .

Let

$$Q^{(k)} = {R^{(k+1)}}^T \begin{bmatrix} 1 & \mathbf{0}^T \\ \mathbf{0} & \alpha_k^{-2}\boldsymbol{I} \end{bmatrix} R^{(k+1)}$$

$$c_k = x^{(k)} - \delta_f \hat{u}^{(k)}$$

$$\epsilon = \delta^2$$

By Theorem 10.14, we know that $\mathcal{E}^k_{\text{feasible}} \subseteq P_k$. By Theorem 10.15, we know that $x^{(k)} \in \hat{\mathcal{E}}^k_{\text{feasible}}$. The condition number of $\sigma(Q^{(k)}) = \frac{\max\{1,\alpha_k^{-2}\}}{\min\{1,\alpha_k^{-2}\}} = \alpha_k^{-2}$. This is because $\det(R^{(k+1)}) = 1$ means the condition number of $Q^{(k)}$ is not affected $R^{(k+1)}$. □

We use the following result shown [**?**], Corollary 4.7. We restate the theorem here, with the simplication that f is deterministic function:

Assumption 10.8 *Suppose that a set S and a radius Δ_{max} are given. Assume that f is twice continuously differentiable with Lipshcitz continuous Hessian in an open domain containing the Δ_{max} neighborhood $\cup_{x\in S} B(x; \Delta_{max})$ of the set S.*

Lemma 10.17 *Let Y be a poised set of p points, and let $R = \max_i \|y^i - y^0\|$. Let f satisfy Assumption 10.8 over some convex set Ω, and let $m(x)$ denote the quadratic model of f using* (70)*. If f is a Lipschitz continuous function with Lipschitz constant L_g, and $m_f(x)$ is a quadratic model of f. Then, there exist constrants $\Lambda_1, \Lambda_2, \Lambda_3$ independent of R such that for all $x \in B(y^0, R)$,*

$$\begin{aligned}\|f(x) - m_f(x)\| &\leq 4\Lambda_1 R^3 L\sqrt{p+1}\\ \|\nabla f(x) - \nabla m(x)\| &\leq 4\Lambda_2 R^2 L\sqrt{p+1}\\ \|\nabla^2 f(x) - \nabla^2 m(x)\| &\leq 4\Lambda_3 R L\sqrt{p+1}\end{aligned}$$

We can now satisfy the accuracy condition.

Theorem 10.18 *Suppose that Assumption 10.5, Assumption 10.8 hold. The accuracy condition* (75) *is satisfied for each iterate k. Namely, there exists a κ_g such that $\|\nabla m_f(x^{(k)}) - \nabla f(x^{(k)})\| \leq \kappa_g \Delta_k$.*

Proof:

Fix an iterate k. If there are no active constraints, then we can use Theorem 10.10 to construct a suitable ellipsoid. Otherwise, we can use Theorem 10.9 to satisify the conditions of Theorem 10.16. This provides two ellipsoids:

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

As in Theorem 10.4, we can give $Q^{(k)} = LD^2L^T$ its eigen-decomposition, and define $\delta = \max_{x\in\mathcal{E}^k_{\text{feasible}}} \|x - \mu^{(k)}\|$, so the transformation $T(x) = \delta D L^T(x - \mu^{(k)})$ maps $E^{(k)}$ to the δ ball. As also described in Theorem 10.4, we create shifted functions $\hat{m}_f(u) = m_f(T^{-1}(u))$ and $\hat{f}(u) = f(T^{-1}(u))$. After using Algorithm 6 to choose sample points, we know by Theorem 10.1 that there exists a constants $\kappa_f, \kappa_g, \kappa_h$ such that for all $u \in B(0, \delta)$:

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

We can then use Theorem 10.17 to conclude that there is another constant Λ_2 such that for all $u \in B(0, 2\delta)$:

$$\left\| \nabla \hat{f}(u) - \nabla \hat{m}_f(u) \right\| \leq 4\Lambda_2 (2\delta)^2 L\sqrt{p+1} = \kappa'_g \delta^2$$

where $\kappa'_g = 16\Lambda_2 L\sqrt{p+1}$. Notice that Lemma 3.1 and Lemma 3.2 within [**?**] show that $\lim_{k\to\infty} \Delta_k = 0$ without requiring the accuracy condition. This is because Lemma 3.1 assumes that $\Delta_k \leq 1$ explicitly, while Lemma 3.2 uses the accuracy of the model's hessians. This justifies the use of a $\Delta_{\text{max}} > 0$, such that $\Delta_k \leq \Delta_{\text{max}}\ \forall k \in \mathbb{N}$. After using Theorem 10.9 to construct an $\epsilon_\alpha > 0$ and defining $\kappa''_g = \frac{\kappa'_g}{\epsilon_\alpha}\Delta_{\text{max}}$, we see that we can use Theorem 10.4 to conclude that for all $x_0 \in \hat{\mathcal{E}}^k_{\text{feasible}}$:

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

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

Notice that although Theorem 10.18 requires $\delta \leq 1$, the authors of show that $\Delta_k \to 0$ within Lemma 3.1 and Lemma 3.2. Within Lemma 3.1, $\Delta_k \leq 1$ explicitly.

10.19 Results

10.20 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 here:

Shape	Search	Num. Segments	Basis	Iterations	Evaluations		
spherical	anywhere		linear	159	202	470	630
spherical	anywhere		quadratic	164	277	467	805
spherical	none		linear	77*	122*	255	387
spherical	none		quadratic	74	149	250	561
spherical	segment	1	linear	74	116	224	413
spherical	segment	1	quadratic	74	164	224	525
spherical	segment	2	linear	164	223	313	503
spherical	segment	2	quadratic	152	259	313	657
circumscribed ellipsoid	none		linear	41	50	41	55
circumscribed ellipsoid	none		quadratic	41	104	41	105
ellipsoid	anywhere		linear	65	109	67	110
ellipsoid	anywhere		quadratic	66	170	67	185
ellipsoid	none		linear	53	65	50	52
ellipsoid	none		quadratic	53	88	50	75
ellipsoid	segment	1	linear	55	70	58	75
ellipsoid	segment	1	quadratic	55	97	58	104
ellipsoid	segment	2	linear	67	144	68	121
ellipsoid	segment	2	quadratic	67	196	64	159
polyhedral	none		linear	37	43	38	46
polyhedral	none		quadratic	37	82	38	89
scaled ellipsoid	anywhere		linear	66	103	67	104
scaled ellipsoid	anywhere		quadratic	68	156	67	169
scaled ellipsoid	segment	1	linear	44	65	45	67
scaled ellipsoid	segment	1	quadratic	44	94	45	93
scaled ellipsoid	segment	2	linear	67	125	68	122
scaled ellipsoid	segment	2	quadratic	67	189	63	146
simplex	max volume		linear	41	44	41	45
simplex	max volume		quadratic	41	94	41	84

*The spherical trust region with no search for the center did not converge.

In general, the linear models converge more quickly than quadratic models. We see that the method with fewest iterations and function evaluations is the linear polyhedral shape. This is likely due to the fact that 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.

10.21 Schittkowksi Test Problems for Nonlinear Optimization

We tested these algorithms on several problems from the Hot-Schittkowski problem set [**?**], [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 here.

Algorithm	Prob.	n	Message	N. It.	N. Eval.	Ret. Min.	Min.	Solution	Minimizer
circumscribed ellipse	21	2	converged	24	71	-99.960	-99.960	[2.00,0.00]	[2.00,0.00]
ellipse	21	2	converged	36	96	-99.960	-99.960	[2.00,0.00]	[2.00,0.00]
ellipse everywhere	21	2	converged	24	82	-99.960	-99.960	[2.00,0.00]	[2.00,0.00]
ellipse segment 1	21	2	converged	24	84	-99.960	-99.960	[2.00,0.00]	[2.00,0.00]
ellipse segment 2	21	2	converged	24	85	-99.960	-99.960	[2.00,0.00]	[2.00,0.00]
polyhedral	21	2	converged	26	75	-99.960	-99.960	[2.00,-0.00]	[2.00,0.00]
circumscribed ellipse	224	2	failed	16	56	-304.000	-304.000	[4.00,4.00]	[4.00,4.00]
ellipse	224	2	converged	32	97	-304.000	-304.000	[4.00,4.00]	[4.00,4.00]
ellipse everywhere	224	2	converged	24	94	-303.774	-304.000	[3.73,4.27]	[4.00,4.00]
ellipse segment 1	224	2	converged	24	91	-303.774	-304.000	[3.73,4.27]	[4.00,4.00]
ellipse segment 2	224	2	converged	23	89	-303.774	-304.000	[3.73,4.27]	[4.00,4.00]
polyhedral	224	2	converged	17	64	-304.000	-304.000	[4.00,4.00]	[4.00,4.00]
circumscribed ellipse	231	2	converged	14	44	0.000	0.000	[1.00,1.00]	[1.00,1.00]
ellipse	231	2	converged	99	193	0.000	0.000	[1.00,1.00]	[1.00,1.00]
ellipse everywhere	231	2	converged	181	336	0.000	0.000	[1.00,1.00]	[1.00,1.00]
ellipse segment 1	231	2	converged	78	162	0.000	0.000	[1.00,1.00]	[1.00,1.00]
ellipse segment 2	231	2	converged	173	321	0.000	0.000	[1.00,1.00]	[1.00,1.00]
polyhedral	231	2	converged	95	193	0.000	0.000	[1.00,1.00]	[1.00,1.00]
circumscribed ellipse	232	2	failed	15	35	-1.000	-1.000	[3.00,1.73]	[3.00,1.73]
ellipse	232	2	converged	34	91	-1.000	-1.000	[3.00,1.73]	[3.00,1.73]
ellipse everywhere	232	2	converged	41	102	-1.000	-1.000	[3.00,1.73]	[3.00,1.73]
ellipse segment 1	232	2	converged	34	91	-1.000	-1.000	[3.00,1.73]	[3.00,1.73]
ellipse segment 2	232	2	converged	35	95	-1.000	-1.000	[3.00,1.73]	[3.00,1.73]
polyhedral	232	2	converged	15	47	-1.000	-1.000	[3.00,1.73]	[3.00,1.73]
circumscribed ellipse	24	2	failed	15	35	-1.000	-1.000	[3.00,1.73]	[3.00,1.73]
ellipse	24	2	converged	34	90	-1.000	-1.000	[3.00,1.73]	[3.00,1.73]
ellipse everywhere	24	2	converged	41	104	-1.000	-1.000	[3.00,1.73]	[3.00,1.73]
ellipse segment 1	24	2	converged	34	91	-1.000	-1.000	[3.00,1.73]	[3.00,1.73]
ellipse segment 2	24	2	converged	35	95	-1.000	-1.000	[3.00,1.73]	[3.00,1.73]
polyhedral	24	2	converged	15	47	-1.000	-1.000	[3.00,1.73]	[3.00,1.73]
circumscribed ellipse	250	3	failed	24	103	-3300.000	-3300.000	[20.00,11.00,15.00]	[20.00,11.00,15.00]
ellipse	250	3	converged	53	207	-3300.000	-3300.000	[20.00,11.00,15.00]	[20.00,11.00,15.00]
ellipse everywhere	250	3	failed	48	103	-3298.196	-3300.000	[19.99,10.99,15.02]	[20.00,11.00,15.00]
ellipse segment 1	250	3	failed	53	112	-3299.243	-3300.000	[19.99,11.00,15.01]	[20.00,11.00,15.00]
ellipse segment 2	250	3	failed	54	112	-3299.082	-3300.000	[19.99,11.00,15.01]	[20.00,11.00,15.00]
ellipse segment 3	250	3	failed	56	116	-3299.504	-3300.000	[19.99,11.00,15.00]	[20.00,11.00,15.00]
polyhedral	250	3	converged	26	113	-3300.000	-3300.000	[20.00,11.00,15.00]	[20.00,11.00,15.00]
circumscribed ellipse	251	3	failed	20	86	-3456.000	-3456.000	[24.00,12.00,12.00]	[24.00,12.00,12.00]
ellipse	251	3	failed	10	53	-3454.018	-3456.000	[23.55,12.06,12.16]	[24.00,12.00,12.00]
ellipse everywhere	251	3	failed	6	17	-3209.710	-3456.000	[22.82,13.03,10.79]	[24.00,12.00,12.00]
ellipse segment 1	251	3	failed	4	13	-1698.807	-3456.000	[21.71,9.21,8.50]	[24.00,12.00,12.00]
ellipse segment 2	251	3	failed	6	17	-3210.341	-3456.000	[21.49,11.46,13.04]	[24.00,12.00,12.00]
ellipse segment 3	251	3	failed	14	64	-3449.803	-3456.000	[22.84,12.29,12.29]	[24.00,12.00,12.00]
polyhedral	251	3	converged	22	124	-3456.000	-3456.000	[24.00,12.00,12.00]	[24.00,12.00,12.00]
circumscribed ellipse	25	3	converged	760	1472	0.000	0.000	[52.92,24.90,1.52]	[50.00,25.00,1.50]
ellipse everywhere	25	3	failed	1	1	32.835	0.000	[100.00,12.50,3.00]	[50.00,25.00,1.50]
ellipse segment 1	25	3	failed	1	1	32.835	0.000	[100.00,12.50,3.00]	[50.00,25.00,1.50]
ellipse segment 2	25	3	failed	1	1	32.835	0.000	[100.00,12.50,3.00]	[50.00,25.00,1.50]
ellipse segment 3	25	3	failed	1	1	32.835	0.000	[100.00,12.50,3.00]	[50.00,25.00,1.50]
polyhedral	25	3	converged	1793	3396	0.000	0.000	[50.94,24.97,1.51]	[50.00,25.00,1.50]
circumscribed ellipse	35	3	failed	781	1028	0.111	0.111	[1.33,0.78,0.44]	[1.33,0.78,0.44]
ellipse	35	3	converged	28	122	0.111	0.111	[1.33,0.78,0.44]	[1.33,0.78,0.44]
ellipse everywhere	35	3	failed	24	82	0.113	0.111	[1.36,0.79,0.42]	[1.33,0.78,0.44]
ellipse segment 1	35	3	failed	16	76	0.112	0.111	[1.31,0.79,0.45]	[1.33,0.78,0.44]
ellipse segment 2	35	3	failed	16	75	0.112	0.111	[1.31,0.79,0.45]	[1.33,0.78,0.44]
ellipse segment 3	35	3	failed	16	77	0.112	0.111	[1.31,0.79,0.45]	[1.33,0.78,0.44]
polyhedral	35	3	converged	14	112	0.111	0.111	[1.33,0.78,0.44]	[1.33,0.78,0.44]
circumscribed ellipse	36	3	failed	19	81	-3300.000	-3300.000	[20.00,11.00,15.00]	[20.00,11.00,15.00]
ellipse	36	3	converged	53	202	-3300.000	-3300.000	[20.00,11.00,15.00]	[20.00,11.00,15.00]
ellipse everywhere	36	3	failed	63	135	-3299.808	-3300.000	[20.00,11.00,15.00]	[20.00,11.00,15.00]
ellipse segment 1	36	3	failed	64	140	-3299.951	-3300.000	[20.00,11.00,15.00]	[20.00,11.00,15.00]
ellipse segment 2	36	3	failed	64	133	-3299.918	-3300.000	[20.00,11.00,15.00]	[20.00,11.00,15.00]
ellipse segment 3	36	3	failed	65	137	-3299.870	-3300.000	[20.00,11.00,15.00]	[20.00,11.00,15.00]
polyhedral	36	3	converged	22	108	-3300.000	-3300.000	[20.00,11.00,15.00]	[20.00,11.00,15.00]
circumscribed ellipse	37	3	failed	27	98	-3456.000	-3456.000	[24.00,12.00,12.00]	[24.00,12.00,12.00]
ellipse	37	3	failed	15	65	-3455.761	-3456.000	[23.94,12.01,12.02]	[24.00,12.00,12.00]
ellipse everywhere	37	3	failed	36	88	-3406.653	-3456.000	[27.34,10.93,11.40]	[24.00,12.00,12.00]
ellipse segment 1	37	3	failed	30	82	-3421.619	-3456.000	[21.29,12.62,12.73]	[24.00,12.00,12.00]
ellipse segment 2	37	3	failed	36	85	-3415.110	-3456.000	[21.05,12.70,12.78]	[24.00,12.00,12.00]
ellipse segment 3	37	3	failed	28	75	-3421.157	-3456.000	[21.31,12.55,12.79]	[24.00,12.00,12.00]
polyhedral	37	3	converged	28	139	-3456.000	-3456.000	[24.00,12.00,12.00]	[24.00,12.00,12.00]
circumscribed ellipse	44	4	failed	17	124	-13.000	-15.000	[3.00,-0.00,4.00,0.00]	[0.00,3.00,0.00,4.00]
ellipse	44	4	converged	50	298	-15.000	-15.000	[0.00,3.00,-0.00,4.00]	[0.00,3.00,0.00,4.00]
ellipse everywhere	44	4	converged	110	416	-15.000	-15.000	[0.00,3.00,0.00,4.00]	[0.00,3.00,0.00,4.00]
ellipse segment 1	44	4	converged	50	262	-15.000	-15.000	[-0.00,3.00,0.00,4.00]	[0.00,3.00,0.00,4.00]
ellipse segment 2	44	4	converged	71	304	-15.000	-15.000	[-0.00,3.00,0.00,4.00]	[0.00,3.00,0.00,4.00]
ellipse segment 3	44	4	failed	7	32	-10.672	-15.000	[-0.00,2.55,0.36,3.41]	[0.00,3.00,0.00,4.00]
ellipse segment 4	44	4	failed	1	1	-1.338	-15.000	[1.44,0.98,1.80,1.80]	[0.00,3.00,0.00,4.00]
polyhedral	44	4	converged	15	134	-13.000	-15.000	[3.00,-0.00,4.00,-0.00]	[0.00,3.00,0.00,4.00]
circumscribed ellipse	45	5	failed	21	190	1.000	1.000	[1.00,2.00,3.00,4.00,5.00]	[1.00,2.00,3.00,4.00,5.00]
ellipse	45	5	converged	51	405	1.000	1.000	[1.00,2.00,3.00,4.00,5.00]	[1.00,2.00,3.00,4.00,5.00]
ellipse everywhere	45	5	converged	414	1121	1.000	1.000	[1.00,2.00,3.00,4.00,5.00]	[1.00,2.00,3.00,4.00,5.00]
ellipse segment 1	45	5	converged	63	374	1.000	1.000	[1.00,2.00,3.00,4.00,5.00]	[1.00,2.00,3.00,4.00,5.00]
ellipse segment 2	45	5	converged	82	404	1.000	1.000	[1.00,2.00,3.00,4.00,5.00]	[1.00,2.00,3.00,4.00,5.00]
ellipse segment 3	45	5	converged	104	453	1.000	1.000	[1.00,2.00,3.00,4.00,5.00]	[1.00,2.00,3.00,4.00,5.00]
ellipse segment 4	45	5	converged	130	497	1.000	1.000	[1.00,2.00,3.00,4.00,5.00]	[1.00,2.00,3.00,4.00,5.00]
ellipse segment 5	45	5	converged	158	478	1.000	1.000	[1.00,2.00,3.00,4.00,5.00]	[1.00,2.00,3.00,4.00,5.00]
polyhedral	45	5	converged	22	269	1.000	1.000	[1.00,2.00,3.00,4.00,5.00]	[1.00,2.00,3.00,4.00,5.00]
circumscribed ellipse	76	4	failed	15	111	-4.682	-4.682	[0.27,2.09,-0.00,0.55]	[0.27,2.09,-0.00,0.55]
ellipse	76	4	failed	17	137	-4.682	-4.682	[0.27,2.09,0.00,0.54]	[0.27,2.09,-0.00,0.55]
ellipse everywhere	76	4	failed	25	106	-4.604	-4.682	[0.45,1.89,-0.00,0.77]	[0.27,2.09,-0.00,0.55]
ellipse segment 1	76	4	failed	17	111	-4.682	-4.682	[0.27,2.09,-0.00,0.54]	[0.27,2.09,-0.00,0.55]
ellipse segment 2	76	4	failed	24	105	-4.680	-4.682	[0.31,2.08,0.00,0.52]	[0.27,2.09,-0.00,0.55]
ellipse segment 3	76	4	failed	25	106	-4.677	-4.682	[0.34,2.07,0.00,0.51]	[0.27,2.09,-0.00,0.55]
ellipse segment 4	76	4	failed	23	107	-4.674	-4.682	[0.34,2.03,-0.00,0.60]	[0.27,2.09,-0.00,0.55]
polyhedral	76	4	converged	15	169	-4.682	-4.682	[0.27,2.09,-0.00,0.55]	[0.27,2.09,-0.00,0.55]

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:

$$\rho(s,\alpha) = \frac{1}{\|\mathcal{P}\|}\|p \in \mathcal{P} | r(s,p) \leq \alpha\|. \tag{108}$$

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 $alpha$ 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 Hot-Schittkowski problem set is give in figure Figure 14.

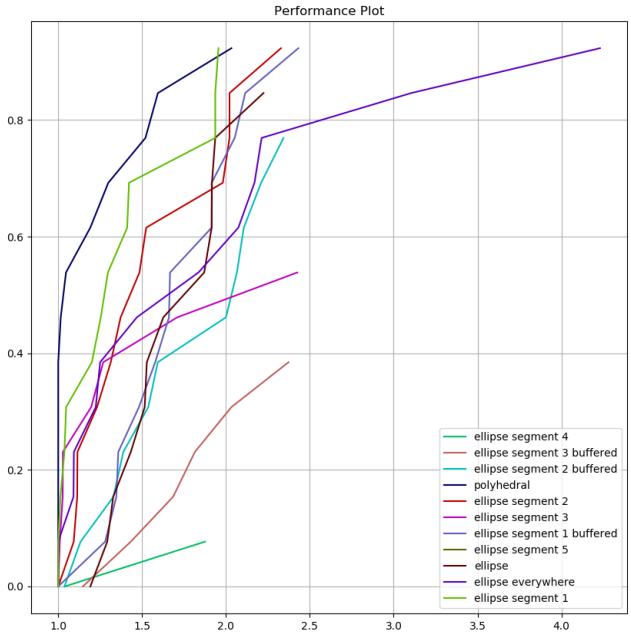

Figure 14: Performance profile

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 ran on these. Notice that the polyhedral search does very well. We conjecture that this may not hold with modelled, nonlinear constraints.

10.22 Summary

We still experienced a problem with iterates coming too close to 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 $Ax \leq b\upsilon + (1-\upsilon)Ax^{(k)}$. Before doing this, the rows of A are normalized. This produces the buffered segment searches withing the results.

10.23 Figure out where goes

From here on, we will assume that the iterates $x^{(k)}$ are chosen according to Algorithm 7. This implies that each of the sample points used to construct ${m_f}^{(k)}$ are output of Algorithm 6.

Because Assumption 10.1 and Assumption 10.2 are satisfied, f also satisfies **??** and hence the assumptions for Theorem 10.1. Notice that because κ_f, κ_g, κ_h only depend on p, L_h, and Λ, these values do not depend on the iteration k: using the same tolerance ξ_{min} within Algorithm 6 implies a bound on Λ. , and therefore ${m_f}^{(k)}$ satisfies the requirements for Theorem 10.1. is a fully quadratic model over $B_\infty(x^{(k)}, \Delta_k)$.

.1 Table of Notation

$x^{(k)}$	is the current iterate in iteration k
${m_f}^{(k)}$	is the model of the objective f during iteration k
$m_{c_i}^{(k)}$	is the model of the i-th constraint c_i during iteration k
${m_c}^{(k)}$	is the model of the constraint c during iteration k
Δ_k	is the outer trust region radius in iteration k
This	is some filler.....
Ω	is the domain
$\mathbb{R}$	is the real numbers
f	is the objective function
f_{min}	is a lower bound on the objective function
β	is a bound on the hessian of the model functions
c_i	are the constraints $\forall i \in \mathcal{I} \cup \mathcal{E}$
e_i	is the unit vector
ϕ_i	is a basis vector
Y	is the set of sample points
V	is a vander mode matrix
y^i	is a sample point
d	is the dimension of the space of model functions
λ_i	are the weights of a linear combination
α_i	are the coefficients of a model function on its basis polynomials
ϕ_i	are basis polynomials
$p-1$	is the size of the sample set
l_i	is a lagrange polynomial
Λ	is a constant bounding the poisedness of a sample set
$\kappa_{ef}, \kappa_{eg}, \kappa_{eh}$	are constants used to bound the model's error

κ_f	is a constant in the efficiency condition
κ_g	is a constant in the accuracy condition
$s^{(k)}$	is the trial point in iteration k
$\chi_m^{(k)}$	is the criticality measure in iteration k
$T_{\text{out}}^{(k)}$	is the outer trust region in iteration k
$\mathcal{F}$	is the feasible region
$\mathcal{F}$	is the feasible region during iteration k
D_{out}	is the feasible region intersect the outer trust region
s	is the decision variable within the trust region subproblem
$B_k(c;\Delta)$	is the ball of radius Δ centered at point c in the k norm $B_k(c;\Delta) = \{x \in \mathbb{R}^n : \|x - c\|_k \leq \Delta\}$
Δ	is the trust region radius
ρ	measures actual improvement over the predicted improvement
γ_1, γ_2	are bounds on ρ
$\omega_{\text{dec}}, \omega_{\text{inc}}$	are scalars used to manipulate the trust region radius
τ	is a tolerance
$\mathcal{F}^k$	is the feasible region
f^k	is the function value
g^k	is the gradient of the model
A	is the jacobian of the constraint model functions
ξ_{min}	is a tolerance within the LU pivoting algorithm
T	is an affine transformation that brings the trust region back to the origin
Q	is the semi-definite matrix defining the affine transformation T
L	is a cholesky factorization of Q
L	is the Lipschitz constant of f
μ^k	is the center of the ellipse
$E^{(k)}$ is the ellipse during iteration k	
π	is a scaling factor while finding the ellipse
d	are the differences between the center of the ellipse and where the ellipse intersect the constraints?
M	is an upper bound
P	is a polyhedron
$\delta_{i,j}$	is the kronecker delta function, $\delta_{i,i} = 1$, $\delta_{i,j} = 0$ if $i \neq j$
τ_ξ	is a threshold for the criticallity measure
τ_Δ	is a threshold for the trust region radius

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.

[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.

[80] G. Fasano, J. L. Morales, and J. Nocedal, "On the geometry phase in model-based algorithms for derivative-free optimization," *Optimization Methods and Software*, vol. 24, no. 1, pp. 145–154, 2009.