Derivative Free Model-Based Methods for Optimization with 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 [**?**], circuitry arrangements [1], machine learning [2], volume optimization [3], and reliability based optimization [4]. In such applications, function evaluations are expensive, so it is sensible to invest significant computational resources to minimize the number of function evaluations.

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

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

where f and $c_i, 1 \leq i \leq m$ 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 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

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

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 this problem and the original is 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, or at least near, the trust-region.

We are specifically interested in applications where some of the black box functions cannot be evaluated outside the feasible region. We therefore impose the restriction on our algorithm that all sample points used to construct the model functions must be feasible.

An important consideration in fitting model functions is the "geometry" of the sample set. This will be discussed in more detail in Section 2.2, but the key point is that the relative positions of the sample points 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 to improve the geometry of the sample set. It is well understood how to do this for unconstrained problems (see for example [**?**]); but for constrained problems, our requirement that the sample points must be feasible poses some interesting challenges with respect to maintaining good geometry. As a first step toward developing an algorithm to solve such problems, *we consider a simplified problem where all of the constraints are linear*; namely:

$$\begin{array}{ll} \min_{x\in\mathbb{R}^n} & f(x) \\ & Gx \leq g, \end{array} \tag{1}$$

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

The central idea to our method 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 [**?**] 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. First, in section ??, we 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 [**?**], which describes a class of trust-region algorithms for convex constrained minimization without derivatives.

The paper is organized as follows. Section Section 2 reviews existing methods for constructing interpolation models, selecting sample points, and blah, blah, blah...

Notation Any function or variable that depends 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 is given by ${m_f}^{(k)}$. The i-th row of the matrix A is denoted A_i, while the i-th column is denoted $A_{\bullet i}$. Subscripts on vectors are used as an index into the vector, while vectors in a sequence of vectors use superscripts. Matrices are denoted with capital letters, and we use e_i to denote the i-th unit vector.

$B_k(c;\Delta)$ is the ball of radius Δ in the k norm, centered at point c . $\delta_{i,j}$ is the kronecker delta, $\delta_{i,i} = 1$, $\delta_{i,j} = 0$ if $i \neq j$.

2 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.

2.1 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*

$$m_f(y^i) = \sum_{j=0}^{p} \alpha_j \phi_j(y^i) = f(y^i) \quad \forall \quad 0 \leq i \leq p. \tag{2}$$

This equation can be written more compactly in the form

$$V\alpha = \bar{f}, \tag{3}$$

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

$$V = M(\Phi, Y) := \begin{bmatrix} \phi_0(y^0) & \phi_1(y^0) & \ldots & \phi_p(y^0) \\ \phi_0(y^1) & \phi_1(y^1) & \ldots & \phi_p(y^1) \\ & & \vdots & \\ \phi_0(y^p) & \phi_1(y^p) & \ldots & \phi_p(y^p) \end{bmatrix}, \tag{4}$$

The interpolation equation (3) 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.

¡

2.2 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

$$l_i(y^j) = \delta_{i,j} \tag{5}$$

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

$$m_f(x) = \sum_{j=0}^{p} f(y^i) l_i(x). \tag{6}$$

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

$$\Lambda \geq \max_{0 \leq i \leq p} \max_{x \in B} |l_i(x)|. \tag{7}$$

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 2.1. *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 $B(y^0; \Delta)$:*

$$\|\nabla^2 f(y) - \nabla^2 m_f(y)\| \leq \kappa_h \Delta \quad \forall y \in B_2(y^0; \Delta) \tag{8}$$

$$\|\nabla f(y) - \nabla m_f(y)\| \leq \kappa_g \Delta^2 \quad \forall y \in B_2(y^0; \Delta) \tag{9}$$

$$|f(y) - m_f(y)| \leq \kappa_f \Delta^3 \quad \forall y \in B_2(y^0; \Delta). \tag{10}$$

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 2.2. *([**?**, 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

$$\|\nabla {m_f}^{(k)}(x^{(k)}) - \nabla f(x^{(k)})\| \leq \kappa_g \Delta_k \tag{11}$$

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 [5], but a step to ensure good geometry is required for convergence analysis although it may come at the expense of adding more function evaluations.

2.3 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 1.

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 1 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 $\xi_{\min} > 0$.

Step 1 (Pivot)

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

Step 2 (Check threshold)

If $|V_{i,i}| < \xi_{\min}$ 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 2.3. *The sample set Y obtained from Algorithm 1 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 3.1 that this algorithm can be used to create a poised set over an ellipsoidal region.

3 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

$$T_{\text{out}}^{(k)} = B_\infty(x^{(k)}, \Delta_k) = \{x \in \mathbb{R}^n |\ {x^{(k)}}_i - \Delta_k \leq x_i \leq {x^{(k)}}_i + \Delta_k \quad \forall 1 \leq i \leq n\}. \tag{12}$$

Defining $A = \begin{bmatrix} G \\ I \\ -I \end{bmatrix}$ and $b = \begin{bmatrix} g \\ \xi^{(k)} + \Delta_k \\ -\xi^{(k)} + \Delta_k \end{bmatrix}$, 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

3.1 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 3.1. *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*

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

Then, with

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

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

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

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 $Q^{(k)}$.

3.2 Finding the maximum volume feasible ellipsoid for a fixed center

The error bounds given in Theorem 3.1 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 [6], 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:

$$Q = V(\mu) = \sup_{Q \succeq 0} \det(Q^{-1}) \tag{13}$$
$$s.t. \quad A_i^T Q^{-1} A_i \leq \frac{1}{2}\bar{b_i}^2.$$

Theorem 3.2. *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 (13), 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,

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

Because the volume of the ellipsoid is proportional to the determinant of Q^{-1}, the maximal ellipsoid is defined by (13). □

3.3 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.

3.3.1 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 1. 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.

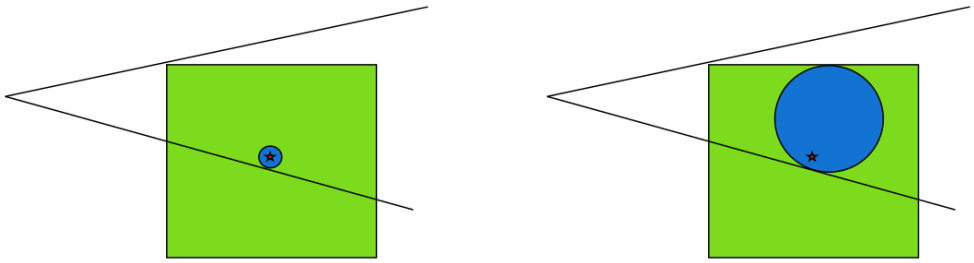

Figure 1: When the current iterate is too close to a constraint, the circular trust region becomes too small. Shifting the trust region center helps remedy this. The star is the current iterate, the green is the outer trust region, and blue the inner.

3.3.2 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 Figure 2, we have this type of iterate, but by using an ellipsoid we are still able to search towards the vertex of the feasible region.

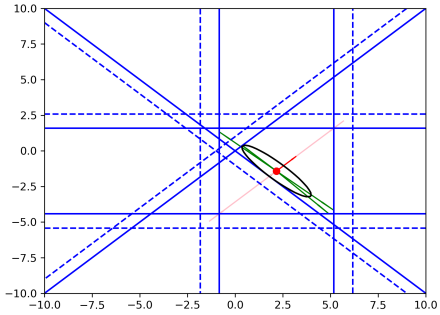

Figure 2: A nicer trust region

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 $\mu^k = x^k$.

3.3.3 Search Everything

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

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

where $V(\mu)$ is the volume of the ellipsoid defined in (13). 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 3, although the ellipsoid found has larger volume than before being shifted, this ellipsoid contains points farther from the corner containing the minimizer.

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

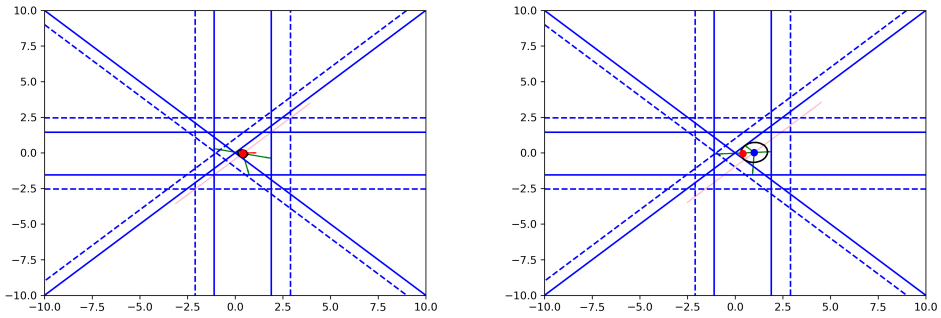

Figure 3: Searching $\mathcal{F}$

3.3.4 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 4, as we should avoid letting the new center get closer to another constraint:

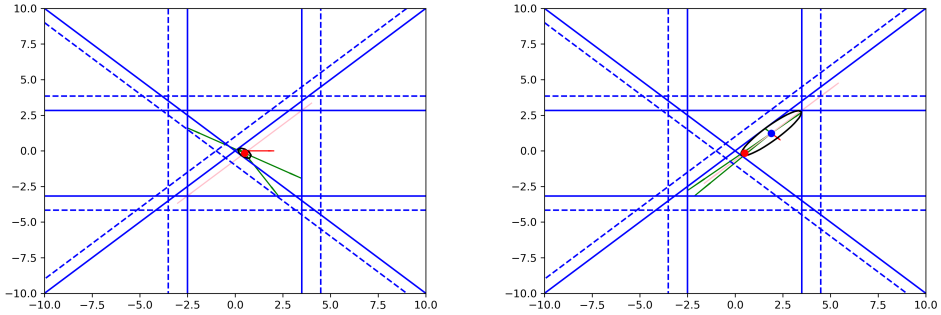

Figure 4: Line searches

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 *ConstructrustRegion* 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

$$t = \arg\min_j \frac{d(A_{E0}, s_i) - d(A_{Rj}, s_i)}{A_{Rj} - d(A_{E0})p} | (A_{Rj} - d(A_{E0})p > 0. \tag{14}$$

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 5, 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.

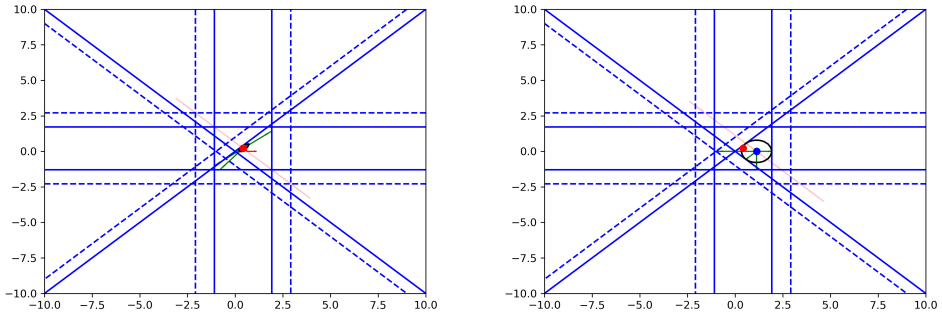

Figure 5: Ellipse runs away from the optimizer

==========

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

3.3.5 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: $T^{(k)}_{\text{search}} = E^{(k)} \cap \mathcal{X}$.

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.

3.4 Algorithm Components

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

3.4.1 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

$$\chi^{(k)} = \|x^{(k)} - \text{Proj}_{\mathcal{F}}(x^{(k)} - \nabla {m_f}^{(k)}(x^{(k)}))\| \tag{15}$$

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.

3.4.2 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

$$\rho_k = \frac{f(x^{(k)}) - f(x^{(k)} + s^{(k)})}{{m_f}^{(k)}(x^{(k)}) - {m_f}^{(k)}(x^{(k)} + s^{(k)})} \tag{16}$$

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 [7] and within a derivative free context [**?**]. The user supplies fixed constants $0 < \gamma_{\min} \leq \gamma_{\text{sufficient}} \leq 1$ 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.

3.5 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

$$T_{\text{out}}^{(k)} = B_\infty(x^{(k)}, \Delta_k) = \{x \in \mathbb{R}^n |\ x_i^{(k)} - \Delta_k \leq x_i \leq x_i^{(k)} + \Delta_k \quad \forall 1 \leq i \leq n\}. \tag{17}$$

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 $T_{\text{interp}}^{(k)}$ satisfying $T_{\text{interp}}^{(k)} \subset T_{\text{out}}^{(k)} \cap \mathcal{F}$ and $x^{(k)} \in T_{\text{interp}}^{(k)}$. 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 $T_{\text{interp}}^{(k)}$ and $T_{\text{search}}^{(k)}$.

1. Take

$$T_{\text{interp}}^{(k)} = T_{\text{search}}^{(k)} = T_{\text{out}}^{(k)} \cap \mathcal{F}. \tag{18}$$

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

2. Force

$$T_{\text{interp}}^{(k)} \subseteq T_{\text{search}}^{(k)} \subseteq T_{\text{out}}^{(k)} \cap \mathcal{F} \tag{19}$$

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

The advantage of the ellipsoidal trust region approach (19) is that we can reuse classical methods for ensuring good geometry. We can construct $T_{\text{interp}}^{(k)}$ to be ellipsoidal and use efficient algorithms within [**?**] to satisfy (11). 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

$$T_{\text{search}}^{(k)} = T_{\text{interp}}^{(k)} \tag{20}$$

or

$$T_{\text{search}}^{(k)} = T_{\text{out}}^{(k)} \cap \mathcal{F}. \tag{21}$$

Namely, in (20) 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 (18), 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 $T_{\text{interp}}^{(k)}$ to be a sphere. This saves the computation of $T_{\text{interp}}^{(k)}$ when it is not needed, as there are no nearby constraints.

4 Algorithms

4.0.1 Sufficient Model Reduction

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

$$m_f{}^{(k)}(x^{(k)}) - m_f{}^{(k)}(x^{(k)} + s^{(k)}) \geq \kappa_f \chi_k \min\left\{\frac{\chi_k}{1 + \|\nabla^2 m_f{}^{(k)}(x^{(k)})\|}, \Delta_k, 1\right\} \tag{22}$$

where κ_f is a constant independent of k. This is widely used within trust region frameworks such as [**?**] and [7]. It can be shown that the *generalized Cauchy point* satisfies this condition [7].

4.1 Algorithm Template

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

Algorithm 2 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 $k = 0$, $0 < \omega_{\text{dec}} < 1 \leq \omega_{\text{inc}}$, $0 < \gamma_{\min} < \gamma_{\text{sufficient}} \leq 1$, $\alpha > 0$, $k \leftarrow 1$, $0 < \omega_{\text{dec}} < 1 \leq \omega_{\text{inc}}$, $0 < \gamma_{\min} < \gamma_{\text{sufficient}} < 1$.

Step 1 (Construct the model)
$T^{(k)}_{\text{interp}} \leftarrow$ ConstructTrustRegion$(\Delta_k, x^{(k)})$. Ensure that the sample points are poised with respect to $T^{(k)}_{\text{interp}}$ for (11) by calling Algorithm 1. Construct $m_f{}^{(k)}$ as described in (6) to construct $m_f{}^{(k)}(x)$.

Step 2 (Check stopping criteria)
Compute χ_k as in (15).

If $\chi^{(k)} < \tau_\xi$ and $\Delta_k < \tau_\Delta$ then return $x^{(k)}$ as the solution.
Otherwise, if $\Delta_k > \alpha\chi^{(k)}$ 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 (16)

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

4.2 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 1. 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 1, 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 1 uses a parameter ξ_{min} as a lower bound of the pivot values of the Vandermonde matrix. For unconstrained problems, this approach could always find a pivot value for any $\xi_{\text{min}} \in (0, 1)$ because it 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 6, 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.

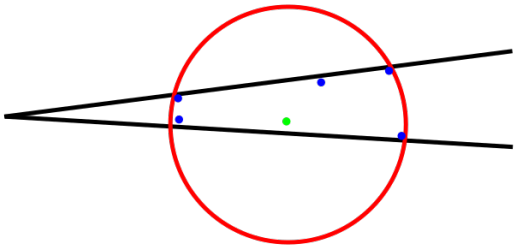

Figure 6: Limited sample point choice

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 3:

Algorithm 3 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 $0 < \xi_{\min} < \xi_{\text{desired}}$, $0 < \delta_{\text{improv}} < 1$, $\xi_{\text{cur}} = \xi_{\text{desired}}$.

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

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

If $|\phi_i(\hat{y})| < \xi_{\min}$ 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 $V_{,j} \leftarrow V_{,j} - V_{i,j} V_{\bullet,i} \forall j = i \dots p$

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

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

4.3 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 (11). Namely, we know from Section 3.1 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 (20), we have no guarantee of satisfying the efficiency condition (22) because Δ_k is the outer trust region radius. However, the model will likely be more accurate over this region.

5 Convergence Discussion

5.1 Algorithm Assumptions

Here, we show convergence for one version of Algorithm 2. Namely, we choose to satisfy (19) and let $T^{(k)}_{\text{interp}}$ have an ellipsoidal shape as described in Section 3.5. Also, we will select (21) as discussed in Section 3.5: namely, $T^{(k)}_{\text{search}} = T^{(k)}_{\text{out}} \cap \mathcal{F}$. To do this, we require the following assumptions.

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

Assumption 1 *The function f is differentiable and its gradient ∇f is Lipchitz continuous with constant $L_g > 0$ in $\mathcal{X}$. That is,*

$$\|\nabla f(x) - \nabla f(y)\| \leq L\|x - y\| \quad \forall x, y \in \mathcal{X}. \tag{23}$$

Assumption 2 *The function f has Lipschitz continuous hessian with constant $L_h > 0$ in $\mathcal{X}$. That is,*

$$\|\nabla^2 f(x) - \nabla^2 f(y)\| \leq L_h \|x - y\| \quad \forall x, y \in \mathcal{X}. \tag{24}$$

Assumption 3 *The function f is bounded below over $\mathcal{X}$. That is,*

$$f(x) \geq f_{min} \quad \forall x \in \mathcal{X}. \tag{25}$$

Assumption 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*

$$\|\nabla^2 {m_f}^{(k)}(x^{(k)})\| \leq \beta - 1 \quad \forall k \geq 0. \tag{26}$$

Assumption 5 *There exists a point $\bar{x}$ within the interior of the feasible region:*

$$G\bar{x} < g. \tag{27}$$

5.2 Required Assumptions

If the tolerances $\tau_\chi = 0$ and $\tau_\Delta = 0$ are set to zero, Algorithm 2 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 (22) is satisfied.

H_1 The function f is differentiable and its gradient ∇f is Lipchitz continuous with constant $L > 0$ in $\mathcal{X}$.

H_2 The function f is bounded below over $\mathcal{X}$.

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 [7] 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 5.3.2. However, first we show a way of using a more straightforward version of H_3 in Section 5.3.1.

5.3 Satisfying these assumptions

5.3.1 More simple H_3

First, we show that Assumption 7 can be used instead of Assumption 4 under some restrictions. Namely, for this to be true, we also first assume:

Assumption 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 4 with Assumption 7:

Assumption 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 5.1. *Assume that Assumption 7, Assumption 2, ??, and Assumption 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 2.1. Then Assumption 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 (8) is satisfied. Combining this with Assumption 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

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

□

5.3.2 Satisfying the Accuracy Assumption

The only remaining assumption is the accuracy condition H_4.

As discussed in Section 3.1, we know that if the condition number of $Q^{(k)}$ is bounded, we can map a poised set over the unit ball to $T^{(k)}_{\text{interp}}$. 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 5.1.1. *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 (28)*
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

$$P_k = \{x \in \mathbb{R}^n | \quad Gx \leq g, \|x - x^{(k)}\|_\infty \leq \Delta_k\} = \{x \in \mathbb{R}^n | A^{(k)} x \leq b^{(k)}, \|{A^{(k)}}_i\| = 1\}. \tag{28}$$

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

$$\mathcal{A}(x) = \{1 \leq i \leq m | c_i(x) = 0 \Leftrightarrow G_i x = g_i\} \tag{29}$$

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

$$\alpha_{\text{minimizers}}(\mathcal{S}) = \begin{cases} \arg\max_{\|u\|=1} \min_{i \in \mathcal{S}} -u^T G_i & \text{if } \mathcal{S} \neq \emptyset \\ \emptyset & \text{if } \mathcal{S} = \emptyset \end{cases} \tag{30}$$

$$\alpha_{\text{minimum}}(\mathcal{S}) = \begin{cases} \max_{\|u\|=1} \min_{i \in \mathcal{S}} -u^T G_i & \text{if } \mathcal{S} \neq \emptyset \\ 1 & \text{if } \mathcal{S} = \emptyset \end{cases} \tag{31}$$

$$\alpha_{\text{func}}(x) = \alpha_{\text{minimum}}\left(\mathcal{A}(x)\right) \tag{32}$$

$$\alpha_k = \alpha_{\text{func}}\left(x^{(k)}\right) \tag{33}$$

$$u^{(k)} \in \alpha_{\text{minimizers}}\left(\mathcal{A}(x^{(k)})\right) \tag{34}$$

Here, α_{func} 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

$$\mathcal{C}(\alpha, d, c) = \left\{x \in \mathbb{R}^n | \quad x = c + td + s, s^T d = 0, t \geq 0, \|s\| \leq \alpha t\right\} \tag{35}$$

$$\mathcal{C}^k_{\text{feasible}} = \mathcal{C}(\alpha_k, u^{(k)}, x^{(k)}) = \{x \in \mathbb{R}^n | \quad x = x^{(k)} + tu^{(k)} + s, s^T u^\star = 0, t \geq 0, \|s\| \leq \alpha_k t\} \tag{36}$$

$$\mathcal{C}^k_{\text{shifted}} = \mathcal{C}(\alpha_k, e_1, 0) = \{x = (t, s)^T \in \mathbb{R}^n, t \in \mathbb{R}_{\geq 0}, s \in \mathbb{R}^{n-1} | \quad \|s\| \leq \alpha_k t\} \tag{37}$$

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

$$R_k = 2\frac{(e_1 + u^{(k)})(e_1 + u^{(k)})^T}{(e_1 + u^{(k)})^T(e_1 + u^{(k)})} - \boldsymbol{I} \tag{38}$$

$$T_k(x) = R_k\left(x - x^{(k)}\right) \tag{39}$$

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

$$f_e(\alpha, \delta, r; x) = (x - \delta e_1)^T \begin{bmatrix} 1 & \mathbf{0}^T \\ \mathbf{0} & \alpha^{-2}\boldsymbol{I} \end{bmatrix} (x - \delta e_1) - r \tag{40}$$

Lemma 5.2. *Assume that Assumption 5 is satisfied and $x_0 \in \mathcal{F}$. Then $1 \geq \alpha_{func}(x_0) > 0$.*

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

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

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

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

Using this along with definitions (30), (31), and (32) we see

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

We know that $\alpha_{\text{func}}(x_0) \leq 1$ 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 5.3. *If Assumption 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 $\alpha_{\text{minimizers}}$, α_{minimum}, and α_k can only take on at most $1 + 2^m$ values. By Lemma 5.2, we know that each of these values must be positive. Thus, we are free to choose ϵ_α to be the smallest of these values. □

Lemma 5.4. *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: $\epsilon \leq \min_{1 \leq i \leq m} b_i^{(k)} - \left(A^{(k)}{}_i\right)^T x_0$. 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 5.5. *The set $\mathcal{C}^k_{feasible}$ defined in (36) 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 $y = x^{(k)} + tu^{(k)} + s \in \mathcal{C}^k_{\text{feasible}}$ and $i \in \mathcal{A}(x^{(k)})$ be arbitrary. Then,

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

□

Lemma 5.6. *Let $\mathcal{C}^k_{shifted}, f_e, \alpha_k$ be defined as in* (37), (40), (33). *Then the ellipsoid*

$$\left\{x \in \mathbb{R}^n | f_e(\alpha_k, \delta, \frac{1}{2}\delta^2; x) \leq 0\right\} \subseteq \mathcal{C}^k_{shifted} \quad \forall \delta > 0 \tag{41}$$

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, $x \in \mathcal{C}^k_{\text{shifted}}$. □

Lemma 5.7. *Let R_k, T_k, $\mathcal{C}^k_{feasible}$, $\mathcal{C}^k_{shifted}$ be defined as in* (38), (39), (36), (37). *Then $T_k(\mathcal{C}^k_{feasible}) = \mathcal{C}^k_{shifted}$.*

Proof. Observe that

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

Suppose that $x \in \mathcal{C}^k_{\text{feasible}}$. Then there exists $t \geq 0$ and $s \in \mathbb{R}^n$ such that $x = x^{(k)} + tu^{(k)} + s$ where $s^T u^{(k)} = 0$ and $\|s\| \leq \alpha_k t$. Then $T_k(x) = tR_k u^{(k)} + R_k s = te_1 + R_k s$. Observe that $(Rs)_1 = (R_k s)^T e_1 = s^T {R_k}^T (R_k u^{(k)}) = s^T u^{(k)} = 0$. Hence, $T_k(x) = \begin{bmatrix} t \\ \sigma \end{bmatrix}$ where $\sigma \in \mathbb{R}^{n-1}$ satisfies $\|\sigma\| = \|s\| \leq \alpha t$. Thus, $T_k(x) \in \mathcal{C}^k_{\text{shifted}}$. Conversely, if $\begin{bmatrix} t \\ \sigma \end{bmatrix} \in \mathcal{C}^k_{\text{shifted}}$, then let $s = {R_k}^T \begin{bmatrix} 0 \\ \sigma \end{bmatrix}$ to see that $x = T_k^{-1}\left(\begin{bmatrix} t \\ \sigma \end{bmatrix}\right) = {R_k}^T \left(te_1 + \begin{bmatrix} 0 \\ \sigma \end{bmatrix}\right) = tu^{(k)} + s$ where $\|s\| = \|\sigma\| \leq \alpha t$. Hence $T_k^{-1}\left(\begin{bmatrix} t \\ \sigma \end{bmatrix}\right) \in \mathcal{C}^k_{\text{feasible}}$. □

Lemma 5.8. *Let R_k, T_k, $\mathcal{C}^k_{feasible}$, $\mathcal{C}^k_{shifted}$, P_k be defined as in* (38), (39), (36), (37), (28).
For each iteration k, there exists a $\delta_f > 0$ such that the ellipsoid

$$\mathcal{E}^k_{feasible} = \left\{x \in \mathbb{R}^n | f_e(\delta_f, \frac{1}{2}{\delta_f}^2, \alpha_k, T_k(x)) \leq 0\right\} \tag{42}$$

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

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 $\delta_{\text{f}} = \frac{1}{\alpha'}L$. We see that if $x \in \mathcal{E}^k_{\text{feasible}}$, then by Lemma 5.6 we have that $T_k(x) = \begin{bmatrix} t \\ \sigma \end{bmatrix} \in \mathcal{C}^k_{\text{shifted}}$ for some $\sigma \in \mathbb{R}^{n-1}$, and

$$(x - \delta_{\text{f}} e_1)^T \begin{bmatrix} 1 & \mathbf{0}^T \\ \mathbf{0} & \alpha_k^{-2}\boldsymbol{I} \end{bmatrix} (x - \delta_{\text{f}} e_1) \leq \frac{1}{2}{\delta_{\text{f}}}^2$$
$$\Longrightarrow (t - \delta_{\text{f}})^2 + \frac{1}{\alpha_k^2}\|\sigma\|^2 \leq \frac{1}{2}{\delta_{\text{f}}}^2 \Longrightarrow (t-\delta_{\text{f}})^2 \leq \frac{1}{2}{\delta_{\text{f}}}^2 \Longrightarrow t \leq \left(1 + \frac{1}{\sqrt{2}}\right)\delta_{\text{f}}$$

so that

$$\|x\|^2 = t^2 + \|\sigma\|^2 \leq \left(1 + \alpha_k^2\right)t^2 \leq \left(1 + \alpha_k^2\right)\left(1 + \frac{1}{\sqrt{2}}\right){\delta_{\text{f}}}^2 = {\alpha'}^2{\delta_{\text{f}}}^2$$
$$\Longrightarrow \|x\| \leq \alpha'\delta_{\text{f}} \leq L$$

Thus, all points within $\mathcal{E}^k_{\text{feasible}}$ are closer than the nearest point of a non-active constraint. Combine this with Lemma 5.5 to see that $\mathcal{E}^k_{\text{feasible}} \subseteq P_k$. □

Lemma 5.9. *For some iteration k, let R_k, T_k, $\mathcal{C}^k_{feasible}$, $\mathcal{C}^k_{shifted}$, P_k be defined as in* (38), (39), (36), (37), (28). *Also, let $\delta_f > 0$ be defined as in Lemma 5.8.*

$$\hat{\mathcal{E}}^k_{feasible} = \left\{x \in \mathbb{R}^n | f_e(\delta_f, {\delta_f}^2, \alpha_k, T_k(x)) \leq 0\right\} \tag{43}$$

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

Proof. We have that

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

□

Lemma 5.10. *For iteration k, if $\alpha_k > 0$, then there exists a suitable ellipsoid for iteration k.*

Proof. Let R_k, be defined as in (38) . Also, let δ_{f}, $\hat{\mathcal{E}}^k_{\mathrm{feasible}}$ be defined as in (42) (43) .

Let

$$\begin{aligned} Q^{(k)} &= {R_k}^T \begin{bmatrix} 1 & \mathbf{0}^T \\ \mathbf{0} & \alpha_k^{-2}\boldsymbol{I} \end{bmatrix} R_k \\ c_k &= x^{(k)} - \delta_{\mathrm{f}} u^{(k)} \\ \epsilon &= \delta^2 \end{aligned}$$

By Lemma 5.8, we know that $\mathcal{E}^k_{\mathrm{feasible}} \subseteq P_k$. By Lemma 5.9, we know that $x^{(k)} \in \hat{\mathcal{E}}^k_{\mathrm{feasible}}$. The condition number of $\sigma(Q^{(k)}) = \frac{\max\{1, \alpha_k^{-2}\}}{\min\{1, \alpha_k^{-2}\}} = \alpha_k^{-2}$. This is because $\det(R_k) = 1$ means the condition number of $Q^{(k)}$ is not affected R_k. □

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

Assumption 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 5.11. *Let Y be a poised set of p points, and let $R = \max_i \|y^i - y^0\|$. Let f satisfy Assumption 8 over some convex set Ω, and let $m(x)$ denote the quadratic model of f using* (6). *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 5.12. *Suppose that Assumption 5, Assumption 8 hold. The accuracy condition* (11) *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 Lemma 5.4 to construct a suitable ellipsoid. Otherwise, we can use Lemma 5.3 to satisify the conditions of Lemma 5.10. This provides two ellipsoids:

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

As in Theorem 3.1, 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 $T(x) = \delta DL^T(x - \mu^{(k)})$ maps $E^{(k)}$ to the δ ball. As also described in Theorem 3.1, we create shifted functions $\hat{m}_f(u) = m_f(T^{-1}(u))$ and $\hat{f}(u) = f(T^{-1}(u))$. After using Algorithm 1 to choose sample points, we know by Theorem 2.1 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 Lemma 5.11 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 $\Delta_{\max} > 0$, such that $\Delta_k \leq \Delta_{\max}$ $\forall k \in \mathbb{N}$. After using Lemma 5.3 to construct an $\epsilon_\alpha > 0$ and defining $\kappa''_g = \frac{\kappa'_g}{\epsilon_\alpha} \Delta_{\max}$, we see that we can use Theorem 3.1 to conclude that for all $x_0 \in \hat{\mathcal{E}}^k_{\text{feasible}}$:

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

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

Notice that although Theorem 5.12 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.

6 Results

6.1 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 $x_2 \leq a x_1$, $x_2 \geq -a x_1$ 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.

6.2 Schittkowksi Test Problems for Nonlinear Optimization

We tested these algorithms on several problems from the Hot-Schittkowski problem set [**?**], [8]. 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 [9]. 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:

$$\rho(s, \alpha) = \frac{1}{\|\mathcal{P}\|} \|p \in \mathcal{P} | r(s, p) \leq \alpha\|. \tag{44}$$

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 7.

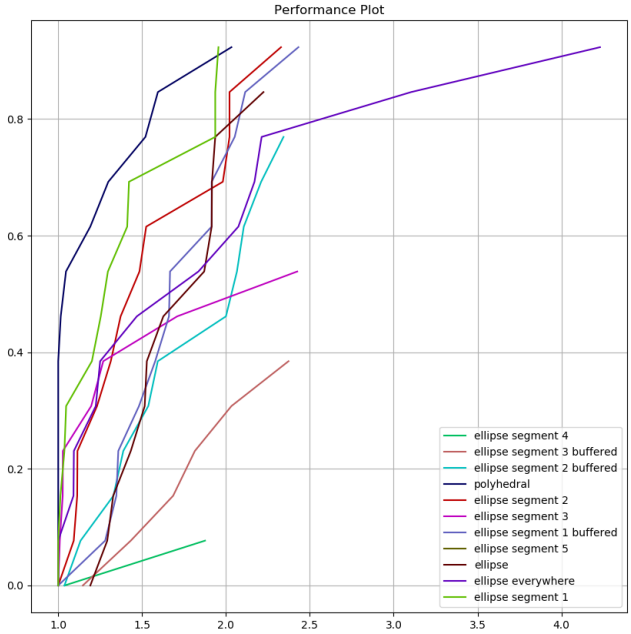

Figure 7: Performance profile

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.

6.3 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.

7 Figure out where goes

From here on, we will assume that the iterates $x^{(k)}$ are chosen according to Algorithm 2. This implies that each of the sample points used to construct ${m_f}^{(k)}$ are output of Algorithm 1.

Because Assumption 1 and Assumption 2 are satisfied, f also satisfies **??** and hence the assumptions for Theorem 2.1. 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 $\xi_{\min}$ within Algorithm 1 implies a bound on Λ. , and therefore ${m_f}^{(k)}$ satisfies the requirements for Theorem 2.1. is a fully quadratic model over $B_\infty(x^{(k)}, \Delta_k)$.

A 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.....
$\mathcal{X}$	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
$\chi^{(k)}$	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

$\xi_{\min}$	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. 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.

[2] 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.

[3] 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.

[4] 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.

[5] 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.

[6] 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.

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

[8] 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.

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