Derivative Free Model-Based Methods for Local Constrained Optimization

Trever Hallock

Comments from Steve Billups, June 29, 2018

Contents

Table 1: Table of Notation

f	is the objective function
c_i	are the constraints $\forall i \in \mathcal{I} \cup \mathcal{E}$
$\mathcal{I}$	is the index set of the inequality constraints
$\mathcal{E}$	is the index set of the equality constraints
e_i	is the unit vector
ϕ_i	is a basis vector
m	is a model function
y^i	is a sample point
d	is the dimension of the space of model functions
λ_i	are the weights of the linear combination
$x^{(k)}$	is the current iterate
s	is the decision variable within the trust region subproblem
$B(x^{(k)}; \Delta)$	is the ball of radius Δ centered at point x
Δ	is the trust region radius
ρ	measures actual improvement over the predicted improvement
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	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
μ^k	is the center of the ellipse
π	is a scaling factor
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

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. 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 aimed at developing algorithms to solve constrained optimization problems of the form

$$\begin{array}{ll} \min_{x \in X} & f(x) \\ \text{subject to} & c_i(x) \leq 0 \quad i \in \mathcal{I} \\ & c_i(x) = 0 \quad i \in \mathcal{E}, \end{array}$$

where X is a subset of $\mathbb{R}^n$, and f and $c_i, i \in \mathcal{I} \cup \mathcal{E}$ are real-valued functions on X with at least one of these functions being a *black-box* function, meaning that derivatives cannot be evaluated directly.

In our work, we assume that all of the functions in the problem are black-box functions. We consider two variations of this problem. In the first variation, we assume that the constraints are fully quantifiable. This means that all of the functions can be evaluated at any point in X and that the values returned for the constraint functions provide meaningful information about how close the point is to a constraint boundary. In the second variation, we assume that the black-box functions return meaningful numerical values only when evaluated at feasible points. In this case, the constraints are called *partially quantifiable*.

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

$$\begin{array}{ll} \min_{s|\|s\|\leq\Delta} & m_f(x^k+s) \\ \text{subject to} & m_{c_i}(x^k+s)\leq 0 \quad i\in\mathcal{I} \\ & m_{c_i}(x^k)=0 \quad i\in\mathcal{E}, \end{array}$$

where $m_f, m_{c_i}, i \in \mathcal{I}\cup\mathcal{E}$ are the model functions, and Δ is the radius of the trust-region. Conceptually, the model functions are "trusted" only within a distance Δ of the current iterate x^k; so the trust-region subproblem restricts the step s to be no larger than Δ. 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.

An important consideration in fitting the model functions is the the "geometry" of the sample set. This will be discussed in more detail in Section (2.2.1), but the key point is that the relative positions of the sample points within the trust region has 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.

In the case of partially quantifiable constraints, all of the sample points must lie within the feasible region. This poses some interesting challenges for geometry of the sample set. As a first step toward developing an algorithm to solve such problems, we consider a simplified problem where all of the constraints are linear, but we impose the restriction that all sample points must lie within the feasible region. This work is presented in Chapter (3)

In the case of fully quantifiable constraints, there is no difficulty in using sample points outside of the feasible region. In Chapter (**??**), we describe a Trust-Region SQP Filter method for solving this class of problems.

2 Background

2.1 Strategy

A reasonable approach to DFO is to modify classical algorithms that rely on derivatives by using the derivatives of the model functions whenever a derivative is needed in the algorithm. However, the lack of explicit derivatives pose several challenges, which necessitate changes in the classical algorithms. First, care must be taken to ensure that the model functions are sufficiently accurate approximations to the true functions. This means that geometry of the sample points must be considered. It also means that the trust-region radius must get arbitrarily small.

The transition to the derivative-free setting also provides some interesting research opportunities. In particular, because function evaluations are so costly, there is potential for significant gains by considering more complex optimization paradigms. For example, Sequential Quadratic Programming (SQP) methods use subproblems involving quadratic approximations of the objective function f and linear approximations of the constraint functions. In the derivative-free setting, it may be worthwhile to explore other variations of the classical algorithms. For example, it might be worthwhile to construct quadratic models of the constraint functions, which might result in fewer iterations overall.

It might also be worthwhile to consider using different sample sets for fitting different functions.

For example, the SQP trust-region filter method involves constructing a quadratic model of the objective function and linear models of the constraint functions. But this raises an interesting question, since a good sample set for constructing a quadratic model may not be ideal for fitting the linear functions modeling the constraints.

2.1.1 Model-based, Trust Region Methods

We will be modifying the following classical algorithm described at a high level here. A set of poised points are chosen for some radius $\Delta > 0$ about the current iterate. The objective and constraints are then evaluated at these points to construct a model function as a linear combination of some set of basis functions. Next, the model is minimized over this trust region and the argument minimum becomes the trial point. The objective is evaluated at the trial point and a measure of reduction ρ is computed. If ρ implies that sufficient reduction has been made and that the model approximates the function well, the trial point is accepted as the new iterate. Otherwise, the trust region is reduced to increase model accuracy.

For unconstrained optimization, the algorithmic framework can be described with these steps:

1. Create a model function $m_k(x)$.
   - In classical trust region methods, the following quadratic model function is used:
   $$m_k(x) = f(x^{(k)}) + \nabla f(x^{(k)})^T(x-x^{(k)}) + \frac{1}{2}(x-x^{(k)})^T\nabla^2 f(x^{(k)})(x-x^{(k)})$$
   - $\nabla f(x^{(k)})$ and $\nabla^2 f(x^{(k)})$ must be approximated
   - Geometric properties of the sample set that must be satisfied
2. If $\nabla m_k(x) < \tau$ stop, where τ is some tolerance

3. Solve the Trust region subproblem: $s^{(k)} = \arg\min_{s \in B(x^{(k)};\Delta_k)} m_k(x^{(k)} + s)$
   - $B(x^{(k)}; \Delta_k) = \{x \in \mathbb{R}^n : \|x - x^{(k)} \leq \Delta_k\}$ is the ball of radius Δ_k centered at $x^{(k)}$
4. Test for improvement
   - Compute
   $$\rho_k = \frac{f(x^{(k)}) - f(x^{(k)} + s^{(k)})}{m_k(x^{(k)}) - m_k(x^{(k)} + s^{(k)})} \tag{1}$$
   which measures the actual improvement over predicted improvement
   - If ρ is small, $x^{(k+1)} = x^{(k)}$ (reject) and decrease radius
   - If ρ is intermediate, $x^{(k+1)} = x^{(k)} + s^{(k)}$ (accept) and decrease radius
   - If ρ is large, $x^{(k+1)} = x^{(k)} + s^{(k)}$ (accept) and either increase the radius or decrease if $\nabla m_k(x_k)$ is small
5. Go to step 1

Our goal is generalize this framework to handle constraints, where we must reduce constraint violation while ensuring the accuracy of the models of the constraints.

2.1.2 Interpolation/regression methods

Within interpolation model-based methods, we construct our model by regressing basis functions onto a set of sampled points. For example, given a function $f(x) : \mathbb{R}^n \to \mathbb{R}$ we can use a set of basis functions $\phi_i : \mathbb{R}^n \to \mathbb{R} \quad \forall 1 \leq i \leq d_1$ to construct a model function $m(x) = \sum_{i=1}^{d_1} \lambda_i \phi_i(x)$ approximating $f(x)$ by selecting appropriate $\lambda_i \in \mathbb{R}$. This is done by choosing a set of sample points $Y = \{y^1, y^2, \ldots, y^{d_2}\}$, evaluating $f = (f_1 = f(y^1), f_2 = f(y^2), \ldots, f_d = f(y^{d_2}))^T$ and forcing model agreement with the original function $f(x)$ by ensuring

$$\begin{bmatrix} \phi_1(y^1) & \phi_2(y^1) & \ldots & \phi_{d_1}(y^1) \\ \phi_1(y^2) & \phi_2(y^2) & \ldots & \phi_{d_1}(y^2) \\ & & \vdots & \\ \phi_1(y^{d_2}) & \phi_2(y^{d_2}) & \ldots & \phi_{d_1}(y^{d_2}) \end{bmatrix} \begin{bmatrix} \lambda_1 \\ \lambda_2 \\ \vdots \\ \lambda_{d_1} \end{bmatrix} \approx \begin{bmatrix} f_1 \\ f_2 \\ \vdots \\ f_{d_2} \end{bmatrix}. \tag{2}$$

When $d_1 = d_2$ this is called interpolation and equality is desired within (1). Note that in practice, the set Y is shifted and scaled.

Sampling issues These methods can have issues with poor sampling choices. Two important aspects of the sample points are their geometry and proximity.

Geometry For geometry, regression based methods require that the set the function is evaluated at must be Λ-poised for a fixed constant Λ. If the set is Λ-poised for a certain trust region radius, then the maximum absolute value of any Lagrange polynomial over the trust region is one. This ensures that the Vandermonde matrix used to find the coefficients used to express the model function in terms of a basis of Lagrange polynomials is well conditioned. Although we do not go into the details here, problems become apparent when comparing the Lagrange polynomials associated with a poised set with those of an ill poised set.

Within the first set of pictures below we see the set of quadratic Lagrange polynomials on the interval $[-1, 1]$. The maximum value of these polynomials over the trust region is simply 1. However, if we use sample points $\{-1, .9, 1\}$ instead of the points $\{-1, 0, 1\}$, we find the second set of polynomials.

If we use these to approximate $3 + \tan^{-1}(x)$, the first basis functions do not vary far from the objective value over the trust region, and the maximum difference between the function and the model function is 0.0711. However, the second basis functions jump far away from the actual function, and the maximum difference between the model function and actual function is 0.1817.

BETTER IMAGES COMING SOON...

Proximity Proximity refers to the trust region radius. The trust region must go to zero if we are to be sure that we have reached a critical point. In general, the smaller the trust region, the closer to linear or quadratic the original function will look. This is because the model's error term given by Taylor's expansion is proportional to the trust region radius.

2.2 Setting

It will be convenient to write $c_{\mathcal{I}}(x) = (c_1(x), c_2(x), \ldots, c_{|\mathcal{I}|})^T$, $c_{\mathcal{E}}(x) = (c_{|\mathcal{I}|+1}(x), c_{|\mathcal{I}|+2}(x), \ldots, c_{|\mathcal{I}|+|\mathcal{E}|})^T$ and $c(x) = (c_{\mathcal{I}}^T(x), c_{\mathcal{E}}^T(x))^T$.

At an iteration k, we first construct a model function $m_f^k(x)$ that we use to approximate the first and second derivatives of $f(x)$. We also construct $m_{\mathcal{I}}^k(x)$ and $m_{\mathcal{E}}^k(x)$ to approximate the first derivatives of $c(x)$. Namely, at an iterate x^k, we let $f^k = f(x^k)$, $g^k = \nabla m_f^k(x^k)$. We also define the $c_{\mathcal{I}}^k = c_{\mathcal{I}}^k(x^k)$, $A_{\mathcal{I}}^k = \nabla m_{\mathcal{I}}^k(x^k)$, $c_{\mathcal{E}}^k = c_{\mathcal{E}}(x^k)$, and $A_{\mathcal{E}}^k = \nabla m_{\mathcal{E}}^k(x^k)$.

It will also be convenient to let $\mathcal{F}^k = \{x \in \mathbb{R}^n | c_i(x) \leq 0 \forall i \in \mathcal{I} \wedge c_i(x) = 0 \forall i \in \mathcal{E}\}$ be the feasible region.

We work within a trust region, sequential quadratic programming framework with interpolating model functions.

We first compute an interpolation set poised for regressing a set of model functions, which we choose to be quadratic functions. Although we have enough sample points to construct a quadratic model of the constraints, we only construct linear models to avoid the complexity of Quadratically Constrained Quadratic Programming which is NP-hard.

2.2.1 Model functions

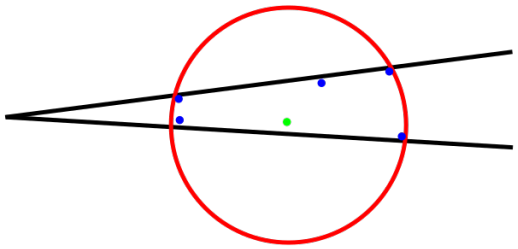
3 Algorithms

4 Simple approach

One simple approach to handling hidden constraints is to simply avoid letting points fall outside the feasible region.

Within this model, we maintain the same policies for updating the trust region radius. However, within the LU factorization algorithm to select new points, we add to the trust region subproblem that the new points must also lie within the current model of the trust region.

This can happen when the feasible region intersect the trust region is small compared to the trust region. As the number of dimensions grows, this can become worse.



5 Bumping ξ

Within the classical algorithm, we used ξ as a lower bound of the pivot values of the Vandermode matrix. When requiring points to live within the trust region intersect the feasible region, it can happen that even after replacing a point, we still have not satisfied this bound. One way to handle this is to simply allow ξ to decrease until a threshold is reached. (The threshold is for maintaining a fixed lambda.)

6 Ellipse

We decided to keep the trust region entirely within the feasible region because of the high lambda when only requiring the points to lie in the feasible region. In order to deal with the numerical problems from having a circular trust region, we can map the feasible region back to a circle. More specifically, at iteration k, we select an ellipse center μ^k, a scaling factor π^k and positive definite matrix Q^k to define and ellipse $E_k = \{x \in \mathbb{R}^n \| \pi^k - \frac{1}{2}(x - \mu^k)^T Q^k (x - \mu^k) \geq 0\}$. We then map this back to the origin with the affine transformation $T^k : \mathbb{R}^n \to R$ $T^k(x) = \frac{\sqrt{2}}{\pi^k} L^k (x - \mu^k)$ where $L = cholesky(Q)$. This will allow us to construct a lambda poised set within narrow trust regions.

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

- How do we ensure that $x^k \in E^k$?
- How do we choose the center of the ellipse μ^k?

- Is $E_k \subset F_k$?

We have tried a couple of ways of ensuring the current iterate is within the trust region, $x^k \in E^k$. If we do not include the current iterate within the interior , we likely lose sufficient reduction. This can be done by either expanding the radius of the ellipse, or by including the original point as a constraint for the ellipse problem.

In order to include the original point as a constraint, we simply add the following constraint to the definition of the ellipse:

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

However, the optimization problem we construct to find Q^k solves for ${Q^k}^{-1}$, so that adding this constraint is a constraint on the inverse of Q^k.

The alternative is to scale Q by a constant. To do this, we use the scaling factor π^k by defining it to be

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

and let the ellipse be:

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

However, this means that we do not ensure $E_k \subset F_k$ so that the trust region subproblem must contain constraints for both the ellipse and the feasible region.

The are some concerns we have to consider while defining the ellipse. One major concern is that we can accidently define the ellise in a way that forces the trust region away from the desired drection. Also, unless we include points outside the ellipse, we cannot find a second order solution because the ellipse can only be tangent to the constraint. We also want consecutive ellipse to share volume, in order to avoid evaluating as many points as possible. (So far, we have been re-evaluating the set of trial points each iteration.) Finally, we have the problem of computing the ellipse once given a suitable definition.

6.1 Finding the maximal E_k given μ^k

Within this paper, we first solve the problem of finding the maximal ellipse given the center, and then perform a simple search over different centers of the ellipse. Because of this, we will first show how to find an ellipse with maximal volume given a fixed center. For now, we also let $s^k = 1$.

Given a polyhedron P defined by an $m \times n$ matrix A,

$$P = \{x \mid Ax \leq b\},$$

and we wish to find the largest ellipse $E \subset P$ centered at a point μ^k within this polyhedron.

We can shift the polyhedron by letting $\bar{b} = b - A\mu^k$ and letting $x \to x - \mu^k$ so that the polyhedron becomes

$$P = \{x \mid Ax \leq \bar{b}\}$$

The ellipse can then be centered at zero, and defined by a positive semi-definite matrix $Q \succeq 0$:

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

where $Q = Q^T$. We can define the auxiliary function

$$f(x) = \frac{1}{2} x^T Q x$$

so that this becomes

$$E = \{d \mid f(d) \leq 1\}.$$

At the locations where the ellipse intersects the polyhedra with minimum value of f, the gradients of f must be orthogonal to the faces. Each face is defined by at least one $A_i x \leq \bar{b}_i$ where A_i is the ith row of A for $1 \leq i \leq m$. This implies that if $d^{(i)}$ is such a point, then

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

for some λ_i. This means

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

We also know that

$$A_i^T d^{(i)} = \bar{b} \quad \forall 1 \leq i \leq m$$

$$A_i^T \lambda_i Q^{-1} A_i = \bar{b} \quad \forall 1 \leq i \leq m$$
$$\lambda_i = \frac{\bar{b}}{A_i^T Q^{-1} A_i} \quad \forall 1 \leq i \leq m$$

so that

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

Finally, we need that for each i, $f(d^{(i)}) \leq 1$:

$$\frac{1}{2}(d^{(i)})^T Q d^{(i)} \leq 1$$
$$\frac{1}{2}(\frac{\bar{b}}{A_i^T Q^{-1} A_i} Q^{-1} A_i)^T Q \frac{\bar{b}}{A_i^T Q^{-1} A_i} Q^{-1} A_i \leq 1$$
$$\frac{1}{2} \frac{1}{A_i^T Q^{-1} A_i} \bar{b}^T A_i^T Q^{-1} Q \frac{\bar{b}}{A_i^T Q^{-1} A_i} Q^{-1} A_i \leq 1$$
$$\frac{1}{2} \frac{1}{A_i^T Q^{-1} A_i} \bar{b}^T \frac{\bar{b}}{A_i^T Q^{-1} A_i} A_i^T Q^{-1} A_i \leq 1$$
$$\frac{1}{2} \bar{b}^T \frac{\bar{b}}{A_i^T Q^{-1} A_i} \leq 1$$
$$\frac{1}{2} \bar{b}^T \bar{b} \leq A_i^T Q^{-1} A_i$$
$$A_i^T Q^{-1} A_i \geq \frac{1}{2} \bar{b}^T \bar{b}$$

Thus, the maximal ellipse is defined by

$$\sup_{Q \succeq 0} \det(Q^{-1})$$
$$s.t. \quad A_i^T Q^{-1} A_i \geq \frac{1}{2} \bar{b}^T \bar{b}$$

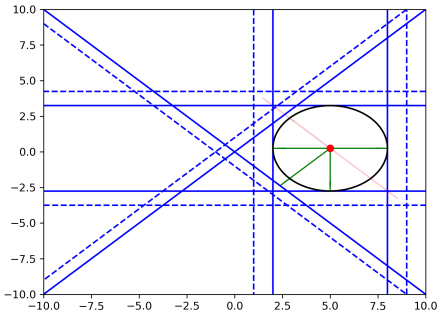
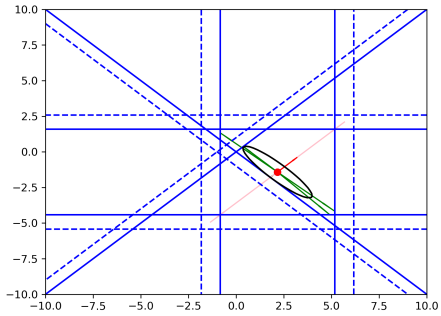
This problem includes a maximization of a determinant. In order to ensure that the trust region goes to zero, we must also ensure that the maximum eigenvalue of the matrix is bounded. Thus, for some bound M^k we define Q^k as

$$Q^k = V(\mu_k) = \sup_{Q \succeq 0} \det(Q^{-1}) \tag{3}$$
$${A^k}_i^T Q^{-1} {A^k}_i \geq \frac{1}{2}(b^k - A^k \mu^k)^T (b^k - A^k \mu^k) \tag{4}$$
$$eig(Q)_i \leq M^k \qquad \forall i \in \mathcal{I} \cup \mathcal{E} \tag{5}$$
$$\pi^k - \frac{1}{2}(x^k - \mu^k)^T Q^k (x^k - \mu^k) \geq 0 \tag{6}$$

if we wish to include the original point explicitly. This gives rise to the trust region sub problem of

6.2 Stationary center

The simplest approach is to not change the center of the ellipse, but instead let $\mu^k = x^k$. In the example below, we begin with a trust region that must be very small to lie within the feasible region, and note that if we even keep the same center, we are still able to search towards the vertex of the feasible region. After solving the trust region subproblem in the first image, the iterate is near a constraint within the next iteration. However, the ellipse elongates along an axis parallel to this constraint.

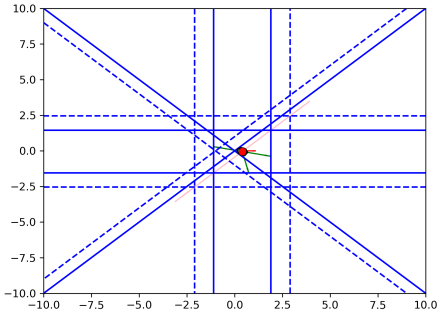
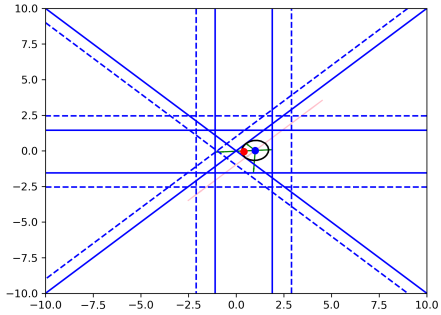



6.3 Search everything

Another simple approach is to define the ellipse as to maximize the area within the ellipse. That is, we add the constraints directly to the trust region subproblem.

$$\mu^k = \sup_{\mu \in \mathcal{F}^k} V(\mu))$$

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.

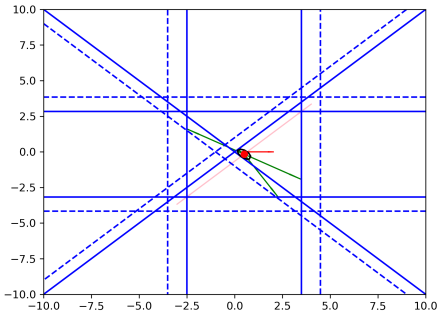
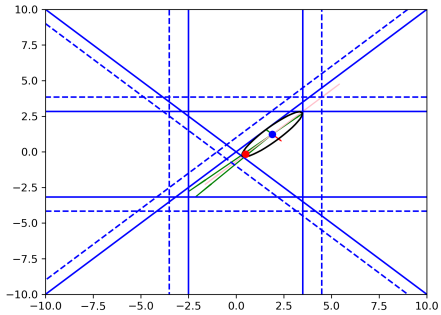
One attempt to fix this problem is by limiting the search direction for the center of the ellipse.

6.4 Line searches

Although the minimum of problem P can appear anywhere within the trust region, there are some reasons for expecting it to be at a "vertex." If it lies on the interiorer, there is little need for using constrained approaches once near the solution.

The ellipse with maximum volume, however, tends to lead away from vertices because there is more space away from vertices. One way of trying to leave the direction towards a vertex within the trust region, while still allowing to incease the ellipse volume is by limiting the search for the new center to lie on line segments starting at the current iterate.

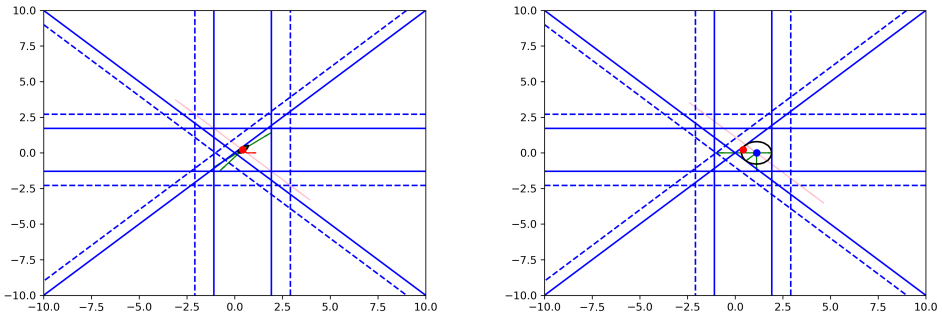
For example, our first attempt was to simply search a line directed away from the closest constraint. This has obvious problems though, as we should avoid letting the new center get closer to another constraint:

To fix this, we break the line into segments based on the nearest constraint. We find the center that maximizes the volume of the ellipse after restricting it to lie on a ray perpendicular to the nearest constraint. Nearest here means means the constraint with the smallest distance from the iterate to the projection of that iterate onto the constraint. If there are two constraints that are both closer than all others, then we follow a ray whose points are equidistant from both of these constraints.

We search along this away from the current iterate until we reach a point that has the same distance to another constraint as the nearest to the iterate. We could then in general continue to follow line segments by always travelling away from the nearest constraints. However, after we include enough constraints, we will eventually once again head away from a vertex.

This means that we can define the a class of searches that each use a different number of line segments to search. In this paper, we consider one and two line segments.



7 Results

the example problem I have been working with linear constraints

Algorithm	Iterations	Evaluations
Search everything (include the original)	24	169
Search everything	19	106
One line segment	21	86
Two line segments	23	115

References

[1] S. L. Digabel and S. M. Wild, "A taxonomy of constraints in simulation-based optimization," 2015.