Possible Future Research

Trever Hallock

March 30, 2017

1 Literature Review

The original filter method was proposed by Gould in [1]. The motivation for the filter method was that the algorithm does not need to tune any parameters as in penalty or merit methods.

A recent paper from September 2016 [2] implements a derivative-free trust region filter method for solving the general program

$$\begin{aligned} \underset{x}{\text{minimize}} \quad & f(x, y, z) \\ \text{subject to} \quad & g_i(x, y, z) \leq 0 \quad \forall i = 1, \ldots, m_1 \\ & h_i(x, y, z) = 0, \quad \forall i = 1 \ldots, m_2 \\ & y = d(z) \end{aligned}$$

where d is a black box function while f, g and h are glass box functions. This is more general than the algorithm we consider as it allows for the objective to depend on other glass box functions of the input and multiple outputs of the black box function. The authors compare their algorithm to finite difference methods as well as kriging on three different applications from Chemistry. Within the algorithm, the current iterate is always feasible with respect all inequalities except for the constraint $y = d(z)$.

In the 2006 paper [3] Fletcher reviews how filter methods have developed. The only references to derivative-free versions of the filter method are those applied to pattern based (direct) methods. However, the authors outline trust region filter methods along with several other variants.

In Brekelman's paper [4], a trust region filter method is developed to minimize function evaluations by constructing linear model functions. This paper also uses experimental designs to choose following iterates.

Within [5] and [6] Biegler uses a filter method to ensure global convergence within a line search framework. We experimented with this before deciding combining line search with the derivative-free trust region approach was not a natural approach. (There have been attempts to employ both of these frameworks in [7].)

Within [8] derivative-free methods are developed in detail. This contains a good explanation of ensuring geometry of the current set with poisedness for unconstrained problems and also covers other derivative-free methods including direct-search and line search.

Within [9] Toint generalizes the filter method with the notion of a trust funnel. This is for glass box functions as it does not include any derivative-free methods.

Within [10] a sequential quadratic programming method is applied to the filter method?

Colson has also applied filter techniques to derivative-free optimization in his 2004 Ph.D. thesis [11]. This focuses on bilevel programming.

I based my algorithm of the trust region filter method described in:

GLOBAL CONVERGENCE OF A TRUST-REGION SQP-FILTER ALGORITHM FOR GENERAL NONLINEAR PROGRAMMING ROGER FLETCHER, NICHOLAS I. M. GOULD, SVEN LEYFFER, PHILIPPE L. TOINT, AND ANDREAS WACHTER I need to find a bibtex reference of this paper to put in the bibliography.

2 Review of Model Based DFO methods

2.1 Optimization with no derivatives

- Evaluating $f(x)$ may involve running a simulation
- The runtime of $f(x)$ may mean that typical finite difference methods are too expensive
- The derivative-free philosophy is to avoid function evaluations

2.2 Derivatives are convenient

- Quadratic convergence in line search methods require second order derivatives
- They appear in conditions for convergence results (Wolf, Armijo, or Goldstein for line search)
- Stopping criteria are based on optimality conditions, which frequently involve derivatives

2.2.1

Consider the following optimization problem

$$\begin{aligned} \underset{x}{\text{minimize}} \quad & f(x, S(x)) \\ \text{subject to} \quad & g_i(x, S(x)) \leq 0,\ i = 1, \ldots, m \\ & h_i(x, S(x)) = 0,\ i = 1, \ldots, n \end{aligned}$$

This gives rise to different classes of derivative-free optimization based on f, g_i, and h_i:

- They may be noisy
- Derivatives of g_i and h_i may be known
- g_i and h_i can sometimes be "Hidden" constraints
- Sometimes the functions g_i and h_i are only found by evaluating $S(x)$
- Many DFO methods simply let $f(x, S(x)) = S(x)$

A slightly more general form is:

$$\begin{aligned} \underset{x}{\text{minimize}} \quad & f(x, y, z) \\ \text{subject to} \quad & g_i(x, y, z) \leq 0 \quad \forall i = 1, \ldots, m_1 \\ & h_i(x, y, z) = 0, \quad \forall i = 1 \ldots, m_2 \\ & y = d(z) \end{aligned}$$

2.3 Approaches

- Finite difference methods
  - $\nabla f(x) \approx (\frac{f(x+he_i)-f(x-he_i)}{2h})_i$ for some h
  - The number of function evaluations can grow large quickly
- Direct search methods
  - Coordinate descent and Pattern based
  - Nelder Mead
  - Do not use derivatives: robust but ignore useful information
- Model-based methods
  - This is what we will discuss

2.4 The Derivative Trust Region Method

- Interpolate or regress model functions onto a sample
- Minimize the model function over a trust region
- Adjust trust region, possibly by testing the model's accuracy

2.5 More details

1. Define $m_k(x) = f(x^{(k)}) + \nabla f(x^{(k)})^T(x - x^{(k)}) + \frac{1}{2}(x - x^{(k)})^T \nabla^2 f(x^{(k)})(x - x^{(k)})$
   - $\nabla f(x^{(k)})$ and $\nabla^2 f(x^{(k)})$ must be approximated
   - There are geometric properties of the sample set that must be satisfied
2. if $\nabla m_k(x) < t$ stop
3. Solve the Trust region subproblem: $s^{(k)} = \arg\min_{s \in B_(x^{(k)}, \Delta_k)} m_k(x^{(k)} + s)$
4. Test for improvement
   - $\rho_k = \frac{f(x^{(k)}) - f(x^{(k)} + s^{(k)})}{m_k(x^{(k)}) - m_k(x^{(k)} + s^{(k)})}$
   - 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 increase radius

3 Possible future work

3.1 Future Directions

- Convert classical NLP Algorithms to DFO
- Different Goals
- Parallelization
- Explore different model functions
- Providing structure to the optimization program

3.2 NLP Methods

We can convert classical algorithms to DFO

- Take an existing constrained Nonlinear programming method
  - Filter method
  - Active Set
  - Augmented Lagrangian
  - Interior Point
  - Line search
- Replace derivatives of f, g_i, h_i with derivatives of a model function

3.3 Goals

We could consider an objective function with a evaluation runtime that varies with x.

3.4 Parallelization

1. parallelization of the function and/or the derivative evaluations in the algorithm
2. parallelization of linear algebra kernels
3. modifications of the basic algorithms which increase the degree of intrinsic parallelism, for instance, by performing multiple function and/or derivative evaluations

The third item is particularily interesting.

3.5 Miscellaneous

- Model Functions Radial basis functions
- Problem Structure Specify the structure of f, g_i, or h_i
- Machine Learning Applying DFO techniques to algorthms that currently use random sampling to tune parameters

4 My Algorithm

4.1 Problem

We consider problems of the form

$$\begin{aligned} &\min_x f(x) \\ &\quad g(x) \leq 0 \\ &\quad h(x) = 0 \end{aligned}$$

where $f : R^n \to R$, $g : R^n \to R^{m_1}$ and $h : R^n \to R^{m_2}$. We assume that all functions are derivative-free: all functions f, g, h are evaluated by a single call to a black box function $d(x) = (f(x), g(x)^T, h(x)^T)^T$.

4.2 First approach

We began with a line search filter method, however we found that this had several drawbacks:

- Trust regions arise naturally within derivative-free algorithms
- Line search algorithms exploit how much easier finding a descent direction is than solving the trust region subproblem, but this saved computation is not as useful in DFO
- As the algorithm backtracks on the step length, the trust region must be reduced, as the models are accurate over a region rather than at a single point

4.3 Algorithm in other paper

We work within a trust region, sequential quadratic programming framework that uses a filter method introduced by Fletcher.

We first compute an interpolation set poised for regressing a set of model functions, which we choose to be quadratic functions. Although we have function values for enough points to construct model functions of the same order as used for the objective, we model the feasible with only the linear constraints.

The core of the algorithm revolves around

4.3.1 Criticality Measure

In order to construct stopping criteria, we introduce a criticality measure χ which goes to zero as the iterates approach a first order critical point.

This is defined as

$$\begin{aligned} \chi = & \quad |\min_t \langle g_k + H_k n_k, t\rangle| \\ A_{eq}t & = 0 \\ c_{ineq} + A_{ineq}t & \leq 0 \\ \|t\| & \leq 1 \end{aligned}$$

4.3.2 Step decomposition

At iteration k, we can decompose the step s_k into a normal step n_k intended to decrease constraint violation and a tangential step t_k intended to reduce the objective. The step n_k projects the current iterate onto the feasible region. Currently, we project x_k onto only the linear model of the feasible region. We require that the the computation of the normal step, which solves:

$$\begin{array}{lll} & n_k = & \arg\min_n \|n\|^2 \\ s.t. & c_{eq} + A_{eq} n & = 0 \\ & c_{ineq} + A_{ineq} n & \leq 0 \\ & \|n\|^2 & \leq \Delta_k^2 \end{array}$$

In addition to this program having a feasible point, we need to know that there is enough space for us to provide sufficient decrease within the tangential step. This means that we require the stronger condition that

$$\|n\| \leq \kappa_\Delta \Delta_k \min\{1, \kappa_\mu \Delta_k^\mu\}$$

If this condition is satisfied, then we say that the program is *compatible*. We are then able to compute a tangential step t_k:

$$\begin{array}{lll} & t_k = & \arg\min_t (g_n + H_k n_k)^T t + \frac{1}{2} t^T H_k t \\ s.t. & c_{eq} + A_{eq} t & = 0 \\ & c_{ineq} + A_{ineq} t & \leq 0 \\ & \|n_k + t_k\|^2 & \leq \Delta_k^2 \end{array}$$

- quadratic information contained in H_k
- H_k is the hessian of the lagrangian, as computed by using KKT the matrix
- This program is a shifted version of another

4.3.3 Compatibility

4.3.4 The filter

The filter is a method used to ensure convergence to a feasible point. It works by ensuring that all new iterates are nondominated with respect to all previous iterates when the problem is viewed as a multi-criteria optimization problem $\min(\theta, f)$. However, simply ensuring that new points are nondominated does not provide sufficient progress, we must ensure that the objective decreases by a greater amount when the current iterate is far from the feasible region:

$$\theta(x_k) \leq (1 - \gamma_\theta)\theta_i$$

or

$$f(x_k) \leq f_i - \gamma_\theta \theta_i$$

for all (θ_i, f_i) that are currently in the filter. When this condition is satisfied, we say that the new iterate x_k is admissible to the filter.

4.3.5 f-type versus θ-type

4.3.6 Restoration Step

The goal of the feasibility restoration step is to find a new iterate and trust region radius that allows the current iterate to be compatible.

While performing the restoration step, we place constraints in the objective by minimizing the squared norm of θ. We take one step to minimize the quadratic model unconstrained optimization problem, and update the trust region based on the new function value.

It is possible that the restoration step is unsuccessful if the iterates approach an infeasible local minimum of the constraints. In this case, the algorithm will fail to find a feasible local minimum, and will need to be restarted.

4.4 When we use derivative-free methods

4.4.1 The change in the criticality measure

One issue with applying the original algorithm within a DFO context was that the trust region radius is not required to go to zero. However, within DFO we must also require that the trust region goes to zero as we approach a stationary point. One way of ensuring this is to decrease the trust region radius when the current step lies well within the trust region radius. However, a better approach may be to introduce a tolerance on the criticality measure and decrease the trust region whenever the criticality falls below the threshold.

4.5 Algorithm Description

4.5.1 Simplified version draft

- compute model functions and hessian of the lagrangian
- compute criticality measure
- if feasible and critical, then decrease trust region radius or return
- compute normal step
- check compatibility, restoring feasibility if necessary and return to step 1
- compute tangential step
- check for sufficient reduction in the model function, adding the current iterate to the filter and returning to step 1 if necessary
- evaluate the function and constraints at the trial point
- check compatibility to the filter, decreasing the trust region radius and returning to step 1 if necessary
- compute ρ, and accept the trial point or reject and decrease the radius

4.5.2 Psuedo code

Algorithm 1 Filter Trust Region Search

```
procedure TRUST REGION FILTER
    initialize
    $k = 0$
    choose an $x_0$
    while $k < maxit$ do
main loop:
        ensure poisedness, possibly adding points to the model
        Compute $m_k, g_k = \nabla m_k(x_k), c_k, A_k, f_k = f(x_k), \mathcal{A}, \theta_k$
        Solve:
            $\begin{aligned} \nabla^2 m_k(x_k)d + A_k^T\lambda &= g_k \\ A_k d &= c_k \end{aligned}$
        $H_k \leftarrow \nabla^2 m_k(x_k) + \sum_i \lambda_i \nabla^2 c_{ik}$
        $\chi_k \leftarrow |\min_t\{\langle g_k + H_k n_k, t\rangle | A_{eq}t = 0 \wedge c_{ineq} + A_{ineq}t \leq 0 \wedge \|t\| \leq 1\}|$
        if constraint violation $= 0 \wedge \chi = 0$ then
            if $tol < \Delta_k$ then
                reduce $\Delta$: $\Delta_{k+1} \leftarrow$ some $\in [\gamma_0\Delta_k, \gamma_1\Delta_k]$
                $k \leftarrow k + 1$
                go to main loop
            success
        $n_k \leftarrow \arg\min_n\{\|n\|^2 | c_{eq} + A_{eq}n = 0 \wedge c_{ineq} + A_{ineq}n \leq 0 \wedge \|n\|^2 \leq \Delta_k\}^2$
        if Feasible region $\neq \emptyset \wedge \|n\| \leq \kappa_\Delta \Delta_k \min\{1, \kappa_\mu \Delta_k^\mu\}$ then
            $t_k \leftarrow \arg\min_t\{(g_n + H_k n_k)^T t + \frac{1}{2}t^T H_k t | c_{eq} + A_{eq}t = 0 \wedge c_{ineq} + A_{ineq}t \leq 0 \wedge \|s\| \leq \Delta_k\}$
            $s_k \leftarrow t_k + n_k$
            if $m_k(x_k) - m_k(x_k + s_k) \geq \kappa_\theta \theta_k^\psi$ then
                add $x_k$ to filter
                reduce $\Delta$: $\Delta_{k+1} \leftarrow$ some $\in [\gamma_0\Delta_k, \gamma_1\Delta_k]$
                go to main loop
            // Here we evaluate new $c$ and $f$ at $x_k + s_k$
            if $x_k + s_k$ is acceptable: $\theta(x_k + s_k) \leq (1 - \gamma_\theta)\theta' \vee f(x_k + s_k) \leq f' - \gamma_\theta\theta' \forall (f', \theta') \in$ Filter then
                $\rho = \frac{f(x_k) - f(x_k + s_k)}{m_k(x_k) - m_k(x_k + s_k)}$
                if $\rho < \eta_1$ then
                    reduce $\Delta$: $\Delta_{k+1} \leftarrow$ some $\in [\gamma_0\Delta_k, \gamma_1\Delta_k]$
                    $k \leftarrow k + 1$
                    go to main loop
                else if $\rho > \eta_2$ then
                    if $\|s\| < \frac{\Delta_k}{2}$ then
                        reduce $\Delta$: $\Delta_{k+1} \leftarrow$ some $\in [\gamma_0\Delta_k, \gamma_1\Delta_k]$
                    else
                        increase $\Delta$: $\Delta_{k+1} \leftarrow$ some $\in [\Delta_k, \gamma_2\Delta_k]$
                $x_{k+1} \leftarrow x_k + s_k$
            else
                reduce $\Delta$: $\Delta_{k+1} \leftarrow$ some $\in [\gamma_0\Delta_k, \gamma_1\Delta_k]$
                $k \leftarrow k + 1$
                go to main loop
        else
            add $x_k$ to filter
            compute new $r$ (restoration step) and $\Delta$
            if impossible to restore then fail
            $x_{k+1} \leftarrow x_k + r$
        $k \leftarrow k + 1$
```

5 Pictures of convergence

6 Comparison to other libraries (If I get enough time)

References

[1] N. I. M. Gould and P. L. Toint, "Nonlinear programming without a penalty function or a filter," *Mathematical Programming*, vol. 122, no. 1, pp. 155–196, 2010.

[2] J. P. Eason and L. T. Biegler, "A trust region filter method for glass box/black box optimization," *AIChE Journal*, vol. 62, no. 9, pp. 3124–3136, 2016.

[3] R. Fletcher, S. Leyffer, R. Fletcher, and S. Leyffer, "A brief history of filter methods," tech. rep., 2006.

[4] R. Brekelmans, L. Driessen, H. Hamers, and D. den Hertog, "Constrained optimization involving expensive function evaluations: A sequential approach," *European Journal of Operational Research*, vol. 160, no. 1, pp. 121 – 138, 2005. Applications of Mathematical Programming Models.

[5] A. Wächter and L. T. Biegler, "Line search filter methods for nonlinear programming: Motivation and global convergence," *SIAM Journal on Optimization*, vol. 16, no. 1, pp. 1–31, 2005.

[6] A. Wächter and L. T. Biegler, "Line search filter methods for nonlinear programming: Local convergence," *SIAM Journal on Optimization*, vol. 16, no. 1, pp. 32–48, 2005.

[7] J. Nocedal and Y. Yuan, "Combining trust region and line search techniques,"

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

[9] R. Sampaio and L. Toint, "A trust-funnel method for nonlinear optimization problems with general nonlinear constraints and its application to derivative-free optimization, most information in this cite is wrong," *NAXYS Namur Center for Complex Systems*, vol. 61, no. 5000, pp. 1–31, 2015.

[10] P. T. Boggs and J. W. Tolle, "Sequential quadratic programming for large-scale nonlinear optimization," *Journal of Computational and Applied Mathematics*, vol. 124, no. 12, pp. 123 – 137, 2000. Numerical Analysis 2000. Vol. IV: Optimization and Nonlinear Equations.

[11] B. Colson, "Trust-region algorithms for derivative-free optimization and nonlinear bilevel programming," *Quarterly Journal of the Belgian, French and Italian Operations Research Societies*, vol. 2, no. 1, pp. 85–88, 2004.