AIChE

Advanced Trust Region Optimization Strategies for Glass Box/Black Box Models

John P. Eason and Lorenz T. Biegler
Dept. of Chemical Engineering, Carnegie Mellon University, Pittsburgh, PA, 15213

DOI 10.1002/aic.16364
Published online September 3, 2018 in Wiley Online Library (wileyonlinelibrary.com)

We present an improved trust region filter (TRF) method for optimization of combined glass box/black box systems. Glass box systems refer to models that are easily expressed in an algebraic modeling language, providing cheap and accurate derivative information. By contrast, black box systems may be computationally expensive and derivatives are unavailable. The TRF method, as first introduced in our previous work (Eason and Biegler, AIChE J. *2016; 62:3124–3136), is able to handle hybrid systems containing both glass and black box components, which can frequently arise in chemical engineering, for example, when a multiphase reactor model is included in a flow sheet optimization problem. We discuss several recent modifications in the algorithm such as the sampling region, which maintains the algorithm's global convergence properties without requiring the trust region to shrink to zero in the limit. To benchmark the development of this optimization method, a test set of problems is generated based on modified problems from the CUTEr and COPS sets. The modified algorithm demonstrates improved performance using the test problem set. Finally, the algorithm is implemented within the Pyomo environment and demonstrated on a rigorous process optimization case study for carbon capture.* © 2018 American Institute of Chemical Engineers *AIChE J*, 64: 3934–3943, 2018
Keywords: Simulation, process, optimization, adsorption/gas, mathematical modeling

Introduction

Nonlinear optimization has matured significantly over recent decades and has become an important tool in process systems engineering. Powerful algorithms and software exist for solving many problem classes: constrained and unconstrained, convex and nonconvex, and derivative free, first-order, and second-order methods. In this work, we focus on the case of nonconvex nonlinear programming (NLP), where some of the derivative information may not be available. This is captured by the term "glass box/black box" optimization, where glass box refers to equations that yield accurate derivatives, and black box refers to parts of the model that are derivative free, for example, generated by an external simulation call. This work is motivated by the case of process optimization problems where unit models are modeled with external simulations, yet we also emphasize generality for a wider class of optimization problems.

For general nonlinear programming with first and possibly second-order derivatives, mature codes are available, using various algorithms including active set methods[1-3] and primal–dual interior point methods.[4-6] Similarly, derivative free optimization (DFO) methods are well developed in the unconstrained or box constrained setting.[7] Equation oriented modeling approaches exist so as to obtain derivatives for gradient-based optimization. By contrast, simulation models are often paired with derivative free methods.

However, not all optimization problems of interest fit neatly into one of these problem classes. In particular, multidisciplinary models are becoming more and more common. Computational models are increasingly complex, and frequently draw on knowledge from various domains, or use multi-scale modeling paradigms to model across length- and time-scales. As a result, a single optimization problem may need to model subsystems that are represented with various external software or simulations. In some circumstances, external models are computationally cheap with accurate derivative estimates available. In this case, these external functions and their derivatives may be provided to an optimization solver, for example, through the AMPL Solver Interface Library (ASL) external function interface.[8] In the case that some part of the model cannot be integrated through this method, the problem is a hybrid "glass box/black box" optimization problem, sometimes termed gray box optimization.

One approach for these hybrid glass box/black box systems is to simply apply DFO for the whole system. However, this strategy is restricted by inherent limitations of DFO methods. First, DFO methods do not scale well on problems with many degrees of freedom. In addition, many DFO methods are limited in handling constraints, especially when the constraints are coupled with the black box. For example, this occurs when a black box unit operation is embedded within a recycle loop of a chemical process model. To apply DFO to the degrees of freedom, the recycle loop must be converged repeatedly, requiring many computationally expensive calls to the black box. In a general constrained setting for DFO, Powell's COBYLA[9] was the earliest

Additional Supporting Information may be found in the online version of this article.

Correspondence concerning this article should be addressed to L. T. Biegler at biegler@cmu.edu.

available solver. Additional solvers handle inequality constraints through a penalty or barrier approach (e.g., NOMAD[10, 11]). The recent trust funnel method[12] is able to handle equality constraints as well. For a recent review covering constrained DFO and applications, see the work by Boukouvala et al..[13] However, we note that most methods do not have specific capabilities to take advantage of information that may be available from glass box subsystems, and instead treat the overall system as black box.

In the engineering literature, glass box/black box problems are often solved with the use of reduced models (also known as surrogate models, model functions, or meta-models). The black box function (or truth model) is replaced by an approximation that is compatible with the equation oriented optimization environment, resulting in a fully equation oriented problem that can be solved with off-the-shelf solvers. The literature surrounding reduced models and their use in optimization is vast, and a full review is beyond the scope of this work. For construction of reduced models, Bhosekar and Ierapetritou[14] provide a recent review from a chemical engineering perspective. The use of simple reduced models for physical properties has a long history in chemical process flow-sheeting.[15-17] Recently, this concept has been extended to entire process units, using reduced models such as artificial neural networks[18] or Kriging interpolation.[19] Global optimization of glass box/black box systems has been proposed with sequential construction of reduced models,[20, 21] although there are no guarantees to converge to the original, truth model solution. Other researchers look to developments in machine learning to build accurate reduced models with simple functional forms.[22, 23]

The accuracy of the optimization depends strongly on the accuracy of the reduced model. From early work by Alexandrov et al.[24] and Arian et al.,[25] trust region methods have offered some guaranteed convergence properties for optimization with reduced models. A trust region strategy for constrained optimization was implemented for the DAKOTA package,[26] but convergence was not directly proved. The ORBIT algorithm solves unconstrained optimization problems specifically using radial basis function reduced models.[27] Recent works extend reduced model-based optimization to constrained problems using filters[28] or merit functions,[29] but here convergence proofs rely on the availability of accurate derivatives for the truth model.

A trust region filter (TRF) method specifically for solving glass box/black box problems was presented in our prior work.[30] The TRF algorithm combines filter methods for constrained nonlinear programming with model-based trust region methods for DFO. We presented the algorithm, discussed convergence, and demonstrated the ability to solve our applications of interest. In this work, we consider significant improvements to the TRF method as well as its implementation in an integrated optimization framework. This article discusses and analyzes several developments including separate trust and sampling regions, and their updating rules. We also develop a test set to benchmark improvements in the algorithm and implementation, and demonstrate performance on a chemical process example for CO_2 capture.

The structure of this article is as follows, we first review the TRF method for glass box/black box problems, then discuss our algorithmic improvements. In addition, we present a Pyomo-based implementation and finally discuss the performance of the algorithm on a set of test problems and a chemical process optimization example.

Background

The TRF method handles glass box/black box problems of the form

$$\min_{w,y,z} f(w,y,z), \quad \text{s.t. } h(w,y,z)=0,\ g(w,y,z)\le 0,\ y=d(w) \quad (1)$$

where $w \in \mathbb{R}^m$, $y \in \mathbb{R}^p$, and $z \in \mathbb{R}^n$. Functions f, h, g are all assumed to be twice differentiable on the domain $\mathbb{R}^{m+n+p}$, with accurate derivatives assumed to be available, for example, through the use of automatic differentiation. Conversely, the function $d(w) : \mathbb{R}^m \to \mathbb{R}^p$ represents the outputs of the black box as a function of inputs w and the remaining "glass box" decision variables are represented by z. To allow for convergence analysis, the black box function $d(w)$ is assumed to be twice continuously differentiable.

Because of the inherent difficulty of black box optimization problems, we assume that the black box model forms a small portion of the overall system. The number of black box inputs m is assumed to be small (less than a hundred). However, this assumes that the black box function is evaluated as a single function call. If $d(w)$ is in fact made up of several smaller simulation calls (e.g., multiple black box unit operations, or black box property calls for each stream), the dimension of w may be able to increase accordingly as long as the dimensionality of each individual simulation call is small. By contrast, the number of glass box variables n could be very large (on the order of tens of thousands), corresponding to conventional equation-based nonlinear programming models. We also introduce a variable x as the following aggregated set of variables $x^T = [w^T, y^T, z^T]$. This will simplify the presentation when we refer to all variables in formulation (1).

Algorithmic concepts

The TRF method generates a sequence of points x_k converging to a first-order Karush-Kuhn-Tucker (KKT) point of (1). This sequence maintains feasibility for glass box constraints $h(x_k) = 0$, $g(x_k) \le 0$ for all k, while simultaneously converging toward optimality of (1) and feasibility of black box constraints $y = d(w)$ with as few evaluations of the black box as possible. The detailed convergence analysis is presented in Appendix A (see Supplemental Information).

The TRF method has the following components. At each iteration, we apply a reduced model that approximates the black box $d(w)$ within a bounded domain called the trust region. Solving an optimization problem using this approximation generates a new point and the algorithm evaluates progress toward the solution. Below, we discuss the details of the trust region subproblem and the conditions we require on the reduced models. Next, we discuss the *compatibility check*, which is solved before the subproblem to ensure that it is sufficiently feasible in a certain sense. Then, we discuss the *criticality check* that is used to help determine when the algorithm can terminate. After this, we discuss the *filter mechanism* that decides how to apply the solution of the subproblem to proceed to the next iteration $k + 1$. Finally, we present the TRF algorithm and also discuss the feasibility restoration procedure that is called when the compatibility check fails.

Trust Region Subproblem. The TRF method combines features of the successive quadratic programming (SQP) filter method[31] with model-based DFO concepts introduced in Conn et al..[32] Whereas filter SQP solves a nonlinear program through a sequence of quadratic programming subproblems, the TRF method solves the glass box/black box optimization problem (1) through a sequence of NLP subproblems. The motivation for this approach is that the glass box portion of the model provides cheap derivative information that should be exploited as much as possible, while function calls to the black box should be minimized.

We will use equation-based reduced models to approximate the black box model. In particular, the TRF method uses a sequence of reduced models $r_k(w)$ approximating black box $d(w)$. Each reduced model is built to be accurate in a trust region of radius Δ_k around a point x_k. By substituting the reduced model in place of the black box and restricting the optimization to stay within the trust region where the reduced model is accurate, we form the trust region subproblem for iteration k (TRSP_k) as follows:

$$\min_x f(x), \quad \text{s.t. } h(x)=0,\ g(x)\le 0,\ y=r_k(w),\ \|x-x_k\|\le\Delta_k. \quad (2)$$

The TRF algorithm solves a sequence of trust region subproblems to converge to the solution of (1). One important ingredient to guarantee convergence is the following accuracy condition for $r_k(w)$. First we denote the trust region as the ball of radius Δ_k centered at x_k, that is, $B(x_k, \Delta_k) := \{x : \|x - x_k\| \le \Delta_k\}$. Then, we define the κ-fully linear property as follows:

Definition 1. (κ-fully linear model) *A model $r_k(w)$ is κ-fully linear approximation of $d(w)$ on $B(w_k, \Delta_k)$ if*

$$\|r_k(w)-d(w)\|\le\kappa_f\Delta_k^2 \quad \text{and} \quad \|\nabla r_k(w)-\nabla d(w)\|\le\kappa_g\Delta_k \quad (3)$$

for all $w \in B(w_k, \Delta_k)$ and for some finite $\kappa_g > 0$ and $\kappa_f > 0$ independent of k.

The r_k models within the TRF method are all required to be κ-fully linear, which means they are considered suitably accurate over a domain of radius Δ_k. The constants κ_f and κ_g derive from Lipschitz constants on r, d, ∇r, and ∇d as well as the sample geometry.[32] Conditions for κ-fully linearity of general smooth interpolation functions are given in Theorem 4.1 in Wild.[33] This agnostic approach to the functional form of the reduced model provides great flexibility in customizing a reduced model to a specific black box $d(w)$. If some knowledge is available from the application as to what functional form to expect, this can be used directly to improve optimization performance. However, most theory assumes that the black box is differentiable with Lipschitz continuous gradients.

Note that the trust region constraint is written for all variables x rather than just black box inputs w. If glass box variables were not bounded by the trust region, that is, if the trust region constraint only bounds $\|w - w_k\| \le \Delta_k$, then subproblem TRSP_k may become unbounded, even if the original problem (1) has a unique minimizer. Moreover, the convergence proof assumes that the solution to the trust region subproblem converges to x_k as the trust region converges to zero. This form of the trust region constraint helps enforce that assumption.

Compatibility Check. After building a reduced model r_k and adding the trust region constraint of radius Δ_k, TRSP_k (2) may not be feasible, even if (1) is feasible. Before attempting to solve (2), we first check a condition called the *compatibility* of TRSP_k, defined as follows.

Definition 2. (Compatibility). *If there exists a point $\hat{x}$ feasible for TRSP_k (2) such that $\|\hat{x}-x_k\|\le\kappa_\Delta\Delta_k\min\left[1,\kappa_\mu\Delta_k^\mu\right]$, where $\kappa_\Delta \in (0, 1)$, $\mu \in (0, 1)$ and $\kappa_\mu > 0$ are fixed parameters, then TRSP_k is compatible.*

The following optimization problem is used to check compatibility:

$$\begin{aligned}\beta_k = \min_x \|y-r_k(w)\|,\ \text{s.t. } h(x)=0, g(x)\le 0\\ \|x-x_k\|\le\kappa_\Delta\Delta_k\min\left[1,\kappa_\mu\Delta_k^\mu\right]\end{aligned} \quad (4)$$

Optimization problem (4) is initialized at $x_k=\left[w_k^T,y_k^T,z_k^T\right]^T$ and the solution is denoted as $x_{c,k}$. If the optimum objective value β_k of (4) is greater than zero (or in practice, a small positive tolerance), the subproblem TRSP_k is incompatible and a feasibility restoration phase is invoked. Otherwise, the subproblem is compatible and may be solved. Simply put, the compatibility check ensures a feasible point close enough to the trust region center, so that the TRSP_k will have room to explore the feasible set and improve the objective function.

Criticality Check. In addition to the compatibility check, a criticality check (for convergence) is also solved at each iteration. The criticality check calculates a measure $\chi(x)$, which is designed to approach zero as x approaches an optimal point of (2) when the trust region constraint is eliminated. This acts as a proxy for an optimality measure of (1). Because of the use of the κ-fully linear property (3), the trust region Δ_k is forced to zero by shrinking Δ_k when $\chi(x_k)$ is small. This guarantees accurate derivatives of $r_k(w)$ in the limit, thus leading to optimality of problem (1). The criticality measure $\chi_k := \chi(x_k)$ is determined by the following optimization problem:

$$\begin{aligned}\chi_k = |\min_v\ \nabla f(x_k)^T v|\quad \text{s.t.}\nabla h(x_k)^T v=0, g(x_k)+\nabla g(x_k)^T v\le 0,\\ v_y-\nabla r_k(w_k)^T v_w=0,\ \|v\|_\infty\le 1\end{aligned} \quad (5)$$

where the partitioning of v into v_w, v_y, and v_z corresponds to that of x. The *criticality check* is given by $\chi_k < \xi\Delta_k$, where ξ is a tuning parameter. If this criticality check holds then the trust region radius Δ_k is decreased.

Filter Method for Global Convergence. After solving the trust region subproblem (2), the TRF method decides whether or not to accept the proposed step. The proposed step at iteration k is defined as $s_k := \bar{x}_k - x_k$, where $\bar{x}_k$ is a minimizer of (2). If the step s_k shows sufficient progress toward a solution of (1), then the step is acceptable, that is, $x_{k+1} := x_k + s_k$. Otherwise, the step is rejected, that is, $x_{k+1} := x_k$.

To determine sufficient progress, we use the notion of a filter originally developed by Fletcher and Leyffer.[34] The filter acts to balance improvements toward feasibility with improvements in the objective function. In the TRF method, feasibility refers to the accuracy of the reduced model. This is quantified using a feasibility measure θ, defined as follows: $\theta(x) := \|y - d(w)\|_1$. In particular, we use the notation $\theta_k := \theta(x_k)$ and $f_k := f(x_k)$, where we recall that $x_k^T=\left[w_k^T,y_k^T,z_k^T\right]$. It is clear that $\theta(x) = 0$ for all feasible points of (1). However, when a trust region subproblem (2) is solved with $r_k(w)$ in place of $d(w)$, its solution $\bar{x}$ typically satisfies $\theta(\bar{x})>0$.

At certain iterates during the solving process, ordered pairs $(\theta_k, f(x_k))$ are stored in the filter set. The set of iterates where these ordered pairs are added to the filter set is denoted as $\mathcal{Z}\subset\mathbb{N}$. Thus, the filter is defined as the set $\mathcal{F}_k=\left\{\left(\theta_j,f\left(x_j\right)\right):j<k,\ j\in\mathcal{Z}\right\}$. When it is determined that $k\in\mathcal{Z}$, we say that x_k is added to the filter at iteration k. These filter pairs are interpreted as a kind of Pareto front in the minimization of f and θ, as shown in Figure 1.

In the TRF method, the filter is used to determine the success of the step s_k. If for all $(\theta_j, f_j) \in \mathcal{F}_k \cup (\theta_k, f_k)$,

$$\theta(x_k+s_k)\le(1-\gamma_\theta)\theta_j \ \text{ or } \ f(x_k+s_k)\le f_j-\gamma_f\theta_j \quad (6)$$

then the step s_k is acceptable to the filter, where $\gamma_\theta, \gamma_f \in (0, 1)$ are fixed parameters.

If a step s_k is deemed acceptable to the filter, then we further classify it as either an f-type step or θ-type step using the switching condition:

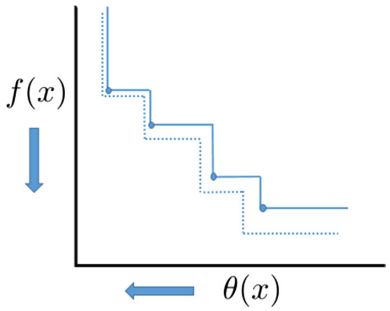

Figure 1. Convergence of the filter.
[Color figure can be viewed at wileyonlinelibrary.com]

$$f(x_k) - f(x_k + s_k) \geq \kappa_\theta \theta_k^{\gamma_s} \tag{7}$$

where $\kappa_\theta \in (0, 1)$ and $\gamma_s > \frac{1}{1+\mu}$ are tuning parameters. For successful steps s_k, if (7) is true, we say that the step s_k is an "*f*-type step". Otherwise it is a θ-type step. When s_k is a θ-type step, x_k is added to the filter. For more detailed explanations about filter methods see Refs. 31, 34, 35.

The new trust region radius Δ_{k+1} is determined based on whether step s_k is *f*-type, θ-type, or rejected. If a step s_k is unacceptable to the filter, then the trust region radius is decreased. Because we require that reduced models are κ-fully linear, a smaller trust region also tightens the accuracy requirement on the reduced model. This means that it may be necessary to improve reduced model accuracy when the trust region decreases. If s_k is acceptable and is an *f*-type step, then the trust region radius may be increased to take larger steps. For θ-type steps, the convergence theory of Fletcher et al.[31] allows great flexibility in updating the trust region radius. We use the following update rule, modified from the classical ratio test for trust region methods.

We define ρ_k to represent the actual reduction of θ divided by the predicted reduction:

$$\rho_k = \frac{\theta(x_k) - \theta(x_k + s_k)}{\theta^r(x_k) - \theta^r(x_k + s_k)} = \frac{\theta(x_k) - \theta(x_k + s_k)}{\|y_k - r_k(w_k)\|} \tag{8}$$

where $\theta^r(x) = \|y - r_k(w)\|$ and $\theta^r(x_k + s_k) = 0$, from solution of (2). This update is based on $\theta(x)$ to control the error in the reduced model. On θ-type steps, the trust region is updated using the following strategy, with parameters $0 < \eta_1 \leq \eta_2 < 1$ and $0 < \gamma_c < 1 \leq \gamma_e$.

$$\Delta_{k+1} = \begin{cases} \gamma_c \Delta_k & \text{if } \rho_k < \eta_1, \\ \Delta_k & \text{if } \eta_1 \leq \rho_k < \eta_2, \\ \gamma_e \Delta_k & \text{if } \rho_k \geq \eta_2. \end{cases} \tag{9}$$

Filter acceptance is the only requirement for step acceptance, that is, $x_{k+1} = x_k + s_k$ for all *f*-type and θ-type steps.

Restoration. The feasibility restoration procedure is called whenever TRSP_k is not compatible. Given a point x_k, a restoration procedure is any algorithm that returns x_{k+1}, Δ_{k+1}, and $r_{k+1}(w)$ such that x_{k+1} is acceptable to the filter and TRSP_{k+1} is compatible. A sufficient condition for these requirements is finding a feasible point where $\theta(x) = 0$. Therefore, feasibility restoration usually involves minimizing $\theta(x)$. In addition to θ-type steps, the filter is augmented whenever restoration is called, that is, (θ_k, f_k) is added to the filter when $TRSP_k$ is not compatible. A basic restoration procedure is presented as follows:

Restoration Algorithm.

1. Initialize restoration phase. For restoration called at iteration k, set $x_0 = x_k$ and $\Delta_0 = \Delta_k$. Filter $\mathcal{F}_k$ remains unmodified, and all other algorithmic parameters inherit values from the main algorithm. Set restoration iteration counter $i := 0$.
2. Generate a reduced model $r_i(w)$ that is κ-fully linear on $B(x_i, \Delta_i)$.
3. Solve compatibility check (4) using reduced model $r_i(w)$. Denote the solution as $x_{c,i}$ and optimal objective function value as β_i. If $\beta_i = 0$ and x_i is acceptable to the filter $\mathcal{F}_k$, then STOP restoration. Set $x_{k+1} = x_i$, $\Delta_{k+1} = \Delta_i$, and return to the main algorithm.
4. Evaluate $\theta(x_{c,i})$ and calculate $\rho_i = \frac{\theta(x_i) - \theta(x_{c,i})}{\|y_i - r_i(w_i)\| - \beta_i}$.
5. If $\rho_i \geq \eta_1$, then go to Step 6. Else, set $x_{i+1} = x_i$, $\Delta_{i+1} = \gamma_c \Delta_i$, and $i := i + 1$ and go to Step 1.
6. Accept the restoration step: $x_{i+1} = x_{c,i}$. If $\rho_i \in (\eta_1, \eta_2]$, then set $\Delta_{i+1} = \Delta_i$. Else $\rho_i > \eta_2$ and set $\Delta_{i+1} = \gamma_e \Delta_i$. Set $i := i + 1$ and go to Step 1.

The Original TRF Algorithm. The algorithm is written as follows[30]:

Algorithm 1.

1. *Initialization:* Choose an initial trust region radius Δ_0, and initial iterate x_0. Choose compatibility check parameters $\kappa_\Delta \in (0, 1)$, $\kappa_\mu > 0$, and $\mu \in (0, 1)$. Choose criticality scaling parameter $\xi > 0$ and update parameter $\omega \in (0, 1)$. Select trust region update parameters $0 < \gamma_c < 1 \leq \gamma_e$. Choose switching condition parameters $\gamma_s > \frac{1}{1+\mu}$ and $\kappa_\theta \in (0, 1)$. Initialize the filter $\mathcal{F}_0 = \emptyset$ and select positive filter parameters γ_f and γ_θ. Select ratio test thresholds $0 < \eta_1 < \eta_2 < 1$. Evaluate $d(x_0)$ and then calculate $\theta(x_0)$. Set $k = 0$.
2. Generate a reduced model $r_k(x)$ that is κ-fully linear on $B(x_k, \Delta_k)$.
3. Solve the compatibility check (4). If $TRSP_k$ is compatible as given by Definition 2, go to Step 4. Otherwise, add (θ_k, f_k) to the filter and go to Step 10.
4. Compute criticality measure χ_k using (5). If the criticality check $\chi_k < \xi \Delta_k$ holds, reassign $\Delta_k := \omega \Delta_k$ and go to Step 2. Else, continue to Step 5.
5. Solve the subproblem (2) to compute a step s_k.
6. *Filter:* Evaluate $\theta(x_k + s_k)$. If the step is acceptable to the filter as given in (6), continue to Step 7. Else, set $x_{k+1} = x_k$, $\theta_{k+1} = \theta_k$, and $\Delta_{k+1} = \gamma_c \Delta_k$, then set $k := k + 1$ and go to Step 2.
7. *Switching condition:* If the switching condition (7) is true, go to Step 8. Else, go to Step 9.
8. *f-type step:* Accept trial step and increase trust region. Set $x_{k+1} = x_k + s_k$, $\Delta_{k+1} = \gamma_e \Delta_k$, and $\theta_{k+1} = \theta(x_k + s_k)$. Set $k := k + 1$ and go to Step 2.
9. *θ-type step:* Add (θ_k, f_k) to the filter and accept trial step $x_{k+1} = x_k + s_k$. Update the trust region using (8) and (9). Set $k := k + 1$ and go to Step 2.
10. *Restoration:* Call restoration algorithm to compute a point x_{k+1}, trust region radius Δ_{k+1}, and reduced model r_{k+1} such that x_{k+1} is acceptable to the filter $\mathcal{F}_k \cup (\theta_k, f_k)$ and TRSP_{k+1} is compatible ($\beta_i = 0$). Set $k := k + 1$ and go to Step 2.

Proposed Modifications

Algorithm was applied to several chemical process optimization problems including an ammonia synthesis process and a power generation process.[30] However, in gaining experience with the method, it was observed that the algorithm took very small steps when moderately close to an optimum. To address this problem, we will introduce the concept of a sampling region. In addition, several algorithmic modifications that help

numerical performance will be discussed; these are motivated by observation of rejected steps at low values of θ.

Sampling region

In trust region-based derivative free methods, the trust region simultaneously serves two purposes: reduced model error management and step size control (globalization). In UOBYQA,[36] a solver for unconstrained DFO, Powell separated these two tasks through the concept of a sampling region. This concept is adapted for the TRF method where the glass box/black box structure makes it especially effective.

We define the sampling region for iteration k as the closed ball $B(x_k, \sigma_k)$, where σ_k represents the sampling region radius. The sampling region is contained within the trust region, that is, $\sigma_k \le \Delta_k$ for all iterations k. The algorithm relies on the κ-fully linear property only for $B(x_k, \sigma_k)$ to capture local behavior. Note that while the reduced model is required to be κ-fully linear on the sampling region, this does not necessarily require that all samples need to be contained within this region.

The benefit of the sampling region is twofold. While Algorithm requires $\Delta_k \to 0$, we now only require $\sigma_k \to 0$. This will allow for larger trust region steps near the solution, which is particularly important as the glass box variables z are also controlled by the trust region for globalization purposes. Second, with the sampling region radius $\sigma_k < \Delta_k$, it may not be necessary to rebuild the reduced model if the step is rejected. Therefore, the trust region can adjust for globalization purposes, and unless the new trust region radius is set below the current sampling region radius, the original reduced model may be re-used.

Update of Sampling Region Radius. If the trust region radius would be set to some value less than the current sampling radius, then the sampling region is shrunk so that it is again contained within the trust region. The original algorithm shrinks the trust region by a constant factor at each iteration that the criticality phase is invoked. When using a sampling region, the criticality phase shrinks the sampling region instead of the trust region. In addition, computational experience showed that reducing the sampling region by a constant factor may be too conservative. The criticality test $\chi_k < \xi\Delta_k$ changes to $\chi_k < \xi\sigma_k$. If this condition is true, then σ_k is reduced to a value where the criticality test no longer holds: $\sigma_k = \max(\min(\sigma_{k-1}, \chi_k/\xi), \Delta_{\min})$.

Step size update

The basic trust region update formula (9) increases or decreases the trust region radius by constant factors. However, it may be better to base the update on the norm of the step computed at iteration k.[37]

As shown in Appendix A, the following two requirements are all that are needed to ensure the global convergence proof holds. First, the trust region must not decrease when taking an f-type step. Next, the trust region must decrease when a step is rejected. Outside of this, we have freedom to define an update procedure as desired, using computational experiments as a guide. The following scheme has been effective in practice.

In previous computational experiments, one problem that emerged was a sequence of f-type steps with small norm $\|s_k\|$. This allowed the trust region to become very large during convergence, in turn causing a prolonged criticality phase to certify optimality. For f-type steps, the trust region is now updated as follows: $\Delta_{k+1} := \max[\gamma_e\|s_k\|, \Delta_k]$, where $\gamma_e \ge 1$ and $s_k = \bar{x}_k - x_k$. This means that if the current trust region constraint is (nearly) active, the trust region should be expanded to allow faster progress. However, if the step at this iteration is small, we maintain the current value of Δ_k as a decrease in the radius is not allowed.

For θ-type steps, the convergence proof has no restrictions on the trust region update (provided $\Delta_k \in \mathbb{R}^+$). Therefore, we propose the following modification.

$$\Delta_{k+1} = \begin{cases} \gamma_c\|s_k\| & \text{if } \rho_k < \eta_1, \\ \Delta_k & \text{if } \eta_1 \le \rho_k < \eta_2, \\ \max\{\gamma_e\|s_k\|, \Delta_k\} & \text{if } \rho_k \ge \eta_2 \end{cases} \tag{10}$$

where ρ_k is given by

$$\rho_k = \frac{\theta(x_k) - \theta(x_k + s_k) + \epsilon_\theta}{\max(\|y_k - r_k(w_k)\|, \epsilon_\theta)} \tag{11}$$

and ϵ_θ is a small tolerance. This new definition of ρ is derived from the standard trust region definition of actual reduction divided by predicted reduction (8), modified to prevent the trust region from decreasing when both θ_k and $\theta(x_k + x_k)$ are very small.

When a step is rejected, the following update is used: $\Delta_{k+1} = \gamma_c\|s_k\|$. This eliminates the possibility that $s_{k+1} = s_k$, which may occur when the reduced model r_{k+1} is nearly or exactly the same as the one used at iteration k.

Changes to filter mechanism

In this section, we discuss additional modifications to the filter mechanism introduced above. We say that a step s_k is acceptable to the filter if:

$$\theta(x_k + s_k) \le (1 - \gamma_\theta)\theta_j \quad \text{or} \quad f(x_k + s_k) \le f_j - \gamma_f\theta_j \tag{12}$$

for all $(\theta_j, f_j) \in \mathcal{F}_k$. In previous TRF methods,[30, 31] the following additional requirement is enforced for all acceptable steps:

$$\theta(x_k + s_k) \le (1 - \gamma_\theta)\theta_k \quad \text{or} \quad f(x_k + s_k) \le f(x_k) - \gamma_f\theta_k \tag{13}$$

By contrast, the line search filter method used in IPOPT[6] only requires (13) for θ-type steps. Because imposition of (13) is a tighter condition on the acceptance of s_k, it may lead to more frequent shrinkage of the trust region and calls to the restoration phase at the current iterate. Conversely, if (13) is not imposed, an unsuitable s_k would be allowed that may impede performance at subsequent iterations. We observed this trade-off in our computational results, and found that taking a "bad" step that doesn't satisfy (13) may help prevent restoration rather than decreasing the trust region directly. Also, note that only condition (12) is required to guarantee convergence.

We also borrow the following requirement from IPOPT[6] that restricts f-type steps to small values of θ. The switching condition (7) is modified to also require that $\theta(x_k) \le \theta_{\min}$. This places more emphasis on reducing infeasibility in the earlier stages of the algorithm.

Modified TRF algorithm

The TRF method with the above modifications is presented below. Unlike the presentation in Algorithm 1, termination parameters are also included here.

Algorithm 2.

1. Initialization: Choose an initial trust region radius Δ_0, an initial iterate x_0, and an initial sampling radius σ_0. Choose

termination tolerances ϵ_θ (for feasibility), ϵ_χ (for criticality), and $\epsilon_\Delta \geq \Delta_{\min}$ for ensuring accurate reduced models and preventing numerical issues with small trust regions, respectively. Choose compatibility check parameters $\kappa_\Delta \in (0, 1)$, $\kappa_\mu > 0$, $\mu \in (0, 1)$, and $\epsilon_{\text{compat}} > 0$. Choose criticality scaling parameter $\xi > 0$. Select trust region update parameters $0 < \gamma_c < 1 \leq \gamma_e$. Choose switching condition parameters $\gamma_s > \frac{1}{1+\mu}$, $\kappa_\theta \in (0,1)$, and $\theta_{\min} > 0$. Initialize the filter $\mathcal{F}_0 = \emptyset$ and select positive filter parameters γ_f, γ_θ, and $\theta_{\max}$. Select ratio test thresholds $0 < \eta_1 < \eta_2 < 1$ and sampling region reset parameter $\psi \in (0, 1]$. Evaluate $d(x_0)$ and then calculate $\theta(x_0)$. Set $k = 0$.

2. Generate a reduced model $r_k(w)$ that is κ-fully linear on $B(x_k, \sigma_k)$. Reuse previous reduced model if possible.

3. *Criticality and termination check:* Calculate criticality measure χ_k (5).

   a. If $\theta_k \leq \epsilon_\theta$, $\chi_k \leq \epsilon_\chi$, and $\sigma_k \leq \epsilon_\Delta$, STOP. A first-order critical point has been found to tolerance.
   b. If $\Delta_k \leq \Delta_{\min}$, $\Delta_{k-1} \leq \Delta_{\min}$, $\theta_k \leq \epsilon_\theta$, and $\theta_{k-1} \leq \epsilon_\theta$, STOP. A feasible solution has been found but progress to optimality is too slow.
   c. *Criticality phase:* If $\chi_k < \xi\sigma_k$, then set $\sigma_k = \max(\min(\sigma_{k-1}, \chi_k/\xi), \Delta_{\min})$

4. *Compatibility Check:* Solve the compatibility check (4) to obtain compatibility measure β. If $\beta < \epsilon_{\text{compat}}$ (i.e., TRSP_k is compatible), go to Step 5. Otherwise, add (θ_k, f_k) to the filter and go to Step 10.

5. Solve subproblem (2) and obtain the solution $\bar{x}$. Define the step $s_k := \bar{x} - x_k$.

6. *Filter:* Evaluate $\theta(x_k + s_k)$. If for all $(\theta_j, f_j) \in \mathcal{F}_k$, $\theta(x_k + s_k) \leq (1 - \gamma_\theta)\theta_j$ or $f(x_k + s_k) \leq f_j - \gamma_f\theta_j$ then the step s_k is acceptable to the filter. If the step is acceptable to the filter, continue to Step 7. Else, set $x_{k+1} = x_k$, $\theta_{k+1} = \theta_k$, $\Delta_{k+1} = \gamma_c\|s_k\|$, and $\sigma_{k+1} = \min\{\sigma_k, \psi\Delta_{k+1}\}$. Then set $k := k + 1$ and go to Step 2.

7. *Switching condition:* If the switching condition $f(x_k) - f(x_k + s_k) \geq \kappa_\theta\theta(x_k)^{\gamma_s}$ and $\theta \leq \theta_{\min}$ go to Step 8. Else, go to Step 9.

8. *f-type step:* Accept trial step and possibly increase trust region: Set $x_{k+1} = x_k + s_k$, $\Delta_{k+1} = \max\{\gamma_e\|s_k\|, \Delta_k\}$, $\sigma_{k+1} = \sigma_k$ and $\theta_{k+1} = \theta(x_k + s_k)$. Set $k := k + 1$ and go to Step 2.

9. *θ-type step:* Add (θ_k, f_k) to the filter and accept trial step $x_{k+1} = x_k + s_k$. Update the trust region using (10) and (11). Set $\sigma_{k+1} = \min[\sigma_k, \psi\Delta_{k+1}]$. Set $k := k + 1$ and go to Step 2.

10. *Restoration phase:* Return a new point x_{k+1}, trust region radius Δ_{k+1}, sample region radius σ_{k+1} and new reduced model $r_{k+1}(w)$ such that $TRSP_{k+1}$ is compatible. Set $k := k + 1$ and go to step 2.

Implementation

The TRF method has been implemented in Python/Pyomo. Pyomo (Python Optimization Modeling Objects) is an open source framework for modeling and analyzing mathematical programming problems.[38, 39] Pyomo includes interfaces to many of the most popular mathematical programming solvers and is especially well suited to the TRF method, where models are repeatedly solved with small modifications. A user has freedom to select a particular solver for the subproblem; by default all subproblems are solved with IPOPT 3.12.4.[6] The criticality check (5) requires the derivatives of problem (2) evaluated at a point; these are obtained using the AMPL pseudo-solver *gjh*.

The current implementation of the TRF method includes methods for building linear and quadratic interpolation models. While more complex models are permitted within the κ-fully linear framework, there is a trade-off between model complexity and sampling requirements. Quadratic interpolation requires $\frac{1}{2}(m+1)(m+2)$ sample points in m-dimensional space. "Higher-order" interpolation strategies would likely require more sample points. However, computational experiments suggest that it is better to build a simpler reduced model and quickly proceed to the next trust region subproblem, rather than evaluate many samples finding the best possible fit. More discussion of this can be found in our previous work.[40] With linear interpolation models, we use a forward difference perturbation with magnitude equal to the sampling radius. When using quadratic interpolation, the method uses a fixed, *a priori* calculated well-poised sample geometry at every iteration. In typical derivative free codes, the geometry is allowed to adapt, reusing previous points when possible. Our results with fixed, well-poised sample geometry provide an upper bound on computational expense to a more sophisticated strategy that reuses interpolation points. Moreover, our initial experiments show that the use of a sampling region greatly decreases the chance of interpolation points being reusable.

One of the advantages of the TRF method is the flexibility for using many types of reduced models. If particular problem structure or information is known, a user may define her own function to construct a reduced model using that information. This can easily be added directly to the code without significant changes. The TRF method assumes that the result will satisfy the κ-fully linear property.

A list of the TRF method's tunable parameters may be found in Appendix C (see Supplemental Information), along with suggested values. These suggested values were based on a small amount of trial and error, as well as suggestions from previous literature.[37] The most sensitive parameters are the initial trust region radius, initial sample region radius, and to a lesser extent, termination tolerances. For the numerical results presented below, all parameters were kept at default values as shown in Appendix C unless otherwise noted. The initial trust region radius Δ_0 was tuned for each problem (but remains the same for all algorithms).

Numerical Results

To test the effectiveness of proposed modifications to the TRF algorithm, a set of test problems was assembled. There is no known standard library of test problems for glass box/black box optimization, so a test set has been developed from modified versions of established NLP benchmarks from the CUTEr and COPS sets.[41, 42] A description of the test set and how it was constructed is given in Appendix B (see Supplemental Information).

The test set is used to compare Algorithm 1 to Algorithm 2 with the modifications described above. Both the original algorithm and the newly modified version are run using both linear and quadratic reduced models for a total of four algorithmic variants. Solutions of the CUTEr problems are verified against a fully glass box formulation to confirm that they are first-order KKT points.

For the algorithms that do not use a sampling region, very few problems are solvable at default tolerance level. The algorithms approach optimality but then proceed to take a sequence of very small *f*-type steps. These steps all remain successful but the rate of convergence is too slow to reach the desired tolerance after 10,000 black box function evaluations. For the results for Algorithms 1 and 2, the tolerance ϵ_χ is

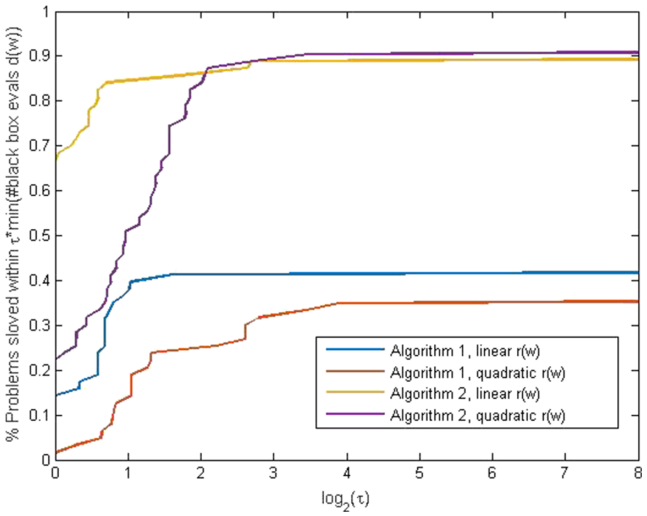

Figure 2. Performance profile for black box calls.
[Color figure can be viewed at wileyonlinelibrary.com]

relaxed to 10^{-3} to recognize that the solver had nearly identified an optimal solution before being trapped in the slow convergence pattern.

The performance of each of the four methods (Algorithm 1 with linear/quadratic reduced models, and Algorithm 2 with linear/quadratic reduced models) is presented in Appendix B. For all test problems, at least one of the four methods was able to solve the problem in under the 10,000 function evaluation limit. These results are summarized using Dolan-Moré performance profiles in Figures 2 and 3.

The results indicate that the recent improvements to the algorithm have successfully improved the performance on this test set. Algorithm 2 is significantly more efficient (fewer function evaluations) and more robust (more problems solved) than Algorithm 1 with both linear and quadratic reduced models. From Figure 3, it is clear that quadratic reduced models require fewer iterations with Algorithm . The number of iterations acts as a proxy for the computational expense of solving the subproblems. If black box evaluations are relatively cheap then considering the number of iterations becomes important. Algorithm 2 with quadratic reduced models is the clear choice in this case. However, Algorithm 2

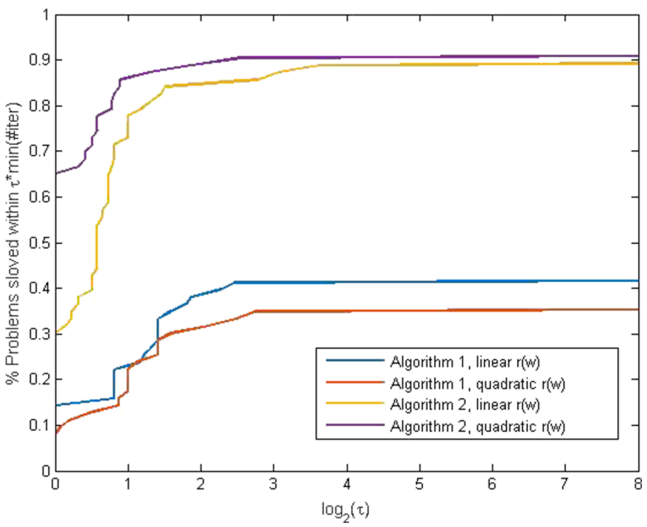

Figure 3. Performance profile for iterations.
[Color figure can be viewed at wileyonlinelibrary.com]

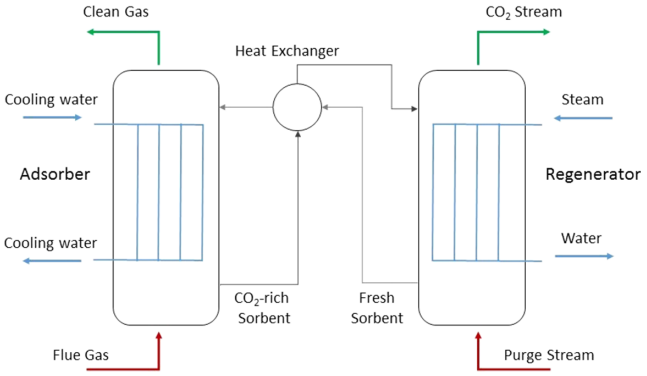

Figure 4. Carbon capture system.
Regenerator column is treated as black box. [Color figure can be viewed at wileyonlinelibrary.com]

with linear models generally requires fewer black box evaluations as shown on Figure 2, because fewer samples are needed to construct the reduced model. In terms of robustness, Algorithm 2 with linear and quadratic models are roughly the same, with quadratic models slightly more robust. The quadratic models are able to solve some highly nonlinear problems that may fail with linear reduced models. We note that all four methods were initialized with the same starting points and initial trust region radius Δ_0. Moreover, we observed that Algorithm 2 was more robust to different initializations than Algorithm 1. Nevertheless, there is still some room for improvement, as no single approach can solve all problems.

Carbon capture system

To demonstrate the improved performance of Algorithm 2 on a large-scale real-world problem, we present an example from chemical process optimization. This case study considers carbon capture from a flue gas stream using solid sorbent. The system consists of two bubbling fluidized beds as shown in Figure 4. The first bubbling fluidized bed is the adsorber that removes CO_2 by contacting solid sorbent with flue gas from a power plant.

The second column is the regenerator, which is used to remove the CO_2 from the solid sorbent. The CO_2 can then be sent for sequestration or utilization. The adsorber is cooled with cooling water to promote CO_2 adsorption, while the regenerator is heated with steam to promote desorption. The sorbent is transported between the two columns with a heat exchanger to preheat (precool) the sorbent before entering the regenerator (adsorber).

Both the adsorber and regenerator are modeled as bubbling fluidized bed reactors. The model was developed by NETL[43, 44] to predict axial and temporal variations in concentrations and temperatures. For our case study, the adsorber is modeled as a fully equation-oriented glass box, while the regenerator is considered to be the black box. In the adsorber model, solid sorbent is added at the top of the column, while flue gas is added at the bottom. The amine adsorption reactions occur in the solid phase:

$$\begin{aligned} &H_2O_{(g)} \rightleftharpoons H_2O_{(phys)} \\ &2R_2NH + CO_2 \rightleftharpoons R_2NH_2^+ + R_2NCO_2^- \\ &R_2NH + CO_2 + H_2O_{(phys)} \rightleftharpoons R_2NH_2^+ + HCO_3^- \end{aligned} \tag{14}$$

where R is an organic functional unit. Flow within the column is modeled using three flow regimes: a downward flowing emulsion region (containing solids), upward flowing bubble region (containing no solids), and upward flowing cloud wake

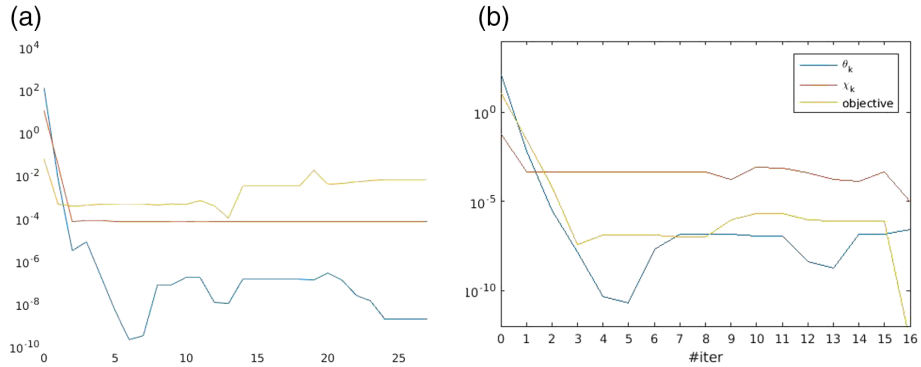

Figure 5. Convergence metrics.

θ_k and χ_k must be below tolerance to converge. *Objective* is absolute value deviation from optimal objective function value. (a) Algorithm 1, (b) Algorithm 2. [Color figure can be viewed at wileyonlinelibrary.com]

region (containing some solids). Heat and mass transfer among these regions are described by a set of 20 partial differential equations. The model also contains algebraic equations that include empirical correlations to describe the hydrodynamics, heat and mass transfer coefficient relations, gas phase properties including viscosity, thermal conductivity, heat capacity, empirical correlations to describe the cooling tubes within the adsorber, and detailed nonlinear reaction kinetics for reactions (14). In this work, the model is considered in steady-state mode (all time derivatives are set to zero). The resulting differential algebraic equations are discretized (in space) using collocation on finite elements. Full details of the model can be found in previous publications.[43-45]

The structure of the regenerator model is the same except with different boundary conditions described in Modekurti et al.[44] This regenerator model is solved independently as a black box and the adsorber/regenerator system creates a chemical process where glass and black box portions interact through the cycling of the solid sorbent. The model is used for minimization of utility usage subject to a 48% CO_2 capture constraint. The size of the problem is given as follows: black box inputs $n_w = 6$, black box outputs $n_y = 5$ and other variables $n_z = 5144$. The reduced models for the regenerator were constructed by linear interpolation.

Algorithm 1 was applied to the carbon capture system, and terminated after 26 iterations, because progress to the solution was too slow (step size below tolerance of 10^{-6} for two consecutive iterations). In this run, $\Delta_k \to 0$ but with the final value of $\chi = 6.39 \times 10^{-3}$. When Algorithm 2 was applied to the carbon capture system, the optimal solution was determined in 16 iterations at essentially the same point (with objective function at 297.632) as with Algorithm 1. At the last iteration $\chi \leq 10^{-5}$, $\sigma_k \to 0$ and Δ_k does not need to shrink to zero and remains at about 10^{-4}. More information on the performance of the two algorithms can be found in Figures 5 and 6. Figure 5 focuses on the NLP convergence metrics, while Figure 6 focuses on the sequence of step sizes. In both figures, the abscissa represents the iteration count and the ordinate is the value for the metrics. All values are shown in log scale to provide a better sense of the convergence behavior.

In Figure 5, we plot infeasibility measure θ_k, the criticality measure χ_k as defined in (5) and $|f(x_k) - f(x^*)|$ as a measure of the convergence of the objective function. Both algorithms converge to feasibility relatively quickly, but Algorithm 1 slows down significantly and does not make much progress toward the optimal solution. The iterates stay within feasibility tolerance $\epsilon_\theta = 10^{-6}$ for most iterations.

In Figure 6, the trust region radius Δ_k, sample region radius σ_k and the step size $\|s_k\|$ are plotted to show the interactions. It is clear that in Figure 6b, the sample radius successfully controlled the accuracy of the reduced model while trust region allows larger step sizes. In many iterations, step size $\|s_k\| < \Delta_k$ is much larger than the sample radius σ_k. The trust region constraint is only active at iteration 8. This is because the algorithm takes mostly *f*-type steps (when $\theta(x)$ is small, *f*-type steps only require small decrease in the objective function). As the trust region radius does not shrink on *f*-type steps, progress continues without updating the trust region. When the trust region is updated, for example at iteration 8, the trust region update is based on the step size. This results

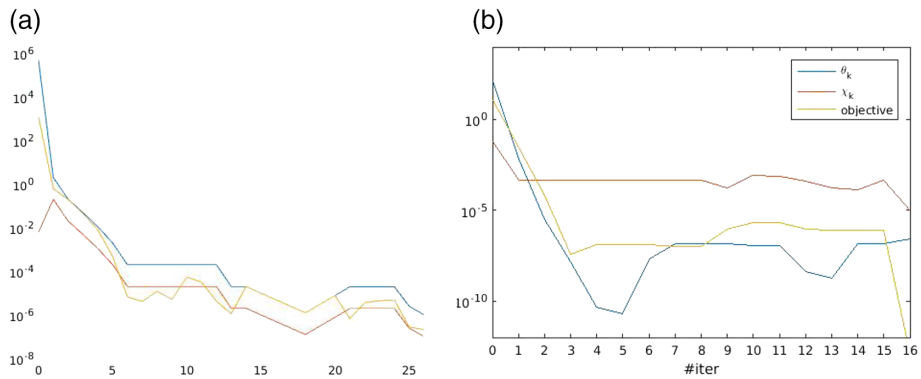

Figure 6. Trust and sample region radius shown together with step length $\|s_k\|$ across iterations.

Sample region radius in Algorithm represents the perturbation size used to construct reduced models. (a) Algorithm 1, (b) Algorithm 2. [Color figure can be viewed at wileyonlinelibrary.com]

in a large change in the trust region radius in one iteration. The trust region update also requires the sample region to adjust so that the sample region lies within the trust region.

In contrast, Figure 6a shows the step size in Algorithm 1 is restricted by the trust region radius, as the perturbation size used to construct a reduced model, is directly tied to the trust region radius. Hence, the trust region constraint is consistently active in Algorithm 2, resulting in small step sizes and very slow convergence, as shown in Figure 5a for the objective function and criticality measure χ_k.

Conclusions

The optimization of hybrid glass box/black box systems is an important yet challenging problem for large-scale process systems. A TRF method for solving these problems was presented, along with recent modifications to this approach to improve performance. Chief among these was the introduction of a sampling region. By decoupling the two purposes of the trust region, the TRF method is made more robust to challenging problems, and faster on easier problems due to the ability to take longer steps. In addition, a Pyomo-based implementation of the method has been developed, along with a set of 62 test problems. Using this test set as a guide, different versions of the algorithm can be benchmarked. The algorithmic modifications proposed in this article more than doubled the number of problems that are successfully solved.

We also demonstrate the sampling region algorithm on a large-scale engineering example. The algorithm's behavior is examined in detail and the sampling region leads to significant performance improvement. Future directions include further tuning of the algorithm, and consideration of automatic or adaptive choices of functional forms for the reduced models.

Acknowledgments

Many thanks to Mingzhao Yu, David Thierry, and Andrew Lee for their help with the BFB model. This material is based on work supported by the National Science Foundation Graduate Research Fellowship Program under Grant No. DGE-1252522 and by the U.S. Department of Energy, Office of Fossil Energy as part of the Institute for the Design of Advanced Energy Systems (IDAES). This article was prepared as an account of work sponsored by an agency of the United States Government. Neither the United States Government nor any agency thereof, nor any of their employees, makes any warranty, express or implied, or assumes any legal liability or responsibility for the accuracy, completeness, or usefulness of any information, apparatus, product, or process disclosed, or represents that its use would not infringe privately owned rights. Reference herein to any specific commercial product, process, or service by trade name, trademark, manufacturer, or otherwise does not necessarily constitute or imply its endorsement, recommendation, or favoring by the United States Government or any agency thereof. The views and opinions of authors expressed herein do not necessarily state or reflect those of the United States Government or any agency thereof.

Literature Cited

1. Drud A. CONOPT: a GRG code for large sparse dynamic nonlinear optimization problems. *Math Program*. 1985;31(2):153-191.
2. Murtagh BA, Saunders MA. MINOS 5.51 users guide; 1983. https://web.stanford.edu/group/SOL/guides/minos551.pdf.
3. Gill P, Murray W, Saunders M. SNOPT: an SQP algorithm for large-scale constrained optimization. *SIAM Rev*. 2005;47(1):99-131.
4. Byrd RH, Nocedal J, Waltz RA. KNITRO: an integrated package for nonlinear optimization. *Large-scale nonlinear optimization*. Springer, Berlin; 2006:35-59.
5. Vanderbei RJ. LOQO: an interior point code for quadratic programming. *Optim Method Softw*. 1999;11(1–4):451-484.
6. Wächter A, Biegler LT. On the implementation of an interior-point filter line-search algorithm for large-scale nonlinear programming. *Math Program.* 2006;106(1):25-57.
7. Rios LM, Sahinidis NV. Derivative-free optimization: a review of algorithms and comparison of software implementations. *J Global Optim*. 2013;56(3):1247-1293.
8. Gay DM. Hooking your solver to AMPL. Technical Report 93–10, AT&T Bell Laboratories, Murray Hill, NJ; 1993, revised 1997.
9. Powell MJ. A direct search optimization method that models the objective and constraint functions by linear interpolation. *Advances in optimization and numerical analysis*. Springer, Berlin; 1994:51-67.
10. Audet C, Dennis JJ. Mesh adaptive direct search algorithms for constrained optimization. *SIAM J Optim*. 2006;17(1):188-217.
11. Audet C, Le Digabel S, Tribes C. NOMAD user guide. Technical Report G-2009-37, Les cahiers du GERAD; 2009. https://www.gerad.ca/nomad.
12. Sampaio PR, Toint PL. A derivative-free trust-funnel method for equality-constrained nonlinear optimization. *Comput Optim Appl*. 2015;61(1):25-49.
13. Boukouvala F, Misener R, Floudas CA. Global optimization advances in mixed-integer nonlinear programming, MINLP, and constrained derivative-free optimization. *CDFO Europ J Operat Res*. 2016;252(3):701-727.
14. Bhosekar A, Ierapetritou M. Advances in surrogate based modeling, feasibility analysis and optimization: a review. *Comput Chem Eng*. 2017.
15. Barrett A, Walsh J. Improved chemical process simulation using local thermodynamic approximations. *Comput Chem Eng*. 1979;3(1–4):397-402.
16. Boston J, Britt H. A radically different formulation and solution of the single-stage flash problem. *Comput Chem Eng*. 1978;2(2–3):109-122.
17. Leesley M, Heyen G. The dynamic approximation method of handling vapor-liquid equilibrium data in computer calculations for chemical processes. *Comput Chem Eng*. 1977;1(2):103-108.
18. Henao CA, Maravelias CT. Surrogate-based superstructure optimization framework. *AIChE J*. 2011;57(5):1216-1232.
19. Caballero JA, Grossmann IE. An algorithm for the use of surrogate models in modular flowsheet optimization. *AIChE J*. 2008;54(10):2633-2650.
20. Boukouvala F, Floudas CA. ARGONAUT: AlgoRithms for global optimization of coNstrAined grey-box compUTational problems. *Optim Lett*. 2017;11(5):895-913.
21. Boukouvala F, Hasan MF, Floudas CA. Global optimization of general constrained grey-box models: new method and its application to constrained PDEs for pressure swing adsorption. *J Global Optim.* 2017;67(1–2):3-42.
22. Cozad A, Sahinidis NV, Miller DC. Learning surrogate models for simulation-based optimization. *AIChE J*. 2014;60(6):2211-2227.
23. Wilson ZT, Sahinidis NV. The ALAMO approach to machine learning. *Comput Chem Eng*. 2017;106:785-795.
24. Alexandrov NM, Dennis JE Jr, Lewis RM, Torczon V. A trust-region framework for managing the use of approximation models in optimization. *Struct Optim*. 1998;15(1):16-23.
25. Arian E, Fahl M, Sachs EW. Trust-region proper orthogonal decomposition for flow control. Technical Report ICASE Report No. 2000-25, Institute for Computer Applications in Science and Engineering; 2000.
26. Giunta AA, Eldred MS. Implementation of a trust region model management strategy in the DAKOTA optimization toolkit. In: Proceedings of the 8th AIAA/USAF/NASA/ISSMO Symposium on Multidisciplinary Analysis and Optimization, Long Beach, CA; 2000.
27. Wild SM, Regis RG, Shoemaker CA. ORBIT: optimization by radial basis function interpolation in trust-regions. *SIAM J Sci Comput*. 2008;30(6):3197-3219.
28. Agarwal A, Biegler LT. A trust-region framework for constrained optimization using reduced order modeling. *Optim Eng*. 2013;14(1):3-35.
29. March A, Willcox K. Constrained multifidelity optimization using model calibration. *Struct Multidiscip Optim.* 2012;46(1):93-109.
30. Eason JP, Biegler LT. A trust region filter method for glass box/black box optimization. *AIChE J*. 2016;62(9):3124-3136.

31. Fletcher R, Gould NI, Leyffer S, Toint PL, Wächter A. Global convergence of a trust-region SQP-filter algorithm for general nonlinear programming. *SIAM J Optim.* 2002;13(3):635-659.
32. Conn AR, Scheinberg K, Vicente LN. *Introduction to Derivative-Free Optimization*. Philadelphia, PA: SIAM; 2009.
33. Wild SM. Derivative-free optimization algorithms for computationally expensive functions. Ph.D. thesis, Cornell University; 2009.
34. Fletcher R, Leyffer S. Nonlinear programming without a penalty function. *Math Program*. 2002;91(2):239-269.
35. Wächter A, Biegler LT. Line search filter methods for nonlinear programming: motivation and global convergence. *SIAM J Optim*. 2005;16(1): 1-31.
36. Powell MJ. UOBYQA: unconstrained optimization by quadratic approximation. *Math Program.* 2002;92(3):555-582.
37. Conn AR, Gould NI, Toint PL. *Trust Region Methods*. Philadelphia, PA: SIAM; 2000.
38. Hart WE, Watson JP, Woodruff DL. Pyomo: modeling and solving mathematical programs in Python. *Math Program Comput*. 2011;3(3): 219-260.
39. Hart WE, Laird C, Watson JP, Woodruff DL. *Pyomo–optimization modeling in Python*. Springer Science & Business Media, Berlin; 2012.
40. Eason JP, Biegler LT. Reduced model trust region methods for embedding complex simulations in optimization problems. *Computer Aided Chemical Engineering*. Vol 37. Elsevier; 2015:773-778.
41. Gould NI, Orban D, Toint PL. CUTEr and SifDec: a constrained and unconstrained testing environment, revisited. *ACM Trans Math Softw*. 2003;29(4):373-394.
42. Dolan ED, Moré JJ, Munson TS. Benchmarking optimization software with COPS 3.0. Technical Report, Argonne National Lab., Argonne, IL; 2004.
43. Lee A, Miller DC. A one-dimensional (1-d) three-region model for a bubbling fluidized-bed adsorber. *Ind Eng Chem Res*. 2012;52(1): 469-484.
44. Modekurti S, Bhattacharyya D, Zitney SE. Dynamic modeling and control studies of a two-stage bubbling fluidized bed adsorber-reactor for solid–sorbent CO_2 capture. *Ind Eng Chem Res.* 2013;52(30): 10250-10260.
45. Yu M, Miller DC, Biegler LT. Dynamic reduced order models for simulating bubbling fluidized bed adsorbers. *Ind Eng Chem Res.* 2015; 54(27):6959-6974.

Manuscript received Feb. 22, 2018, and revision received Jul. 12, 2018.