Noname manuscript No.
(will be inserted by the editor)

A Matrix-Free Trust-Region Newton Algorithm for Convex-Constrained Optimization

D. P. Kouri

Received: date / Accepted: date

Abstract We describe a matrix-free trust-region algorithm for solving convex-constrained optimization problems that uses the spectral projected gradient method to compute trial steps. To project onto the intersection of the feasible set and the trust region, we reformulate and solve the dual projection problem as a one-dimensional root finding problem. We demonstrate our algorithm's performance on multiple problems from data science and PDE-constrained optimization. Our algorithm shows superior performance when compared with five existing trust-region and spectral projected gradient methods, and has the added benefit that it is simple to implement.

Keywords Nonconvex Optimization, Convex Constraints, Trust Regions, Spectral Projected Gradient, Large-Scale Optimization, Newton's Method

Mathematics Subject Classification (2010) 49M15, 49M37, 65K05, 65K10, 90C06, 90C30

1 Introduction

We develop a matrix-free trust-region Newton method for the efficient numerical solution of the optimization problem

$$\min_{x \in H} f(x) \qquad \text{subject to} \qquad x \in \mathcal{C}, \tag{1}$$

This research was sponsored by the U.S. Air Force Office of Scientific Research, Optimization Program under Award NO: F4FGA09135G001.

D. P. Kouri
Optimization and Uncertainty Quantification, Sandia National Laboratories
P.O. Box 5800, MS-1320, Albuquerque, NM 87185-1320
E-mail: dpkouri@sandia.gov
Sandia National Laboratories is a multimission laboratory managed and operated by National Technology and Engineering Solutions of Sandia, LLC., a wholly owned subsidiary of Honeywell International, Inc., for the U.S. Department of Energys National Nuclear Security Administration under contract DE-NA0003525. This paper describes objective technical results and analysis. Any subjective views or opinions that might be expressed in the paper do not necessarily represent the views of the U.S. Department of Energy or the United States Government.

where H is a Hilbert space, $f : H \to \mathbb{R}$ is a smooth function, and $\mathcal{C} \subseteq H$ is a nonempty, closed and convex set. Many applications in science and engineering can be formulated as the optimization problem (1), including optimal control, inverse and design problems (see, e.g., [2,4,31]) as well as statistical learning problems (see, e.g., [32]). In these applications, the objective function and its derivatives are often expensive to evaluate, emphasizing the need for rapidly converging Newton-type methods that do not require factorizations of, e.g., the Hessian matrix.

Motivated by the unconstrained spectral projected gradient (SPG) trust-region method developed in [24] and the SPG quasi-Newton line search method developed in [29], our proposed algorithm approximates a solution to the quadratic trust-region subproblem using SPG [5,7]. In doing so, the computed step maintains feasibility. This approach is simple to implement as it does not require, e.g., explicit treatment of the active set [1,23] or projections onto constraint null spaces [23]. For comparison, the trust-region Newton method proposed in [23] applies only to polyhedral feasible sets and employs a projected truncated conjugate gradient (CG) algorithm, accounting for the inactive constraints, to approximately solve the trust-region subproblem. The algorithm then performs a projected search to restore feasibility and ensure sufficient decrease of the trial step. Aside from the complicated machinery required to implement this algorithm, the CG solver may terminate early due to negative curvature, resulting in poor steps for nonconvex problems. In contrast, our proposed method is not affected by negative curvature and often produces successful trial steps.

A key requirement for our approach is the ability to efficiently project onto the feasible set $\mathcal{C}$. Using this projection, we develop an algorithm to project onto the intersection of $\mathcal{C}$ and the trust region. To do this, we reformulate the dual problem associated with this projection as a one-dimensional root-finding problem, which can be solved using any standard root-finding algorithm. Based on numerical experience, we employ Brent's method [10] to solve the dual problem. Basic secant-type methods, such as the secant and regula falsi methods, suffer from short step sizes due to the nature of the dual function. Brent's method produces steps that are no worse than bisection, but often exhibits rapid local convergence similar to the secant method.

The paper is organized as follows. In Section 2, we describe our proposed algorithm, which is an instance of the basic trust-region algorithm. Consequently, the convergence of the algorithm is guaranteed in Hilbert space by the analysis in [30]. In Section 3, we provide numerical comparisons of our method with five existing trust-region and SPG-based methods on problems from data science and partial differential equation (PDE) constrained optimization.

Notation. We denote the inner product on H by $\langle \cdot, \cdot \rangle$ and the usual norm by $\| \cdot \|$. We denote the projection of $x \in H$ onto a set $\mathcal{S} \subseteq H$ by $\mathbf{P}_{\mathcal{S}}(x)$ and recall that

$$\|\mathbf{P}_{\mathcal{S}}(x) - x\| \leqslant \|y - x\| \quad \forall\, y \in \mathcal{S}. \tag{2}$$

We denote the gradient of a Fréchet differentiable function $g : H \to \mathbb{R}$ at $x \in H$ by $\nabla g(x) \in H$ and recall that $\nabla g(x)$ is the Riesz representer of the Fréchet derivative $g'(x) \in H^*$, where H^* is the topological dual space associated with H. Finally, we denote the space of bounded linear maps from H into itself by $\mathcal{L}(H)$.

To ensure convergence of our proposed algorithm, we assume that $f : H \to \mathbb{R}$ is continuously Fréchet differentiable with Lipschitz continuous gradient on an open

set containing $\mathcal{C}$. We further assume that there exists $\gamma \in \mathbb{R}$ such that the level set $L_\gamma := \{x \in \mathcal{C} \,|\, f(x) \leqslant \gamma\}$ is bounded. This assumption allows us to replace $\mathcal{C}$ with the closed convex hull of L_γ (which is a subset of $\mathcal{C}$). Although this substitution may not be practical, it ensures that $\mathcal{C}$ is bounded as required in [30].

2 Trust-Region Algorithm

The trust-region convergence theory for the convex-constrained optimization problem (1) was established in [30] (see also, e.g., [11,13,14] for finite-dimensional analysis). At each iteration, the trust-region algorithm approximately solves the subproblem

$$\min_{x \in H} \; m_k(x) \qquad \text{subject to} \qquad x \in \mathcal{C}, \;\; \|x - x_k\| \leqslant \Delta_k, \tag{3}$$

where $m_k : H \to \mathbb{R}$ is a model of the objective function, $x_k \in \mathcal{C}$ is the k^{th} iterate, and $\Delta_k > 0$ is the trust-region radius. We require that the model m_k matches both the objective function value and gradient at the iterate x_k, i.e., $m_k(x_k) = f(x_k)$ and $\nabla m_k(x_k) = \nabla f(x_k)$. A typical choice for m_k is the quadratic model

$$m_k(x) = \frac{1}{2}\langle B_k(x - x_k), (x - x_k)\rangle + \langle \nabla f(x_k), (x - x_k)\rangle + f(x_k), \tag{4}$$

where $B_k \in \mathcal{L}(H)$ is some approximation to the Hessian $\nabla^2 f(x_k)$. If $B_k = \nabla^2 f(x_k)$, we recover a globalized Newton's method. To simplify the presentation, we denote the model gradient by $g_k := \nabla m_k(x_k)$ and the feasible arc along the projected-gradient path by

$$d_k(t) := \mathbf{P}_{\mathcal{C}}(x_k - t g_k) - x_k.$$

To approximately solve (3), we first compute a generalized Cauchy point (GCP). At the k^{th} iteration, the GCP has the form

$$s_k^{\text{GCP}} := d_k(t_k) \qquad \text{and} \qquad x_k^{\text{GCP}} := x_k + s_k^{\text{GCP}}$$

for some $t_k > 0$. Following [30], we require that the GCP satisfies both

$$\begin{cases} \|s_k^{\text{GCP}}\| \leqslant \nu_1 \Delta_k \\ m_k(x_k^{\text{GCP}}) - m_k(x_k) \leqslant \mu_1 \langle g_k, s_k^{\text{GCP}} \rangle \end{cases}, \tag{5a}$$

and at least one of the following conditions

$$\begin{cases} t_k \geqslant \nu_2 t_k' \quad \text{with} \quad m_k(x_k + d_k(t_k')) - m_k(x_k) \geqslant \mu_2 \langle g_k, d_k(t_k') \rangle \\ t_k \geqslant \min\{\nu_3 \Delta_k / \|g_k\|, \nu_4\} \end{cases}. \tag{5b}$$

Here, $0 < \mu_1 < \mu_2 < 1$, $0 < \nu_3 < \nu_1$, $0 < \nu_2 \leqslant 1$ and $\nu_4 > 0$. To satisfy these conditions, we perform the bi-directional projected search described in [23]. Once the GCP has been computed, we seek an approximate solution, x_{k+1}, to (3) that satisfies the following three requirements

$$x_{k+1} \in \mathcal{C} \tag{6a}$$

$$\|x_{k+1} - x_k\| \leqslant \nu_5 \Delta_k \tag{6b}$$

$$m_k(x_k) - m_k(x_{k+1}) \geqslant \mu_3 (m_k(x_k) - m_k(x_k^{\text{GCP}})) \tag{6c}$$

for some fixed $\mu_3 \in (0, 1]$ and $\nu_5 \geqslant \nu_1$.

Given the trial iterate x_{k+1}, the trust-region algorithm accepts the iterate if the ratio of actual and predicted reduction,

$$\rho_k := \frac{f(x_k) - f(x_{k+1})}{m_k(x_k) - m_k(x_{k+1})}, \tag{7}$$

satisfies $\rho_k \geqslant \eta_1$ where $\eta_1 > 0$. Otherwise, x_{k+1} is set to x_k. Finally, the trust-region radius Δ_{k+1} is decreased if $\rho_k < \eta_1$ and increased if $\rho_k > \eta_2$ with $\eta_2 \in (\eta_1, 1)$. Algorithm 1 lists the convex-constrained trust-region algorithm.

Algorithm 1 Convex-Constrained Trust Region

Require: An initial guess $x_0 \in \mathcal{C}$, initial trust-region radius $\Delta_0 > 0$, $0 < \eta_1 < \eta_2 < 1$ and $0 < \gamma_1 \leqslant \gamma_2 < 1$

for $k = 1, 2, \ldots$ **do**
  Cauchy Point Computation: Compute $x_k^{\mathrm{GCP}} \in \mathcal{C}$ that satisfies (5)
  Step Computation: Compute $x_{k+1} \in \mathcal{C}$ that approximately solves the trust-region subproblem (3) and satisfies (6)
  Step Acceptance and Radius Update: Compute ρ_k as in (7)
  if $\rho_k < \eta_1$ **then**
    $x_{k+1} \leftarrow x_k$
    $\Delta_{k+1} \in [\gamma_1 \Delta_k, \gamma_2 \Delta_k]$
  else if $\rho_k \in [\eta_1, \eta_2)$ **then**
    $\Delta_{k+1} \in [\gamma_2 \Delta_k, \Delta_k]$
  else
    $\Delta_{k+1} \in [\Delta_k, \infty)$
  end if
end for

Spectral Projected Gradient Subproblem Solver. To compute a trial step x_{k+1} that satisfies (6), we apply the SPG algorithm to approximately minimize the quadratic model (4), using the GCP as the initial guess. To describe this approach, we denote the feasible set for (3) by

$$\mathcal{C}_k := \{x \in \mathcal{C} \mid \|x - x_k\| \leqslant \Delta_k\}$$

and we define the optimality criterion

$$\chi_{k,\ell} := \|\mathbf{P}_{\mathcal{C}_k}(x_{k,\ell} - \nabla m_k(x_{k,\ell})) - x_{k,\ell}\|.$$

The SPG algorithm computes iterates with the form

$$x_{k,\ell+1} = x_{k,\ell} + \alpha_\ell(\mathbf{P}_{\mathcal{C}_k}(x_{k,\ell} - \lambda_\ell \nabla m_k(x_{k,\ell})) - x_{k,\ell}),$$

where $\alpha_\ell > 0$ is traditionally determined by a nonmonotone line search and $\lambda_\ell > 0$ is a safeguarded spectral (Barzilai-Borwein) step length. In our implementation, we exchange the nonmonotone line search for an exact line search, computed by minimizing a one-dimensional quadratic function over $\alpha \in [0, 1]$. We terminate the SPG algorithm if either the iteration limit is exceeded or if the stopping condition

$$\chi_{k,\ell} \leqslant \min\{\epsilon_1, \epsilon_2 \chi_{k,0}\}$$

is satisfied. Here, $\epsilon_1 > 0$ and $\epsilon_2 > 0$ are prescribed absolute and relative tolerances, respectively. Algorithm 2 lists the SPG trust-region subproblem solver. Each it-

Algorithm 2 SPG Trust-Region Subproblem Solver

Require: The initial guess $x_{k,0} = x_k^{\mathrm{GCP}}$, $q_{k,0} = m_k(x_{k,0})$, $d_{k,0} = \nabla m_k(x_{k,0})$, an integer L, and positive constants ϵ_1, ϵ_2, $\lambda_{\min} < \lambda_{\max}$, and $\lambda_0 \in [\lambda_{\min}, \lambda_{\max}]$.

Set $\ell = 0$
while $\ell < L$ **and** $\chi_{k,\ell} > \min\{\epsilon_1, \epsilon_2 \chi_{k,0}\}$ **do**
  Set $s \leftarrow \mathbf{P}_{\mathcal{C}_k}(x_{k,\ell} - \lambda_\ell d_{k,\ell}) - x_{k,\ell}$ and compute $h \leftarrow B_k s$
  Set $\alpha \leftarrow 1$
  if $\langle h, s\rangle > 0$ **then**
    Set $\alpha \leftarrow \min\{1, -\langle d_{k,\ell}, s\rangle / \langle h, s\rangle\}$
  end if
  Set $x_{k,\ell+1} \leftarrow x_{k,\ell} + \alpha s$ and $d_{k,\ell+1} \leftarrow d_{k,\ell} + \alpha h$
  Update the model value $q_{k,\ell+1} \leftarrow q + \alpha\langle d_{k,\ell}, s\rangle + \frac{1}{2}\alpha^2\langle h, s\rangle$
  Compute $\lambda_{\ell+1} \leftarrow \max\{\lambda_{\min}, \min\{\lambda_{\max}, \langle s, s\rangle / \langle h, s\rangle\}\}$
  Set $\ell \leftarrow \ell + 1$
end while
Return $x_{k+1} \leftarrow x_{k,\ell+1}$ as the approximate solution

eration of Algorithm 2 requires a single application of the Hessian B_k. Moreover, due to the decreasing sequence of model values generated by Algorithm 2, (6c) is guaranteed to be satisfied.

Subproblem Projection Algorithm. To ensure that Algorithm 2 is practical, we must efficiently compute the projection $\mathbf{P}_{\mathcal{C}_k}(\cdot)$. In the following result, we show that computing $\mathbf{P}_{\mathcal{C}_k}(\cdot)$ requires the solution to a one-dimensional root finding problem.

Proposition 1 *The projection of $x \in H$ onto $\mathcal{C}_k$ is given by*

$$\mathbf{P}_{\mathcal{C}_k}(x) = \begin{cases} \mathbf{P}_{\mathcal{C}}(x) & \text{if } \|\mathbf{P}_{\mathcal{C}}(x) - x_k\| \leqslant \Delta_k \\ \mathbf{P}_{\mathcal{C}}(x_k + t^\star(x - x_k)) & \text{if } \|\mathbf{P}_{\mathcal{C}}(x) - x_k\| > \Delta_k \end{cases},$$

where $t^\star \in [0,1]$ is any $t \in [0,1]$ that satisfies

$$\phi(t) := \|\mathbf{P}_{\mathcal{C}}(x_k + t(x - x_k)) - x_k\| - \Delta_k = 0. \tag{8}$$

Here, ϕ is nondecreasing and continuous on $[0,1]$ with $\phi(0) = -\Delta_k$ and $\phi(1) > 0$.

Proof If $\|\mathbf{P}_{\mathcal{C}}(x) - x_k\| \leqslant \Delta_k$, then (2) ensures that $\mathbf{P}_{\mathcal{C}_k}(x) = \mathbf{P}_{\mathcal{C}}(x)$. Now, suppose $\|\mathbf{P}_{\mathcal{C}}(x) - x_k\| > \Delta_k$. We first show that $\mathbf{P}_{\mathcal{C}_k}(x)$ satisfies the trust-region constraint with equality. Set $y = \mathbf{P}_{\mathcal{C}}(x)$ and $y_k = \mathbf{P}_{\mathcal{C}_k}(x)$, and define

$$\psi(t) := \|y + t(y_k - y) - x\|^2 = \|y - x\|^2 + 2t\langle y - x, y_k - y\rangle + t^2\|y_k - y\|^2. \tag{9}$$

The second term in (9) is nonnegative by [30, L. 1] and therefore ψ is increasing for $t \geqslant 0$. Consequently, if $\|y_k - x_k\| < \Delta_k$, then we arrive at a contradiction by choosing $t_0 \in [0,1)$ to satisfy $\|y + t_0(y_k - y) - x_k\| = \Delta_k$, and noting that

$\psi(t_0) < \psi(t)$. Now, to determine the explicit form for $\mathbf{P}_{\mathcal{C}_k}(x)$, we reformulate the projection problem as

$$\min_{y\in H}\left\{\sup_{\mu\geqslant 0}\ \left\{\frac{1}{2}\|y-x\|^2+\frac{\mu}{2}(\|y-x_k\|^2-\Delta_k^2)\right\}\right\}\quad\text{subject to}\quad y\in\mathcal{C}.$$

By [19, Th. 1], we can interchange the minimization and supremum to obtain

$$\sup_{\mu\geqslant 0}\ \min_{y\in H}\left\{\left\{\frac{1}{2}\|y-x\|^2+\frac{\mu}{2}(\|y-x_k\|^2-\Delta_k^2)\right\}\quad\text{subject to}\quad y\in\mathcal{C}\right\},$$

where the inner minimization problem has the unique solution

$$y(t) := \mathbf{P}_{\mathcal{C}}(x_k + t(x - x_k)).$$

Here, we have made the change of variables $t = 1/(1+\mu) \in [0,1]$ for $\mu \geqslant 0$. This gives the desired form for $\mathbf{P}_{\mathcal{C}_k}(x)$. By [30, L. 2], ϕ is nondecreasing. In fact, if $0 \leqslant t < t'$ and $y(t) \neq y(t')$, then $\phi(t) < \phi(t')$. This ensures that if $0 < t < t' \leqslant 1$ and both satisfy (8), then $\mathbf{P}_{\mathcal{C}_k}(x) = y(t) = y(t')$. Finally, it is straightforward to verify that ϕ is continuous with $\phi(0) = -\Delta_k$ and $\phi(1) > 0$. □

By Proposition 1, we can compute the projection $\mathbf{P}_{\mathcal{C}_k}(\cdot)$ by finding a root of ϕ. In principle, one can compute a root of ϕ using any method. In our experience, basic secant methods like regula falsi often take short steps, resulting in slow convergence. For this reason, we employ Brent's method [10], which safeguards against small step sizes, while maintaining the rapid local convergence of the secant method. Similar to bisection, Brent's method is guaranteed to converge to a given bracket size $\epsilon > 0$ in $\lceil -\log_2(\epsilon)\rceil^2$ iterations, but often converges faster due to the secant approximation.

3 Numerical Results

The feasible set $\mathcal{C}$ for each problem considered in this section is polyhedral and can be written in the standard form

$$\mathcal{C} = \{\mathbf{x}\in\mathbb{R}^n \,|\, \mathbf{c}^\top\mathbf{x} = d,\ \ \mathbf{l}\leqslant\mathbf{x}\leqslant\mathbf{u}\}.$$

for $\mathbf{c}\in\mathbb{R}^n$, $d\in\mathbb{R}$, and $\mathbf{l}\leqslant\mathbf{u}\in\mathbb{R}^n$. To project onto $\mathcal{C}$, we employ the algorithm described in [16]. This approach solves the dual problem associated with the projection onto $\mathcal{C}$ using bracketing and a modified secant method. We compare our approach (denoted `TRSPG`) with the linearly constrained trust-region algorithm in [23] (denoted `LMTR`), the line-search projected quasi-Newton method developed in [29] (denoted `PQN`), the spectral projected gradient method (denoted `SPG`) [5] as well as the bound-constrained augmented Lagrangian method [15] in which we solve the augmented Lagrangian subproblem using `LMTR` (denoted `AL-LMTR`) and `TRSPG` (denoted `AL-TRSPG`).

All algorithms are implemented in the Trilinos package *Rapid Optimization Library* (ROL) [21]. ROL is an open-source C++ library consisting of numerous matrix-free nonlinear optimization algorithms. The following examples are contained in the directory

rol/examples/PDE-OPT/published/TRSPG_Kouri2021

of ROL. All numerical studies were performed on a Red Hat Enterprise Linux (version 7.8) workstation with four Xeon E5-2670 v2 processors (10 cores, 2.50GHz base frequency) and 64GB of RAM. All software was compiled with the Intel 19.0.5 C++ compiler `icpc`.

- `LMTR` is a realization of Algorithm 1. In fact, our implementations of `LMTR` and `TRSPG` use the same GCP algorithm (see [23] for more details). `LMTR` and `TRSPG` differ in the solution to the trust-region subproblem (3). In particular, `LMTR` utilizes the quadratic model (4) with all active variables removed. `LMTR` then applies a truncated projected CG algorithm to approximately solve the resulting equality constrained quadratic subproblem. Since the CG step is not guaranteed to be feasible, `LMTR` performs a projected search to restore feasibility and to ensure that the fraction of Cauchy decrease condition (6c) is satisfied. If the truncated CG algorithm finds a direction of negative curvature, then the algorithm steps to the trust-region boundary along this direction. This feature can produce poor steps for nonconvex problems.
- `PQN` is a line-search quasi-Newton method that uses the BFGS secant approximation of the Hessian $\nabla^2 f(x)$. At each iteration, `PQN` solves a quadratic subproblem similar to (3) without the trust-region constraint using the SPG algorithm. To globalize, `PQN` performs a line search that ensures sufficient decrease. The positive definite Hessian approximation used in `PQN` may result in small steps if the objective function is nonconvex.
- `AL-TRSPG` and `AL-LMTR` penalize the linear equality constraints using an augmented Lagrangian and approximately solve bound-constrained subproblems using `TRSPG` and `LMTR`, respectively. These approaches are fundamentally different than the previously described methods, which project onto the feasible set $\mathcal{C}$. By penalizing equality constraints, the required projection onto the bounds is trivial. Consequently, it is interesting to compare the performance of these algorithms to see if the more complicated projections onto $\mathcal{C}$ are worthwhile.

We terminate `TRSPG`, `LMTR`, `PQN`, and `SPG` based on the condition

$$\|\mathbf{P}_{\mathcal{C}}(\mathbf{x}_k - \nabla f(\mathbf{x}_k)) - \mathbf{x}_k\| \leqslant 5 \times 10^{-6}.$$

Furthermore, we stop `AL-TRSPG` and `AL-LMTR` based on the conditions

$$|\mathbf{c}^\top \mathbf{x}_k - d| \leqslant 5 \times 10^{-6} \quad \text{and} \quad \|\mathbf{P}_{[\mathbf{l},\mathbf{u}]}(\mathbf{x}_k - \nabla L_k(\mathbf{x}_k)) - \mathbf{x}_k\| \leqslant 5 \times 10^{-6}.$$

Here, L_k denotes the augmented Lagrangian at the k^{th} iteration. In addition, we terminate if the number of iterations exceeds 200 or if the norm of a trial step falls below 10^{-14}. For all SPG-based algorithms, we use the following parameters: $\lambda_{\min} = 10^{-12}$, $\lambda_{\max} = 10^{12}$, $\gamma = 10^{-4}$, $\sigma_1 = 0.1$, $\sigma_2 = 0.9$, and $M = 10$. Here, σ_1 and σ_2 are line-search step size safeguards, γ is the sufficient decrease parameter, and M is the storage limit for the nonmonotone line search. See [5] for more details. For all trust-region algorithms, we set: $\Delta_0 = 20$, $\eta_1 = 0.05$, $\eta_2 = 0.9$, $\gamma_1 = 0.25$, and $\gamma_2 = 2.5$. For `TRSPG`, we choose $L = 25$, $\epsilon_1 = 10^{-4}$ and $\epsilon_2 = 10^{-2}$ in Algorithm 2, and the bracketing tolerance $\epsilon = 10\epsilon_{\text{mach}}$, where ϵ_{mach} is machine epsilon. Similar to `TRSPG`, we limit the SPG subproblem solver in `PQN` to a maximum of 25 iterations and the subproblem solvers in `AL-TRSPG` and `AL-LMTR` to a maximum of 40 iterations.

We provide tables to summarize the performance of the six algorithms for each example. The column headings in these tables are: the number of iterations (`iter`), the number of objective function evaluations (`fval`), the number of gradient evaluations (`grad`), the number of Hessian applications (`hess`), the number of projections onto $\mathcal{C}$ (`proj`), and the wall-clock time in seconds (`time(s)`). We note that `SPG` and `PQN` do not use the Hessian and that `AL-TRSPG` and `AL-LMTR` do not project onto $\mathcal{C}$.

3.1 Data Science

The following two examples are quadratic programs arising in data science. The first example is a convex lasso regression problem whereas the second is a dual kernel support vector machine (SVM) problem. The objective functions and constraints for these examples were implemented using basic linear algebra subprograms (BLAS) from the Intel Math Kernel Library (MKL).

Lasso Regression. In this example, we solve the lasso regression problem

$$\min_{\mathbf{x}\in\mathbb{R}^n} \ \tfrac{1}{2m}\|\mathbf{ADx}-\mathbf{b}\|_2^2 \quad \text{subject to} \quad \|\mathbf{Dx}\|_1 \leqslant t,$$

where $t>0$, $\mathbf{x}=(x_1,\ldots,x_n)^\top$, $\mathbf{A}\in\mathbb{R}^{m\times n}$, $\mathbf{b}\in\mathbb{R}^m$, and $\|\cdot\|_p$ denotes the usual vector p-norm. We set the scaling matrix $\mathbf{D}\in\mathbb{R}^{n\times n}$ to be diagonal with entries $d_i = 1/\max_j |A_{ji}|$ for $i=1,\ldots,n$. The entries in $\mathbf{A}$ and $\mathbf{b}$ are generated from the `houses` dataset, which can be found in the StatLib dataset archive [25]. This data was analyzed in [26] and consists of $m=20{,}640$ data points with $n=9$ factors. For our experiments, we set $t=1$. To test `LMTR`, `AL-LMTR` and `AL-TRSPG`, we reformulate this problem, by splitting $\mathbf{x}$ into positive and negative parts, as

$$\begin{aligned}&\min_{\mathbf{u},\,\mathbf{v}\in\mathbb{R}^n} \ \tfrac{1}{2m}\|\mathbf{AD}(\mathbf{u}-\mathbf{v})-\mathbf{b}\|_2^2\\ &\text{subject to} \quad \sum_{i=1}^n d_i(u_i+v_i)\leqslant t,\ \ u_i\geqslant 0,\ v_i\geqslant 0\ \ i=1,\ldots,n,\end{aligned}$$

where $\mathbf{u}=(u_1,\ldots,u_n)^\top$ and $\mathbf{v}=(v_1,\ldots,v_n)^\top$. Table 1 summarizes the performance of each algorithm. We ran each algorithm from the feasible initial guess

method	iter	fval	grad	hess	proj	time(s)
TRSPG	5	6	6	112	496	0.021
LMTR	3	4	4	19	22	0.009
PQN	13	14	14	---	368	0.067
SPG	41	45	42	---	84	0.007
AL-TRSPG	6	99	99	2464	---	0.306
AL-LMTR	4	14	14	142	---	0.020

Table 1: Algorithm comparison for the lasso regression example.

$\mathbf{u}\equiv 0$ and $\mathbf{v}\equiv 0$. All algorithms, excluding `AL-TRSPG`, performed comparably with `SPG` and `LMTR` performing slightly better than the rest. Although `TRSPG` performed well, it required more applications of the Hessian and more projections onto $\mathcal{C}$ than both `SPG` and `LMTR`, which accounts for the additional computational time.

Dual Support Vector Machine. In this example, we solve the dual form for a sigmoid kernel SVM. The optimization problem is the quadratic program

$$\min_{\alpha\in\mathbb{R}^n} \left\{ \frac{1}{2}\sum_{i=1}^{n}\sum_{j=1}^{n} \alpha_i y_i k(\mathbf{x}_i,\mathbf{x}_j) y_j \alpha_j - \sum_{i=1}^{n}\alpha_i \right\}$$
$$\text{subject to} \quad \sum_{i=1}^{n} \alpha_i y_i = 0, \;\; 0 \leqslant \alpha_i \leqslant 1 \;\; i = 1,\ldots,n.$$

Here, $\mathbf{x}_i \in \mathbb{R}^m$ is a data point and $y_i \in \{-1, 1\}$ is the label corresponding to $\mathbf{x}_i$. In addition, $k : \mathbb{R}^m \times \mathbb{R}^m \to \mathbb{R}$ is the sigmoid kernel given by

$$k(\mathbf{x},\mathbf{y}) := \tanh(\kappa \mathbf{x}^\top \mathbf{y} + c)$$

with $\kappa = 1/m$ and $c = -0.1$. We use the `phishing` data set [18], downloaded from the `LIBSVM` data repository [12], to test the algorithms. This data set consists of $n = 11{,}055$ data points and $m = 68$ features. Table 2 summarizes the performance of each algorithm. We ran each algorithm from the feasible initial guess $\alpha \equiv 0$. `PQN`

method	iter	fval	grad	hess	proj	time(s)
TRSPG	4	5	5	58	282	1.513
LMTR	17	18	16	874	662	17.923
PQN	200	201	201	---	9300	82.314
SPG	23	24	24	---	48	0.909
AL-TRSPG	165	3131	3131	80455	---	1670.322
AL-LMTR	17	430	410	17058	---	342.925

Table 2: Algorithm comparison for the dual SVM example.

did not converge within the iteration limit. However, the final optimality criterion produced by `PQN` was 1.76×10^{-5} and the final objective function value was within $4.297 \times 10^{-9}\%$ of the minimal value computed by the other algorithms. `SPG` and `TRSPG` performed comparably for this example and both outperformed the other algorithms. `SPG` was slightly faster than `TRSPG` since it required 234 fewer projections onto $\mathcal{C}$ than `TRSPG`. For further comparison, we solved this problem using `LIBSVM`, which required roughly 14.199 seconds and 2,976 iterations. Consequently, `TRSPG` (1.513 seconds), `LMTR` (17.923 seconds), and `SPG` (0.909 seconds) compare favorably with `LIBSVM`.

3.2 PDE-Constrained Optimization

In this subsection, we test our approach on two discretized PDE-constrained optimization problems. For both examples, $\Omega \subset \mathbb{R}^2$ is the physical domain, $H = L^2(\Omega)$ and the infinite-dimensional feasible set is given by

$$\mathcal{C} = \left\{ \rho \in L^2(\Omega) \,\middle|\, \int_\Omega \rho(x)\,\mathrm{d}x = v_0|\Omega|, \;\; 0 \leqslant \rho \leqslant 1 \right\}.$$

Here, $v_0 \in (0, 1)$ and $|\Omega|$ denotes the volume of Ω. We refer the reader to [20,31] for more details on the numerical discretization and solution of PDE-constrained optimization problems, including how to evaluate the objective function and its

gradient as well as how to apply its Hessian to a vector. The objective functions and constraints for these examples were implemented using the Trilinos packages Intrepid [8] for finite element discretizations, Amesos [27] for linear solvers, and Tpetra [3] for sparse linear algebra. Within Amesos, we employ the Pardiso solver [28] from Intel MKL to solve the discretized PDEs.

Elastic Topology Optimization. Let $\Omega = (0,2) \times (0,1)$. We consider the archetypal topology optimization problem of determining a distribution of material $\rho : \Omega \to [0,1]$ that minimizes the compliance of the resulting elastic structure and satisfies a constraint on the volume. Here, $\rho(x) = 0$ represents a void and $\rho(x) = 1$ represents the material. This problem is written as

$$\min_{\rho \in \mathcal{C}} \int_{\Gamma_t} T(x)[S(\rho)](x)\,\mathrm{d}x,$$

where the displacement field $u = S(\rho)$ solves the linear elasticity equations

$$\begin{aligned} -\nabla \cdot (K(\rho) : \varepsilon(u)) &= 0 && \text{in } \Omega \\ \varepsilon(u) &= \tfrac{1}{2}(\nabla u + \nabla u^\top) && \text{in } \Omega \\ u &= 0 && \text{on } \Gamma_{\text{fixed}} \\ K(\rho) : \varepsilon(u)\mathbf{n} &= T && \text{on } \Gamma_{\text{tract}}. \end{aligned}$$

The function $T : \Gamma_{\text{tract}} \to \mathbb{R}^2$ is a traction force applied to the boundary segment $\Gamma_{\text{tract}} \subset \partial\Omega$ and the structure is fixed on the boundary segment $\Gamma_{\text{fixed}} \subset \partial\Omega$. We employ the SIMP material model [4] and the Helmholtz filter [22].

We discretize ρ as a piecewise constant function on a mesh of 26,880 quadrilateral elements. We further discretize the linear elasticity equations using continuous piecewise linear finite elements on the same mesh. We note that the objective function is nonlinear and highly nonconvex as it requires the solution to the linear elasticity equations, which depend nonlinearly on ρ. Table 3 summarizes the performance for the six algorithms. We ran each algorithm from the feasible initial

method	iter	fval	grad	hess	proj	time(s)
TRSPG	9	10	10	236	1187	14.88
LMTR	33	34	31	398	370	20.99
PQN	62	117	63	---	2364	65.55
SPG	84	90	85	---	170	41.24
AL-TRSPG	9	52	51	1154	---	38.08
AL-LMTR	10	246	234	3803	---	160.96

Table 3: Algorithm comparison for the elastic topology optimization problem.

guess $\rho \equiv v_0$. `TRSPG` outperforms all other algorithms in function, gradient and Hessian evaluations as well as wall-clock time. Although `TRSPG` and `LMTR` are both versions of Algorithm 1, `TRSPG` outperforms `LMTR` (similarly, `AL-TRSPG` outperforms `AL-LMTR`). A possible reason for this is that the objective function Hessian is indefinite, which causes the `LMTR` CG procedure to terminate after only a few iterations, producing poor steps. In contrast, the `TRSPG` algorithm provides a more accurate solution to the trust-region subproblem and tends to produce successful steps for this example. Since this problem is nonconvex, it is worth noting that the final objective function values produced by each algorithm were within 0.386% of each

other, signifying convergence to stationary points with similar objective function values.

Diffuser Design. Let $\Omega = (0,1)^2$. We consider the design of a diffuser using topology optimization. As in the previous example, the optimization variables represent the distribution of material in Ω. However, for this problem our goal is to compute a material distribution that minimizes the total potential power of a flow and satisfies a volume constraint. This problem is written as

$$\min_{\rho \in \mathcal{C}} \int_\Omega \{\nabla[S_u(\rho)](x) \cdot \nabla[S_u(\rho)](x) + [\alpha(\rho)](x)[S_u(\rho)](x) \cdot [S_u(\rho)](x)\} \, \mathrm{d}x,$$

where the flow velocity $u = S_u(\rho)$ and pressure $\pi = S_\pi(\rho)$ solve the steady Navier-Stokes equations

$$\begin{aligned} -\nu \Delta u + u \cdot \nabla u + \nabla \pi &= -\alpha(\rho) u && \text{in } \Omega \\ \nabla \cdot u &= 0 && \text{in } \Omega \\ u &= u_{\text{in}} && \text{on } \Gamma_{\text{in}} \\ u &= 0 && \text{on } \Gamma_{\text{wall}} \\ \nu \nabla u \cdot \mathbf{n} - \pi \mathbf{n} &= 0 && \text{on } \Gamma_{\text{out}}. \end{aligned}$$

The parameter ν is the kinematic viscosity and u_{in} is the prescribed inflow velocity. The boundary $\partial\Omega$ is partitioned into three non-overlapping segments: the inflow boundary $\Gamma_{\text{in}} = \{0\} \times [0,1]$, the outflow boundary $\Gamma_{\text{out}} = \{1\} \times [\frac{1}{3}, \frac{2}{3}]$ and the no slip boundary $\Gamma_{\text{wall}} = \partial\Omega \backslash (\Gamma_{\text{in}} \cup \Gamma_{\text{out}})$. The function $\alpha(\rho)$ models the inverse permeability and has the form

$$\alpha(\rho) = \overline{\alpha} + (\underline{\alpha} - \overline{\alpha}) \frac{\rho(1+q)}{q+\rho},$$

where $\overline{\alpha} = 2.5 \times 10^4$, $\underline{\alpha} = 2.5 \times 10^{-4}$, and $q = 0.1$. This problem is an extension of the topology optimization problem considered in [9, § 4.2] to the Navier-Stokes equations. See [17] for similar problems.

We discretize ρ as a piecewise constant function on a mesh of 30,720 quadrilateral elements and we discretize the Navier-Stokes equations using Taylor-Hood finite elements on the same mesh. To evaluate the objective function, we solve the discretized Navier-Stokes equations using a line-search Newton method. We again note that the objective function is nonlinear and highly nonconvex. We summarize the algorithmic performance in Table 4. We ran each algorithm from the feasible

method	iter	fval	grad	hess	proj	time(s)
TRSPG	11	12	12	306	1595	657.47
LMTR	22	23	20	730	317	1402.33
PQN	84	85	85	---	3082	1344.72
SPG	130	308	131	---	262	3178.22
AL-TRSPG	5	22	22	437	---	935.01
AL-LMTR	4	33	33	1127	---	2125.18

Table 4: Algorithm comparison for the diffuser design problem.

initial guess $\rho \equiv v_0$. Similar observations can be made as in the previous example. In particular, `TRSPG` significantly outperforms all other algorithms in wall-clock

time as well as function, gradient and Hessian evaluations. Interestingly, for this example, `AL-TRSPG` outperforms all methods with the exception of `TRSPG`, suggesting that the cost difference between projecting onto $\mathcal{C}$ and the penalty iteration is negligible. Since this problem is nonconvex, it is worth noting that the final objective function values produced by each algorithm were within $(2 \times 10^{-9})\%$ of each other, signifying convergence to stationary points with similar objective function values.

4 Conclusions

We described a trust-region subproblem solver for convex-constrained optimization problems based on the SPG method. Our algorithm is simple to implement, compared with other convex-constrained trust-region algorithms. In addition, our approach is completely matrix free, enabling the solution of enormous problems. In order for this approach to be practical, the projection onto the feasible set $\mathcal{C}$ must be efficient to compute. This projection is often performed using iterative methods, leading to inexact projections. An interesting future direction is the rigorous treatment of inexact projections akin to the work in [6]. In addition, the SPG method employs a safeguarded spectral step, which can be quite large (e.g., $\lambda_{\max} = 10^{12}$ in our examples). When the spectral step length is large, the computation of the projection can become unstable, motivating research in specialized procedures to accelerate these projections.

References

1. Andretta, M., Birgin, E.G., Martínez, J.M.: Practical active-set Euclidian trust-region method with spectral projected gradients for bound-constrained minimization. Optimization **54**(3), 305–325 (2005)
2. Antil, H., Kouri, D.P., Lacasse, M.D., Ridzal, D.: Frontiers in PDE-constrained Optimization, vol. 163. Springer (2018)
3. Baker, C.G., Heroux, M.A.: Tpetra, and the use of generic programming in scientific computing. Scientific Programming **20**(2), 115–128 (2012)
4. Bendsoe, M.P., Sigmund, O.: Topology optimization: theory, methods, and applications. Springer Science & Business Media (2013)
5. Birgin, E.G., Martínez, J.M., Raydan, M.: Nonmonotone spectral projected gradient methods on convex sets. SIAM Journal on Optimization **10**(4), 1196–1211 (2000)
6. Birgin, E.G., Martínez, J.M., Raydan, M.: Inexact spectral projected gradient methods on convex sets. IMA Journal of Numerical Analysis **23**(4), 539–559 (2003)
7. Birgin, E.G., Martínez, J.M., Raydan, M.: Spectral projected gradient methods: review and perspectives. J. Stat. Softw **60**(3), 1–21 (2014)
8. Bochev, P., Edwards, H.C., Kirby, R.C., Peterson, K., Ridzal, D.: Solving PDEs with Intrepid. Scientific Programming **20**(2), 151–180 (2012)
9. Borrvall, T., Petersson, J.: Topology optimization of fluids in Stokes flow. International journal for numerical methods in fluids **41**(1), 77–107 (2003)
10. Brent, R.P.: Algorithms for minimization without derivatives. Courier Corporation (2013)
11. Burke, J.V., Moré, J.J., Toraldo, G.: Convergence properties of trust region methods for linear and convex constraints. Mathematical Programming **47**(1-3), 305–336 (1990)
12. Chang, C.C., Lin, C.J.: LIBSVM: A library for support vector machines. ACM Transactions on Intelligent Systems and Technology **2**(3), 1–27 (2011). Software available at `http://www.csie.ntu.edu.tw/~cjlin/libsvm`
13. Conn, A.R., Gould, N.I.M., Sartenaer, A., Toint, P.L.: Global convergence of a class of trust region algorithms for optimization using inexact projections on convex constraints. SIAM Journal on Optimization **3**(1), 164–221 (1993). DOI 10.1137/0803009

14. Conn, A.R., Gould, N.I.M., Sartenaer, A., Toint, P.L.: Convergence properties of minimization algorithms for convex constraints using a structured trust region. SIAM Journal on Optimization **6**(4), 1059–1086 (1996). DOI 10.1137/S1052623492236481
15. Conn, A.R., Gould, N.I.M., Toint, P.L.: A globally convergent augmented Lagrangian algorithm for optimization with general constraints and simple bounds. SIAM Journal on Numerical Analysis **28**(2), 545–572 (1991). DOI 10.1137/0728030
16. Dai, Y.H., Fletcher, R.: New algorithms for singly linearly constrained quadratic programs subject to lower and upper bounds. Mathematical Programming **106**(3), 403–421 (2006)
17. Deng, Y., Liu, Z., Wu, J., Wu, Y.: Topology optimization of steady Navier–Stokes flow with body force. Computer Methods in Applied Mechanics and Engineering **255**, 306–321 (2013)
18. Dua, D., Graff, C.: UCI machine learning repository (2017). URL `http://archive.ics.uci.edu/ml`
19. Hartung, J.: An extension of Sion's minimax theorem with an application to a method for constrained games. Pacific J. Math. **103**(2), 401–408 (1982)
20. Heinkenschloss, M.: Numerical solution of implicitly constrained optimization problems. Tech. rep., Rice University (2008)
21. Kouri, D.P., von Winckel, G., Ridzal, D.: ROL: Rapid Optimization Library. `https://trilinos.org/packages/rol` (2017)
22. Lazarov, B.S., Sigmund, O.: Filters in topology optimization based on Helmholtz-type differential equations. International Journal for Numerical Methods in Engineering **86**(6), 765–781 (2011)
23. Lin, C.J., Moré, J.J.: Newton's method for large bound-constrained optimization problems. SIAM Journal on Optimization **9**(4), 1100–1127 (1999)
24. Maciel, M.C., Mendonça, M.G., Verdiell, A.B.: Monotone and nonmonotone trust-region-based algorithms for large scale unconstrained optimization problems. Computational Optimization and Applications **54**(1), 27–43 (2013)
25. Meyer, M., Vlachos, P.: StatLib—datasets archive. `http://lib.stat.cmu.edu/datasets/`. Accessed: 2021-05-10
26. Pace, R.K., Barry, R.: Sparse spatial autoregressions. Statistics & Probability Letters **33**(3), 291–297 (1997)
27. Sala, M., Stanley, K., Heroux, M.: Amesos: A set of general interfaces to sparse direct solver libraries. In: Proceedings of PARA'06 Conference, Umea, Sweden (2006)
28. Schenk, O., Gärtner, K.: Solving unsymmetric sparse systems of linear equations with PARDISO. Future Generation Computer Systems **20**(3), 475–487 (2004)
29. Schmidt, M., Berg, E., Friedlander, M., Murphy, K.: Optimizing costly functions with simple constraints: A limited-memory projected quasi-Newton algorithm. In: D. van Dyk, M. Welling (eds.) Proceedings of the Twelth International Conference on Artificial Intelligence and Statistics, *Proceedings of Machine Learning Research*, vol. 5, pp. 456–463. PMLR, Hilton Clearwater Beach Resort, Clearwater Beach, Florida USA (2009)
30. Toint, P.L.: Global Convergence of a Clas of Trust-Region Methods for Nonconvex Minimization in Hilbert Space. IMA Journal of Numerical Analysis **8**(2), 231–252 (1988). DOI 10.1093/imanum/8.2.231
31. Tröltzsch, F.: Optimal Control of Partial Differential Equations: Theory, Methods, and Applications. Graduate studies in mathematics. American Mathematical Society (2010)
32. Vapnik, V.N.: The Nature of Statistical Learning Theory. Springer New York (2013)