IMA Journal of Numerical Analysis (2016) **36**, 717–742
doi:10.1093/imanum/drv034
Advance Access publication on July 8, 2015

Handling nonpositive curvature in a limited memory steepest descent method

FRANK E. CURTIS* AND WEI GUO
Department of Industrial and Systems Engineering, Lehigh University, Bethlehem, PA, USA
*Corresponding author: frank.e.curtis@gmail.com weg411@lehigh.edu

[Received on 7 November 2014; revised on 10 March 2015]

We propose a limited memory steepest descent (LMSD) method for solving unconstrained optimization problems. As a steepest descent method, the step computation in each iteration requires the evaluation of a gradient of the objective function and the calculation of a scalar step size only. When employed to solve certain convex problems, our method reduces to a variant of the LMSD method proposed by Fletcher (2012, *Math. Program.*, **135**, 413–436), which means that, when the history length parameter is set to 1, it reduces to a steepest descent method inspired by that proposed by Barzilai & Borwein (1988, *IMA J. Numer. Anal.*, **8**, 141–148). However, our method is novel in that we propose new algorithmic features for cases when nonpositive curvature is encountered. That is, our method is particularly suited for solving nonconvex problems. With a nonmonotone line search, we ensure global convergence for a variant of our method. We also illustrate with numerical experiments that our approach often yields superior performance when employed to solve nonconvex problems.

Keywords: unconstrained optimization; nonconvex optimization; steepest descent methods; Barzilai–Borwein methods; limited memory methods.

1. Introduction

Algorithms for finding minimizers of continuously differentiable functions have been the subject of research for centuries. In particular, steepest descent methods—the most basic gradient-based methods—have been the focus of a great deal of work due to their simplicity and effectiveness in many applications. Over the past few decades, great improvements in the practical performance of steepest descent methods have been made simply by the design of clever techniques for choosing the step size in each iteration.

In this paper, we propose a limited memory steepest descent (LMSD) method for solving unconstrained optimization problems whose objective functions are continuously differentiable. Our method is based on the LMSD method recently proposed by Fletcher (2012). In a given iteration, this method, by exploiting previously computed gradient information stored as a set of m vectors, computes a sequence of m step sizes to be employed in a 'sweep' over the next m iterations. The calculations involved in determining these step sizes are motivated by the case of minimizing a strictly convex quadratic form defined by a positive-definite matrix A, where with m previously computed gradients, one can define a Krylov sequence that provides m estimates of eigenvalues of A. (These estimates, or Ritz values, are contained in the spectrum of A, so Fletcher's method belongs to the class often referred to as spectral gradient descent methods.) In particular, considering the choice of $m = 1$ leads to step sizes as chosen in the algorithms proposed by Barzilai & Borwein (1988), which many consider to be the work responsible for inspiring renewed interest in steepest descent methods.

Many have observed the impressive practical performance of Barzilai–Borwein (BB) methods when solving unconstrained optimization problems. Moreover, in his work, Fletcher illustrates that his approach represents a competitive alternative to the well-known limited memory variant of the Broyden–Fletcher–Goldfarb–Shanno (BFGS) algorithm (see Broyden, 1970; Fletcher, 1970; Goldfarb, 1970; Shanno, 1970), otherwise known as the L-BFGS method (see Nocedal, 1980). However, in our opinion, these approaches and their proposed enhancements (see Section 2) suffer from the fact that when the objective function is nonconvex, the sophisticated mechanisms designed to compute step sizes are abandoned, and instead the step sizes are chosen arbitrarily (e.g., as prescribed constants). Such choices can lead to poor performance when solving nonconvex problems.

The main contribution of the algorithm proposed in this paper is that it provides a novel strategy for computing step sizes when solving nonconvex optimization problems. In particular, when nonpositive curvature (as defined later) is encountered, our method adopts a local cubic model of the objective function in order to determine a step size (for $m = 1$) or sequence of step sizes (for $m > 1$). (As mentioned in Section 2, cubic models have previously been employed in the computation of step sizes for steepest descent methods. However, in the work that we cite, emphasis was placed on computing step sizes in convex settings. By contrast, when only positive curvature is encountered, we use a standard quadratic model, as such a choice typically yielded good performance in our experiments. We employ a cubic model in iterations only when nonpositive curvature is present.) As in the case of the original BB methods and the LMSD method of Fletcher, our basic algorithm does not enforce sufficient decrease in the objective in every iteration. However, as is commonly done for variants of BB methods, we remark that, with a nonmonotone line search, a variant of our algorithm attains global convergence guarantees under weak assumptions. Our method also readily adopts the convergence rates attainable by a BB method if/when it reaches a neighbourhood of the solution in which the objective is strictly convex (see Section 2).

Overall, our proposed algorithm is designed to strike a balance between multiple (potentially conflicting) goals simultaneously. On the one hand, our method preserves the impressive theoretical and practical performance of an LMSD method when nonpositive curvature is not an issue; indeed, when nonpositive curvature is not encountered, our approach reduces to a variant of Fletcher's LMSD method. On the other hand, however, our method is designed to compute and employ meaningful step sizes when nonpositive curvature is encountered in such a way that (i) the cost of the step-size computation remains negligible, (ii) the strategy for handling nonpositive curvature can be generalized to cases when more historical information is maintained and exploited (i.e., when $m > 1$), and (iii) on a diverse set of large-scale test problems, our method yields consistently better performance than a method that follows the typical strategy of setting the step size to a large prescribed constant when nonpositive curvature is encountered. To achieve these goals, we do not adopt the approach of other authors who attempt to compute accurate higher-order models using, e.g., Hermite interpolation, as such a technique may not shed any light on what may be a reasonable step size when nonpositive curvature is encountered, and may not be readily generalized when $m > 1$. Rather, we account for nonpositive curvature in the objective function by observing the difference between two quadratic models obtained using typical LMSD estimation strategies—with which we construct a (bounded below) cubic model for determining a step size—in a manner that offers a generalized approach for cases when $m > 1$.

This paper is organized as follows. In Section 2, we provide a brief summary of the original BB methods and a few of their proposed variants. We also briefly review Fletcher's LMSD algorithm, which can be viewed as another BB method variant/extension. In Section 3, we present the details of our proposed algorithm. We first motivate the ideas underlying our approach by considering the case when, at a given iteration, information from the previous iteration is exploited, and then discuss a generalization of the method for cases when information from any number of previous iterations is maintained and utilized. We discuss the details of an implementation of our method in Section 4, and then present the results of numerical experiments in Section 5 which illustrate that our strategies typically yield better performance than some related approaches when solving nonconvex problems. Finally, in Section 6, we present concluding remarks.

The problem that we consider herein is the unconstrained optimization problem

$$\min_{x \in \mathbb{R}^n} f(x), \tag{1.1}$$

where $\mathbb{R}^n$ is the set of n-dimensional real vectors (with $\mathbb{R} := \mathbb{R}^1$) and $f : \mathbb{R}^n \to \mathbb{R}$ is continuously differentiable. The algorithms that we discuss are iterative in that, over $k \in \mathbb{N} := \{0, 1, 2, \ldots\}$, they produce a sequence $\{x_k\} \subset \mathbb{R}^n$ of iterates, where, for each element of the sequence, the subscript corresponds to the iteration number in the algorithm. Given an iterate x_k for some $k \in \mathbb{N}$, we define $f_k := f(x_k)$ as the corresponding function value and $g_k := \nabla f(x_k)$ as the corresponding gradient value. Throughout the paper, we also apply the subscript k to other quantities that appear in an algorithm during iteration k.

2. Literature review

The simplest type of gradient-based method for solving problem (1.1) is a steepest descent method, which is the term we use to describe any iterative method of the form

$$x_{k+1} \leftarrow x_k - \alpha_k g_k \quad \text{for all } k \in \mathbb{N}. \tag{2.1}$$

Here, $x_0 \in \mathbb{R}^n$ is a given initial point and, for all $k \in \mathbb{N}$, the scalar $\alpha_k > 0$ is the kth step size. In the classical steepest descent method of Cauchy, each step size is obtained by an exact line search (see Cauchy, 1847), i.e., assuming that f is bounded below along the ray from x_k along $-g_k$, one sets

$$\alpha_k \in \arg\min_{\alpha \geqslant 0} f(x_k - \alpha g_k).$$

However, in modern variants of steepest descent, alternative step sizes that are cheaper to compute are employed to reduce per-iteration (and typically overall) computational costs. For instance, popular alternatives—which may or may not be modified by a subsequent inexact line search—are those proposed by Barzilai and Borwein in their seminal work, which are described next.

2.1 *Barzilai–Borwein methods*

The 'two-point step-size' method proposed by Barzilai and Borwein has two variants, which differ only in the formulas used to compute the step sizes. We derive these formulas simultaneously now for reference throughout the paper. During iteration $k \in \mathbb{N}_+ := \{1, 2, \ldots\}$, defining the displacement vectors

$$s_k := x_k - x_{k-1} \quad \text{and} \quad y_k := g_k - g_{k-1}, \tag{2.2}$$

the classical secant equation is given by $H_k s_k = y_k$ where H_k represents an approximation of the Hessian of f at x_k. In quasi-Newton methods of the Broyden class (such as the BFGS method), a Hessian approximation $H_k \succ 0$ is chosen such that the secant equation is satisfied and the kth search direction is set as $-H_k^{-1} g_k$ (see Nocedal & Wright, 2006). However, the key idea in BB methods is to maintain a steepest descent framework by approximating the Hessian by a scalar multiple of the identity matrix in such a way that the secant equation is satisfied only in a least-squares sense. In particular, consider

$$\min_{\bar{q}\in\mathbb{R}} \tfrac{1}{2}\|(\bar{q}I)s_k - y_k\|_2^2 \quad \text{and} \quad \min_{\hat{q}\in\mathbb{R}} \tfrac{1}{2}\|s_k - (\hat{q}^{-1}I)y_k\|_2^2.$$

Assuming that $s_k^T y_k > 0$ (which is guaranteed, e.g., when $s_k \neq 0$ and f is strictly convex), the solutions of these one-dimensional problems are, respectively,

$$\bar{q}_k := \frac{s_k^T y_k}{s_k^T s_k} \quad \text{and} \quad \hat{q}_k := \frac{y_k^T y_k}{s_k^T y_k}. \tag{2.3}$$

That is, $\bar{q}_k I$ and $\hat{q}_k I$ represent simple approximations of the Hessian of f along the line segment $[x_{k-1}, x_k]$, meaning that if one minimizes the quadratic model of f at x_k along $-g_k$ given by

$$f(x_k - \alpha g_k) \approx f_k - \alpha\|g_k\|_2^2 + \tfrac{1}{2}\alpha^2 q_k\|g_k\|_2^2,$$

then for $q_k = \bar{q}_k$ and $q_k = \hat{q}_k$, respectively, one obtains two potential values for the step size α_k, namely

$$\bar{\alpha}_k := \frac{s_k^T s_k}{s_k^T y_k} \quad \text{and} \quad \hat{\alpha}_k := \frac{s_k^T y_k}{y_k^T y_k}. \tag{2.4}$$

(Further discussion on the difference between these step sizes and their corresponding Hessian approximations is given in Section 3.) Overall, the main idea in such an approach is to employ a two-point approximation to the secant equation in order to construct a simple approximation of the Hessian of f at x_k, which in turn leads to a quadratic model of f at x_k that can be minimized to determine the step size α_k.

BB methods and enhancements to them have been a subject of research for over two decades. In their original work, Barzilai & Borwein (1988) proved that either of their two step-size choices leads to global convergence and an R-superlinear local convergence rate when (2.1) is applied to minimize a two-dimensional strictly convex quadratic. Raydan (1993) extended these results to prove that such methods are globally convergent when applied to minimize any finite-dimensional strictly convex quadratic. Dai & Liao (2002) also extended these results to show that, on such problems, BB methods attain an R-linear rate of convergence.

An interesting feature of BB methods, even when applied to minimize strictly convex quadratics, is that they are not guaranteed to yield monotonic decreases in the objective function or a typical stationarity measure for problem (1.1). That is, when they converge to a minimizer of f, neither the sequence of function values $\{f_k\}$ nor the sequence of gradient norms $\{\|g_k\|\}$ is guaranteed to decrease monotonically. Hence, a variety of extensions of the original BB methods have been designed that ensure convergence when minimizing general continuously differentiable objective functions by incorporating a nonmonotone line search such as the one proposed by Grippo *et al.* (1986), or, more recently, the one proposed by Zhang & Hager (2004). Extensions of BB methods also typically try to produce better step sizes by employing higher-order models of f and/or alternating exact line searches (i.e., Cauchy step sizes) into the iterative sequence (2.1). A few examples are the following. Raydan (1997) proposed a globally convergent BB method using the line search of Grippo *et al.* (1986) and Dai *et al.* (2002) followed this work by proposing interpolation techniques to derive a few alternative step sizes; they use interpolation to recover the original BB step sizes and employ a cubic model to derive alternatives. Their methods are also globalized by the line search of Grippo *et al.* (1986). More recently, Yuan (2006) proposed the incorporation of Cauchy step sizes into the iterative process to improve the efficiency of the algorithm, a technique later extended by De Asmundis *et al.* (2014), motivated by work in De Asmundis *et al.* (2013) with their collaborator Riccio in a monotone gradient scheme. There has also been recent work by Xiao *et al.* (2010) that proposes alternative step sizes using an alternative secant equation, as well as work by Biglari & Solimanpur (2013) that proposes alternative step sizes derived by fourth-order interpolation models. These later articles employ the nonmonotone line search of Zhang & Hager (2004).

Despite all of the unique features of the BB method variants that have been proposed in the literature, to the best of our knowledge there are no variants that focus on the inefficiencies that may arise when f is nonconvex. (One exception is the recent work by Kafaki & Fatemi (2013)) that modifies a BB step size using a similar strategy to the modified BFGS method proposed by Li & Fukushima (2001). However, this strategy is quite different from the strategy proposed in this paper.) In such cases, the inner product $s_k^T y_k$ may be nonpositive, and must be handled as a special case in all of the algorithms previously described. For example, in Raydan (1997), Dai *et al.* (2002), Xiao *et al.* (2010) and Biglari & Solimanpur (2013), when a nonpositive step size is computed, the algorithms revert to setting the step size to a large user-defined constant. (In Barzilai & Borwein, 1988; Raydan, 1993; Yuan, 2006; De Asmundis *et al.*, 2014, only convex quadratics are considered, so no strategies are proposed for handling nonpositive curvature.) Such a choice fails to capture any information from the objective function, which may be detrimental to performance.

As a brief illustration of the step sizes computed in a BB method in which $s_k^T y_k < 0$ implies that one sets α_k to a prescribed positive constant (as in Raydan, 1997; Dai *et al.*, 2002; Xiao *et al.*, 2010; Biglari & Solimanpur, 2013), consider an arbitrary $k \in \mathbb{N}_+$ and suppose that $g_{k-1} = (-1, 0)$ and $\alpha_{k-1} = 1$ so that $s_k = -\alpha_{k-1} g_{k-1} = (1, 0)$. The contour plots in Fig. 1 illustrate the step sizes that would be computed as a function of the gradient $g_k \in [-3, 1] \times [-2, 2]$. The plots differ since, on the left (respectively, right), we plot the step sizes that would be computed when $s_k^T y_k > 0$ implies $\alpha_k \leftarrow 1/\bar{q}_k$ (respectively, $\alpha_k \leftarrow 1/\hat{q}_k$). These plots lead to a few important observations. First, one can observe that when $s_k^T y_k > 0$ and the vectors s_k and y_k are parallel (corresponding to the horizontal axes in the plots), the step sizes in the two plots are the same since $\bar{q}_k = \hat{q}_k$ in such cases. However, it is interesting to note the step sizes that result when $s_k^T y_k > 0$ while s_k and y_k are nearly orthogonal: setting $\alpha_k \leftarrow 1/\bar{q}_k$ leads to extremely large step sizes, whereas setting $\alpha_k \leftarrow 1/\hat{q}_k$ leads to extremely small step sizes. Clearly, the two BB alternatives differ significantly for such g_k. That being said, if one were to employ a globalization mechanism such as a Wolfe line search (see Nocedal & Wright, 2006), then a typical strategy would ensure that $s_k^T y_k$ is large in proportion to $\|s_k\|_2^2$. In such an approach, the only values computed in the algorithm would be those illustrated in the regions between the two lines emanating from $(-1, 0)$ drawn in the plots. In these regions, the two BB step-size alternatives do not reach such extremes, though they still differ substantially for certain values of g_k. Hence, a Wolfe line search can diminish the effect of the differences between these step sizes, though it should be noted that such a line search can be expensive as it may require many additional function and gradient evaluations. One final (striking) observation about the contours in Fig. 1 is that both strategies fail to exploit any useful information when $s_k^T y_k < 0$. We comment on this further in Section 3.

2.2 *Fletcher's LMSD method*

Fletcher's LMSD method represents an alternative to the original BB methods that is entirely different from those described in the previous subsection. Rather than attempt to compute a better step size based on information that can be extracted only from the previous iteration, his approach involves the storage and exploitation of information from m previous iterations, with which a sequence of m step sizes—to be employed in the subsequent m iterations—are computed. To be more precise, consider iteration $k \geqslant m$

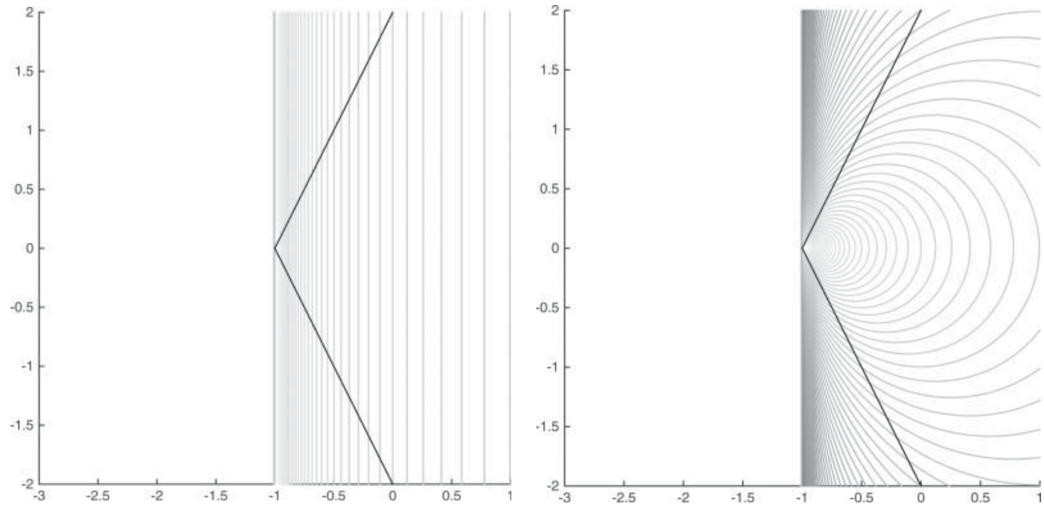

FIG. 1. Illustration of the step sizes computed, as a function of the gradient g_k, in an algorithm in which $s_k^{\mathrm{T}} y_k > 0$ implies $\alpha_k \leftarrow 1/\bar{q}_k$ (left) versus one in which $s_k^{\mathrm{T}} y_k > 0$ implies $\alpha_k \leftarrow 1/\hat{q}_k$ (right). In both cases, $s_k^{\mathrm{T}} y_k < 0$ implies α_k is set to a constant; hence, no contour lines appear in the left half of each plot.

for some user-specified integer parameter $m \geqslant 1$ and suppose that a matrix of gradients (computed at previous iterates), namely

$$G_k := [g_{k-m} \quad \cdots \quad g_{k-1}], \tag{2.5}$$

is available. The key idea underlying Fletcher's proposed method is that, in the case of minimizing a strictly convex quadratic form defined by a positive-definite matrix A, a reasonable set of step sizes can be obtained by computing the reciprocals of the eigenvalues of the symmetric tridiagonal matrix

$$T_k := Q_k^{\mathrm{T}} A Q_k,$$

where Q_k is the orthogonal matrix obtained in the (thin) QR-factorization of the matrix G_k (see Golub and Van Loan, 1996, Theorem 5.2.2). In fact, if one has $m = n$, then choosing step sizes in this manner leads to finite termination of the algorithm in n steps. Furthermore, the matrix T_k can be obtained without access to the matrix A, such as through the partially extended Cholesky factorization $G_k^{\mathrm{T}}[G_k \; g_k] = R_k^{\mathrm{T}}[R_k \; r_k]$ to obtain

$$T_k = [R_k \quad r_k] J_k R_k^{-1}, \tag{2.6}$$

where R_k is the upper triangular matrix obtained in the (thin) QR-factorization of G_k (meaning that it is the upper triangular Cholesky factor of $G_k^{\mathrm{T}} G_k$; see Golub and Van Loan, 1996, Theorem 5.2.2) and

$$J_k := \begin{bmatrix} \alpha_{k-m}^{-1} & & \\ -\alpha_{k-m}^{-1} & \ddots & \\ & \ddots & \alpha_{k-1}^{-1} \\ & & -\alpha_{k-1}^{-1} \end{bmatrix}. \tag{2.7}$$

Exploiting this latter representation, Fletcher extends his approach to the minimization of general objective functions. In particular, by storing G_k and computing T_k in a manner similar to (2.6), he outlines a 'Ritz Sweep' algorithm that, in his experiments, performs as well as an L-BFGS method. In this extension to a more general setting (i.e., nonquadratic functions), Fletcher incorporates line searches and other features to overcome certain issues that may arise and to promote convergence. Some of his procedures are discussed in Section 3.2, but the reader should refer to his article for a more complete discussion.

It should be noted that in the case of minimizing a strictly convex quadratic and with $m = 1$, the formula (2.6) yields $T_k = \bar{q}_k$ (recall (2.3)), which reveals that choosing step sizes as the reciprocals of the eigenvalues of T_k corresponds to the first BB alternative. Fletcher also remarks that a similar strategy can be designed corresponding to the second BB step size. In particular, defining

$$\begin{bmatrix} R_k & r_k \\ 0 & \rho_k \end{bmatrix}^{\mathrm{T}} \begin{bmatrix} R_k & r_k \\ 0 & \rho_k \end{bmatrix} \quad \text{as the Cholesky factorization of } [G_k \quad g_k]^{\mathrm{T}}[G_k \quad g_k],$$

he defines the corresponding pentadiagonal matrix

$$P_k := R_k^{-T} J_k^{\mathrm{T}} \begin{bmatrix} R_k & r_k \\ 0 & \rho_k \end{bmatrix}^{\mathrm{T}} \begin{bmatrix} R_k & r_k \\ 0 & \rho_k \end{bmatrix} J_k R_k^{-1}. \tag{2.8}$$

He explains that, in the case of minimizing a strictly convex quadratic form defined by A, appropriate step sizes are given by the eigenvalues of $P_k^{-1}T_k$; in particular, with $m = 1$, the formulas (2.6) and (2.8) yield $P_k^{-1}T_k = \hat{\alpha}_k$ (recall (2.4)). While he refers to the eigenvalues of T_k as Ritz values, he refers to the reciprocals of the eigenvalues of $P_k^{-1}T_k$ as harmonic Ritz values (see Paige *et al.*, 1995).

Despite the sophisticated mechanisms employed in his step-size computation procedure, Fletcher admits that his approach leaves unanswered the question of how to handle nonconvexity. (In the case of the two-point step-size methods described above, we say that nonpositive curvature is encountered whenever one computes $s_k^{\mathrm{T}} y_k \leqslant 0$, which can hold strictly only when the objective function is nonconvex. By contrast, in the case of an LMSD method, we say that nonpositive curvature is encountered whenever the matrix whose eigenvalues are used to compute the step sizes has a nonpositive eigenvalue. It should be noted that nonpositive eigenvalues may arise merely due to nonquadratic features in the objective function f. For ease of exposition, however, we refer to the phenomenon of having nonpositive eigenvalues as nonpositive curvature.) In his implementation, Fletcher employs a strategy that carries out a line search whenever a nonpositive step size is computed, and then terminates the sweep to effectively throw out previously computed information. By contrast, in our approach, we avoid discarding previously computed information, yet are still able to obtain reasonable step sizes.

3. Algorithm descriptions

In this section, we present our LMSD algorithm. We motivate our method by describing the development of a variant of our approach in which information from only one previous iteration is stored throughout the algorithm. We then present a generalized version of our approach that can exploit information maintained from any number of previous iterations.

3.1 *An algorithm that stores information from one previous iteration*

Suppose that an initial solution estimate x_0 (with $g_0 \neq 0$) and an initial step size $\alpha_0 > 0$ are given. Then, after k iterations, we obtain a solution estimate x_k for $k \in \mathbb{N}_+$. At this point, the calculations in the kth iteration of our algorithm are based on the cubic model $m_k : \mathbb{R}^n \to \mathbb{R}$ of f at x_k defined by

$$m_k(s) = f_k + g_k^{\mathrm{T}} s + \tfrac{1}{2} q_k \|s\|_2^2 + \tfrac{1}{6} c_k \|s\|_2^3 \approx f(x_k + s), \tag{3.1}$$

where q_k and c_k are scalars to be determined. In particular, as will be seen in this section, we choose $c_k \geqslant 0$ in such a way that m_k has a unique minimizer from the origin along its steepest descent direction $-g_k$. As such, we choose the step size α_k as an optimal solution of the one-dimensional problem

$$\min_{\alpha \geqslant 0} \phi_k(\alpha), \quad \text{where } \phi_k(\alpha) = m_k(-\alpha g_k) \quad \text{for all } \alpha \in \mathbb{R}. \tag{3.2}$$

A solution of this problem is easily obtained in the cases that will be of interest in our algorithm. In particular, if $q_k > 0$ and $c_k = 0$, then, similar to a basic BB method (recall (2.4)), we have $\alpha_k \leftarrow 1/q_k > 0$; otherwise, if $q_k \leqslant 0$ and $c_k > 0$, then it is easily verified that (3.2) is solved by setting

$$\alpha_k \leftarrow \frac{2}{q_k + \sqrt{q_k^2 + 2c_k \|g_k\|_2}} > 0. \tag{3.3}$$

We now present our strategies for setting $q_k \in \mathbb{R}$ and $c_k \geqslant 0$ for a given $k \in \mathbb{N}$. (For simplicity in the majority of our algorithm development, let us suppose that $s_k \neq 0$, $y_k \neq 0$ and $s_k^{\mathrm{T}} y_k \neq 0$; our techniques for handling cases when one or more of these conditions do not hold will be considered later.) First, consider q_k. Defining θ_k as the angle between s_k and y_k, two options for q_k come from (2.3):

$$\begin{aligned} \bar{q}_k &:= \frac{s_k^{\mathrm{T}} y_k}{s_k^{\mathrm{T}} s_k} = \cos(\theta_k) \frac{\|y_k\|_2}{\|s_k\|_2} \\ \text{and} \quad \hat{q}_k &:= \frac{y_k^{\mathrm{T}} y_k}{s_k^{\mathrm{T}} y_k} = \frac{1}{\cos(\theta_k)} \frac{\|y_k\|_2}{\|s_k\|_2}. \end{aligned} \tag{3.4}$$

Through these representations, it is clear that $|\bar{q}_k| \leqslant |\hat{q}_k|$, and hence the quantities in (2.4) satisfy $|\bar{\alpha}_k| \geqslant |\hat{\alpha}_k|$. Indeed, even though both $\bar{q}_k I$ and $\hat{q}_k I$ are valid approximations of the Hessian of f along $[x_{k-1}, x_k]$, it can be seen that $\bar{q}_k$ estimates the curvature of f only by observing the change in its gradient along the line segment $[x_{k-1}, x_k]$, whereas $\hat{q}_k$ actually accounts for changes in the gradient along an orthogonal vector as well. To see this, let

$$u_k := \left(\frac{s_k^{\mathrm{T}} y_k}{s_k^{\mathrm{T}} s_k} \right) s_k \quad \text{and} \quad v_k := y_k - u_k \quad \text{so that } y_k = u_k + v_k,$$

i.e., we define u_k to be the vector projection of y_k onto $\text{span}(s_k)$, which implies that we have $s_k^T y_k = s_k^T u_k$ and $s_k^T v_k = 0$. We then find from (2.3) that

$$\begin{aligned} \bar{q}_k &:= \frac{s_k^T y_k}{s_k^T s_k} = \frac{s_k^T u_k}{s_k^T s_k} \\ \text{and} \quad \hat{q}_k &:= \frac{y_k^T y_k}{s_k^T y_k} = \frac{u_k^T u_k + v_k^T v_k}{s_k^T u_k} = \bar{q}_k + \frac{v_k^T v_k}{s_k^T u_k}, \end{aligned} \tag{3.5}$$

where the last equation follows since $s_k^T u_k / s_k^T s_k = u_k^T u_k / s_k^T u_k$. Through these representations, it is clear that $\bar{q}_k$ is unaffected by v_k (i.e., the component of y_k orthogonal to s_k), whereas $\hat{q}_k$ takes the magnitude of this vector into account. Comparing these representations to those in (3.4) and recalling that u_k is parallel to s_k, one observes that if $v_k = 0$, then $\bar{q}_k = \hat{q}_k$, whereas if $v_k \neq 0$, then $|\bar{q}_k| < |\hat{q}_k|$. Overall, these observations provide a clearer understanding of the differing contour lines illustrated in Fig. 1.[1]

In our approach we could follow the common strategy of setting $q_k \leftarrow \bar{q}_k$ or $q_k \leftarrow \hat{q}_k$, or even choose randomly between these two options based on some probability distribution. For various reasons we choose always to set $q_k \leftarrow \hat{q}_k$. This reasoning can be explained by considering the two cases depending on the sign of the inner product $s_k^T y_k$. If $s_k^T y_k > 0$, then we set $q_k \leftarrow \hat{q}_k$ primarily due to the fact that, when $s_k^T y_k \approx 0$, this leads to smaller (i.e., more conservative) step sizes. Indeed, this will be consistent with our preference for choosing a small step size in the extreme case when $s_k^T y_k = 0$ (as explained in our later discussion of handling special cases). On the other hand, when $s_k^T y_k < 0$ and $s_k^T y_k \approx 0$, then setting $q_k \leftarrow \hat{q}_k$ corresponds to an extremely large *negative* quadratic coefficient, which has the potential to cause (3.13) to yield large step sizes. This would be inconsistent with our choice of having smaller step sizes when $s_k^T y_k > 0$ and $s_k^T y_k \approx 0$! Hence, when $s_k^T y_k < 0$, we set $q_k \leftarrow \hat{q}_k$, but will rely on a nonzero cubic term to lead the algorithm to computing a reasonable step size, as explained next.

With q_k fixed at $\hat{q}_k$, consider c_k. If $s_k^T y_k > 0$, then, as mentioned, a reasonable choice is $c_k \leftarrow 0$, since then m_k is strictly convex from the origin along $-g_k$. On the other hand, if $s_k^T y_k < 0$, then we desire an intuitive, meaningful strategy for choosing $c_k > 0$ so that problem (3.2) has a unique minimizer. We examine two possible strategies, both of which lead to a similar conclusion.

- Consider choosing c_k to minimize the least-squares error between the gradient of the model m_k at $-s_k$ (corresponding to the previous point x_{k-1}) and the previous gradient g_{k-1}, i.e.,

$$\tfrac{1}{2}\|\nabla m_k(-s_k) - g_{k-1}\|_2^2. \tag{3.6}$$

Differentiating m_k, we have for all $s \in \mathbb{R}^n$ that

$$\nabla m_k(s) = g_k + q_k s + \tfrac{1}{2} c_k \|s\|_2 s. \tag{3.7}$$

It can then easily be verified that one minimizes (3.6) by choosing

$$c_k \leftarrow \frac{2}{\|s_k\|_2}(\bar{q}_k - q_k). \tag{3.8}$$

[1] These observations have motivated our choice of notation. In particular, the 'bar' quantities are computed based on information *straight* along s_k, whereas the 'hat' quantities are computed based on information along s_k plus an orthogonal direction—a vector that can be visualized by the *bent* line in the 'hat' over the corresponding quantities.

- Consider choosing c_k so that the curvature of m_k at $-s_k$ along s_k is equal to $\bar{q}_k$, i.e., so that

$$s_k^{\mathrm{T}} \nabla^2 m_k(-s_k) s_k = \bar{q}_k \|s_k\|_2^2. \tag{3.9}$$

This is a reasonable goal since, in a BB method, it is established that $\bar{q}_k I$ is a sensible approximation of the Hessian of f along $[x_{k-1}, x_k]$, and in particular at x_{k-1} (i.e., the point corresponding to m_k evaluated at $-s_k$) along s_k. Differentiating ∇m_k (recall (3.7)), we have for all $s \in \mathbb{R}^n$ that

$$\nabla^2 m_k(s) = q_k I + \frac{1}{2} c_k \left(\|s\|_2 I + \frac{1}{\|s\|_2} s s^{\mathrm{T}} \right).$$

Hence, we obtain (3.9) by setting

$$c_k \leftarrow \frac{1}{\|s_k\|_2} \left(\bar{q}_k - q_k \right). \tag{3.10}$$

The similarity between (3.8) and (3.10) is immediately apparent, as they differ only by a constant factor. Overall, we propose that the cubic term coefficient should be set, for some constant $c > 0$, as

$$c_k \leftarrow \frac{c}{\|s_k\|_2} \left(\bar{q}_k - q_k \right). \tag{3.11}$$

Using the notation of (3.4), if $s_k^{\mathrm{T}} y_k < 0$ and $\cos(\theta_k) \neq 1$, then this formula yields $c_k > 0$. Similarly, using the notation of (3.5), if $s_k^{\mathrm{T}} u_k = s_k^{\mathrm{T}} y_k < 0$ and $v_k^{\mathrm{T}} v_k > 0$, then $c_k > 0$. Overall, one can see that we have taken the curvature that has been captured orthogonal to s_k and have used it to determine an appropriate magnitude of a cubic term so that (3.2) has a unique minimizer. A relatively large discrepancy between $\bar{q}_k$ and $q_k = \hat{q}_k$ indicates a relatively large displacement in the gradient orthogonal to s_k, which in turn suggests that a relatively large cubic term coefficient should be used to safeguard the next step size. One may also observe that (3.10) is particularly appealing in the sense that it represents a finite-difference approximation of a third-order (directional) derivative using the difference between the two available second-order (directional) derivative estimates for the interval $[x_{k-1}, x_k]$.

We are almost prepared to present a complete description of our algorithm (for $m = 1$), but first we must remark on the special cases that we have ignored until this point. That is, we must specify how the algorithm is to proceed when $s_k = 0$, $y_k = 0$, $s_k^{\mathrm{T}} y_k = 0$ or $s_k^{\mathrm{T}} y_k < 0$ while s_k and y_k are parallel. In fact, as long as the algorithm terminates in any iteration $k \in \mathbb{N}$ for which $g_k = 0$, and otherwise computes a step size $\alpha_k > 0$, the algorithm cannot produce $s_k = 0$. Hence, we need consider only the computation of step sizes when $s_k \neq 0$, so the only special cases that remain to be considered are as follows.

- If $y_k = 0$, then the step from x_{k-1} to x_k has yielded a zero displacement in the gradient of f. Consequently, between the points x_{k-1} and x_k, we have no useful information to approximate the Hessian of f; in fact, based on the relationship between gradients at these points, f 'appears affine' at x_k along $-g_k = -g_{k-1}$. In such cases, we set α_k to a maximum allowable step size, call it $\Omega > 0$, in an attempt to aggressively minimize f along the steepest descent direction $-g_k$.

- If $y_k \neq 0$, but $s_k^{\mathrm{T}} y_k = 0$, then the displacement from x_{k-1} to x_k has yielded a nonzero displacement in the gradient of f, but this displacement is orthogonal to $s_k = x_k - x_{k-1}$. Hence, as there has been no displacement of the gradient in a direction parallel to the displacement in the iterate, a two-point step-size approximation of the Hessian of f at x_k is inadequate. Thus, since the next iteration will involve exploring f along $-g_k \neq -g_{k-1}$, in this 'new' direction we conservatively set α_k to a minimum allowable step size, call it $\omega > 0$ (with $\omega \leqslant \Omega$).

- If $y_k \neq 0$, $s_k^{\mathrm{T}} y_k < 0$, and s_k and y_k are parallel, then the displacement from x_{k-1} to x_k has yielded a nonzero displacement in the gradient of f only in the direction of s_k. Consequently, similar to cases when $y_k = 0$, we have no useful information to approximate the Hessian of f in any direction other than s_k; in fact, based on the relationship between the gradients at these points, f 'appears affine' at x_k along any direction other than s_k. In such cases, since $-g_k$ is parallel to s_k, we set α_k to the large step size $\Omega > 0$ to try to aggressively minimize f along $-g_k$.

We are now prepared to provide a complete description of our first approach, given as Algorithm 1. Along with the safeguards employed in the special cases discussed above, we employ the universal safeguard of projecting any computed step size onto the interval $[\omega, \Omega]$. For simplicity in our description, we omit mention of the computation of function and gradient values, as well as of the displacement vectors in (2.2); these are implied whenever a new iterate is computed. Furthermore, we suppress any mention of a termination condition, but remark that any practical implementation of our algorithm would terminate as soon as a gradient is computed that has a norm that is approximately zero. Hence, in the algorithm, we assume for all practical purposes that $g_k \neq 0$ for all $k \in \mathbb{N}$.

Algorithm 1 LMSD Method with Cubic Regularization (for $m = 1$)

choose $(\omega, \Omega) \in \mathbb{R} \times \mathbb{R}$ satisfying $0 < \omega \leqslant \Omega$ and $c \in \mathbb{R}_+ := \{c \in \mathbb{R} : c > 0\}$
choose $x_0 \in \mathbb{R}^n$ and $\alpha_0 \in [\omega, \Omega]$
set $x_1 \leftarrow x_0 - \alpha_0 g_0$ and $k \leftarrow 1$
loop
    if $y_k = 0$ or $s_k^{\mathrm{T}} y_k = -\|s_k\|_2 \|y_k\|_2 < 0$ **then**
        set $\alpha_k \leftarrow \Omega$
    else if $s_k^{\mathrm{T}} y_k = 0$ **then**
        set $\alpha_k \leftarrow \omega$
    else
        set $\bar{q}_k \leftarrow s_k^{\mathrm{T}} y_k / s_k^{\mathrm{T}} s_k$ and $q_k \leftarrow y_k^{\mathrm{T}} y_k / s_k^{\mathrm{T}} y_k$
        if $q_k > 0$ **then** set $c_k \leftarrow 0$ **else** set $c_k \leftarrow c(\bar{q}_k - q_k)/\|s_k\|_2$
        if $q_k > 0$ **then** set $\alpha_k \leftarrow 1/q_k$ **else** set $\alpha_k \leftarrow 2/\left(q_k + \sqrt{q_k^2 + 2c_k\|g_k\|_2}\right)$
        replace α_k by its projection onto the interval $[\omega, \Omega]$
    set $x_{k+1} \leftarrow x_k - \alpha_k g_k$ and $k \leftarrow k + 1$
end loop

We close this section by providing an illustration of the types of step sizes computed in Algorithm 1, which may be compared with those illustrated in Fig. 1. Using the same set-up as for Fig. 1 and with $c \leftarrow 1$, we plot in Fig. 2 the step sizes computed via Algorithm 1 as a function of the gradient $g_k \in [-3, 1] \times [-2, 2]$. In this plot, it is clear that, when $s_k^{\mathrm{T}} y_k > 0$, Algorithm 1 computes step sizes that are equal to those in the plot on the right in Fig. 1. More important, however, is that Algorithm 1 computes reasonable step sizes that exploit problem information even when $s_k^{\mathrm{T}} y_k < 0$. In particular, when $s_k^{\mathrm{T}} y_k < 0$ and $s_k^{\mathrm{T}} y_k \approx 0$, the algorithm computes step sizes consistent with those computed in 'nearby cases' when $s_k^{\mathrm{T}} y_k > 0$. On the other hand, if $s_k^{\mathrm{T}} y_k < 0$ while s_k and y_k are (nearly) parallel, then the algorithm computes very large step sizes, which has been motivated in the third special case above.

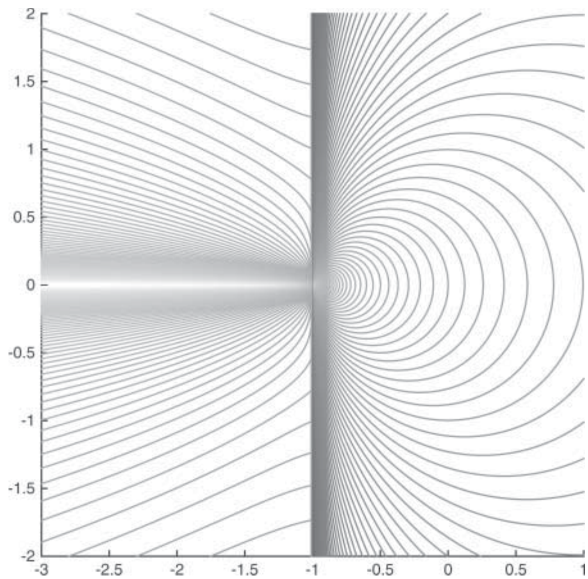

FIG. 2. Illustration of the step sizes computed, as a function of the current gradient g_k, by Algorithm 1.

3.2 *An algorithm that stores information from $m \geqslant 1$ previous iteration(s)*

We have presented our algorithm for $m = 1$ as one that, at x_k, computes a step size based on minimizing a cubic model of the objective from x_k along the steepest descent direction $-g_k$. On the other hand, in Section 2, we described Fletcher's LMSD method for $m \geqslant 1$ as one that, at x_k, computes m step sizes to be employed in the next m iterations by computing (reciprocals of) the eigenvalues of an $m \times m$ matrix. On the surface, these approaches are quite different. Therefore, in order to extend our approach to cases when $m \geqslant 1$ and have it reduce to (a variant of) Fletcher's LMSD method, we must explain how Fletcher's step-size computation procedure can be understood in terms of minimizing a local model of f at x_k. This can be done with the use of a quadratic model, but for our purposes of generalizing the approach, we employ a cubic model (and refer to his approach as one in which the cubic term is zero).

For a given $j \in \{0, \ldots, m-1\}$, consider the cubic model $m_{k+j} : \mathbb{R}^n \to \mathbb{R}$ of f at x_{k+j} defined by

$$m_{k+j}(s) = f_{k+j} + g_{k+j}^{\mathrm{T}} s + \tfrac{1}{2} q_{k+j} \|s\|_2^2 + \tfrac{1}{6} c_{k+j} \|s\|_2^3 \approx f(x_{k+j} + s), \tag{3.12}$$

where q_{k+j} and c_{k+j} are scalars to be determined. Furthermore, consider the problem

$$\min_{\alpha \geqslant 0} \phi_{k+j}(\alpha), \quad \text{where } \phi_{k+j}(\alpha) := m_{k+j}(-\alpha g_{k+j}) \quad \text{for all } \alpha \in \mathbb{R}. \tag{3.13}$$

It is easily seen that when $q_{k+j} > 0$ and $c_{k+j} = 0$, the solution of problem (3.13) is given by $\alpha_{k+j} = q_{k+j}^{-1}$, while if $q_{k+j} \leqslant 0$ and $c_{k+j} > 0$, then the solution is given by a formula similar to (3.3).

Supposing that iteration k represents the beginning of a 'sweep', Fletcher's approach can be understood in terms of minimizing the local model of f at x_{k+j} given by (3.12) *simultaneously* for all $j \in \{0, \ldots, m-1\}$, despite the fact that, for $j > 0$, the point x_{k+j} and gradient g_{k+j} are unknown at the beginning of iteration k. In particular, his approach involves the construction of an $m \times m$ matrix, call it M_k, based on information obtained from the m iterations prior to, and including, iteration k. He computes the eigenvalues of M_k, composing $\{q_k, \ldots, q_{k+m-1}\}$, the reciprocals of which, composing $\{\alpha_k, \ldots, \alpha_{k+m-1}\}$, are to be used as the step sizes in the next m iterations. In fact, Fletcher orders these step sizes from smallest to largest before employing them in the algorithm.

Our approach proceeds in a similar manner, but with some notable differences. Specifically, at the beginning of a 'sweep' occurring from iteration k, we compute scalars that may be used for q_{k+j} for all $j \in \{0, \ldots, m-1\}$. However, we do not simply use the reciprocals of these values as the step sizes in the subsequent m iterations, especially since some or all of these values may be negative. Instead, we iterate in the usual manner through iterations $k+j$ for all $j \in \{0, \ldots, m-1\}$, where, for each such j, we compute $c_{k+j} \geqslant 0$ such that m_{k+j} is bounded below over $-g_{k+j}$ and (3.13) yields a unique minimizer.

For computing the quadratic-term coefficients, we follow the approach of Fletcher and assume that, at the beginning of iteration k, we have available an invertible symmetric tridiagonal matrix $\tilde{T}_k \in \mathbb{R}^{m\times m}$ and an invertible symmetric pentadiagonal matrix $\tilde{P}_k \in \mathbb{R}^{m\times m}$. (It is possible that our formulas for these matrices, provided later, yield matrices that are not invertible, but we handle these as special cases later on.) We then define the following sets, each ordered from largest to smallest:

$$\begin{aligned} &\{\bar{q}_k, \ldots, \bar{q}_{k+m-1}\} \text{ as the eigenvalues of } \tilde{T}_k \\ \text{and} \quad &\{\hat{q}_k, \ldots, \hat{q}_{k+m-1}\} \text{ as the eigenvalues of } (\tilde{P}_k^{-1}\tilde{T}_k)^{-1}. \end{aligned} \tag{3.14}$$

At iteration $k+j$ for $j \in \{0, \ldots, m-1\}$, we follow the strategy of the $m=1$ case and set $q_{k+j} \leftarrow \hat{q}_{k+j}$. If $\hat{q}_{k+j} > 0$, then we set $c_{k+j} \leftarrow 0$, but otherwise we again follow the strategy of the $m=1$ case and set

$$c_{k+j} \leftarrow \frac{c}{\|s_{k+j}\|_2}(\bar{q}_{k+j} - q_{k+j}). \tag{3.15}$$

Observe that when $m=1$, the approach described in the previous paragraph reduces to that in Section 3.1, and in that case we have $\bar{q}_{k+j} - q_{k+j} > 0$ for $j=0$ whenever $s_{k+j}^{\mathrm{T}} y_{k+j} < 0$ and y_{k+j} is not parallel to s_{k+j}. However, for $m > 1$, we must ensure that (3.15) yields $c_{k+j} \geqslant 0$. In fact, this is guaranteed by our approach for constructing $\tilde{T}_k$ and $\tilde{P}_k$, but, before describing this construction, we remark the following points.

- If $m=1$, $\tilde{T}_k = \bar{q}_k$ and $\tilde{P}_k = \|y_k\|_2^2/\|s_k\|_2^2$, then $(\tilde{P}_k^{-1}\tilde{T}_k)^{-1} = \hat{q}_k$ and (3.15) reduces to (3.11).
- If $m \geqslant 1$ and $f(x) = \frac{1}{2}x^{\mathrm{T}}Ax$ for some $A \succ 0$, then with $\tilde{T}_k \leftarrow T_k$ from (2.6) and $\tilde{P}_k \leftarrow P_k$ from (2.8), we have that $\tilde{T}_k$ is symmetric tridiagonal and $\tilde{P}_k$ is symmetric pentadiagonal. Furthermore, in this case, we have $T_k \succ 0$ and $(P_k^{-1}T_k)^{-1} \succ 0$, meaning that our approach for computing step sizes reduces to Fletcher's method when harmonic Ritz values are employed. On the other hand, when $\tilde{T}_k$ and $\tilde{P}_k$ are set in the same manner but $A \not\succ 0$, the eigenvalues of $\tilde{T}_k$ and $(\tilde{P}_k^{-1}\tilde{T}_k)^{-1}$ will be interlaced. In such a case, (3.15) involves $\bar{q}_{k+j} - q_{k+j} \geqslant 0$ for any $j \in \{0, \ldots, m-1\}$ with $\bar{q}_{k+j} < 0$.

A critical feature of our algorithm is how we choose $\tilde{T}_k$ and $\tilde{P}_k$ when f is not quadratic. In Fletcher's work, he remarks that, in the nonquadratic case, the matrix T_k in (2.6) will be upper Hessenberg, but not necessarily tridiagonal. He handles this by constructing $\tilde{T}_k$, which is set to T_k except that its strict upper triangle is replaced by the transpose of its strict lower triangle, thus creating a tridiagonal matrix.

We employ the same strategy in our algorithm. Then, observing from (2.8) that

$$P_k = T_k^{\mathrm{T}} T_k + \zeta_k \zeta_k^{\mathrm{T}}, \quad \text{where } \zeta_k^{\mathrm{T}} := [0 \quad \rho_k] J_k R_k^{-1}, \tag{3.16}$$

we set $\tilde{P}_k$ as $\tilde{T}_k^{\mathrm{T}} \tilde{T}_k + \zeta_k \zeta_k^{\mathrm{T}}$, i.e., we use the same expression as in (3.16), but with T_k replaced by $\tilde{T}_k$.

The strategy described in the previous paragraph is well defined if T_k in (2.6) is well defined, and (3.15) ensures $c_{k+j} \geqslant 0$ for all $k \in \mathbb{N}_+$ and $j \in \{0, \ldots, m-1\}$ if the eigenvalues of $\tilde{T}_k$ and $(\tilde{P}_k^{-1} \tilde{T}_k)^{-1}$ are interlaced. For these cases, we provide the following theorems with proofs in Appendix A.

THEOREM 3.1 The matrix T_k in (2.6) is well defined if and only if the columns of G_k are linearly independent. Furthermore, with $\alpha_k > 0$ for all $k \in \mathbb{N}$, the matrix T_k is invertible if and only if the columns of G_k are linearly independent and the elements of the vector $R_k^{-1} Q_k^{\mathrm{T}} g_k$ do not sum to 1.

THEOREM 3.2 Suppose that T_k in (2.6) is well defined. Then, let $\tilde{T}_k$ be set equal to T_k, except that its strict upper triangle is replaced by the transpose of its strict lower triangle, and let $\tilde{P}_k \leftarrow \tilde{T}_k^{\mathrm{T}} \tilde{T}_k + \zeta_k \zeta_k^{\mathrm{T}}$ where ζ_k is defined in (3.16). Then, $\tilde{T}_k$ is symmetric tridiagonal and $\tilde{P}_k$ is symmetric pentadiagonal.

THEOREM 3.3 Suppose that T_k in (2.6) is well defined and that the matrices $\tilde{T}_k$ and $\tilde{P}_k$, constructed as described in Theorem 3.2, are invertible. Then, the eigenvalues in (3.14) satisfy

$$|\bar{q}_{k+j}| \leqslant |\hat{q}_{k+j}| \quad \text{for all } j \in \{0, \ldots, m-1\}.$$

In particular, if for some $j \in \{0, \ldots, m-1\}$ one has $\hat{q}_{k+j} < 0$, then (3.15) yields $c_{k+j} \geqslant 0$.

Overall, like Fletcher's method, our strategy reduces to using Ritz and harmonic Ritz values in the quadratic case, and otherwise manipulates the matrices in (2.6) and (2.8) to obtain matrices with similar structure as would be obtained automatically in the quadratic case.

We are almost prepared to discuss our main algorithm, but first we must discuss the special cases that must be considered for our algorithm to be well defined.

- Suppose that the columns of G_k are linearly dependent, or the columns of G_k are linearly independent while the elements of $R_k^{-1} Q_k^{\mathrm{T}} g_k$ sum to 1. In such cases, T_k is not well defined, so our desired procedure for constructing $\tilde{T}_k$ and $\tilde{P}_k$ also is not well defined. To handle this, we iteratively consider fewer previous gradients (where, at each stage, the eldest member is ignored in favour of newer gradients) until the set of considered previous gradients consists of linearly independent vectors for which $R_k^{-1} Q_k^{\mathrm{T}} g_k$ has elements that do not sum to 1. (Note that the situation in which G_k has linearly dependent columns does not need to be considered when $m = 1$ since, in that case, $G_k = [g_{k-1}]$ having a linearly dependent column corresponds to $g_{k-1} = 0$, in which case Algorithm 1 would have terminated in iteration $k-1$. Also note that, in the $m = 1$ case, having $1 = R_k^{-1} Q_k^{\mathrm{T}} g_k$ corresponds to the special case of having $y_k = 0$.) In the extreme case when the set of previously computed gradients is reduced to only the most recently computed previous gradient, we compute a step size as in Algorithm 1.
- Suppose that T_k is well defined, but for some $q_{k+j} \leqslant 0$ we obtain $c_{k+j} = 0$. This is similar to the last of the special cases considered when $m = 1$, and so, as in that case, we aggressively minimize f by computing a step size as $\Omega > 0$.

Our main approach is presented as Algorithm 2. As for Algorithm 1, we suppress mention of the computation of function values, gradient values and displacement vectors, and suppress mention of termination checks throughout the algorithm. Correspondingly, we assume for all practical purposes that

Algorithm 2 LMSD Method with Cubic Regularization (for $m \geqslant 1$)

choose $(\omega, \Omega) \in \mathbb{R} \times \mathbb{R}$ satisfying $0 < \omega \leqslant \Omega$, $c \in \mathbb{R}_+$ and $m \in \mathbb{N}_+$
choose $x_0 \in \mathbb{R}^n$ and $\alpha_j \in [\omega, \Omega]$ for all $j \in \{0, \ldots, m-1\}$
for $j = 0, \ldots, m-1$ **do**
    set $x_{j+1} \leftarrow x_j - \alpha_j g_j$
end for
set $k \leftarrow m$
loop
    loop
        set G_k as in (2.5) and J_k as in (2.7)
        compute the (thin) QR-factorization $G_k = Q_k R_k$
        if G_k is composed of only one column **then break**
        if R_k is invertible and the elements of $R_k^{-1} Q_k^{\mathrm{T}} g_k$ do not sum to 1 **then break**
        remove the first column each of G_k and J_k
    end loop
    set $\tilde{m}$ as the number of columns of G_k
    if $\tilde{m} = 1$ **then**
        set α_k as in Algorithm 1 and $x_{k+1} \leftarrow x_k - \alpha_k g_k$
    else
        set T_k as in (2.6) and set $\tilde{T}_k$ and $\tilde{P}_k$ as described in Theorem 3.2
        set $\{\bar{q}_k, \ldots, \bar{q}_{k+\tilde{m}-1}\}$ and $\{\hat{q}_k, \ldots, \hat{q}_{k+\tilde{m}-1}\}$ as in (3.14)
        for $j = 0, \ldots, \tilde{m}-1$ **do**
            set $q_{k+j} \leftarrow \hat{q}_{k+j}$
            if $q_{k+j} > 0$ **then** set $c_{k+j} \leftarrow 0$ **else** set $c_{k+j} \leftarrow c(\bar{q}_{k+j} - q_{k+j})/\|s_{k+j}\|_2$
            if $q_{k+j} > 0$ **then** set $\alpha_{k+j} \leftarrow 1/q_{k+j}$
            else if $c_{k+j} > 0$ **then** set $\alpha_{k+j} \leftarrow 2/\left(q_{k+j} + \sqrt{q_{k+j}^2 + 2c_{k+j}\|g_{k+j}\|_2}\right)$
            else if $\bar{q}_{k+j} = 0$ **then** set $\alpha_{k+j} \leftarrow \omega$
            else set $\alpha_{k+j} \leftarrow \Omega$
            set $x_{k+j+1} \leftarrow x_{k+j} - \alpha_{k+j} g_{k+j}$
        end for
    end if
    set $k \leftarrow k + \tilde{m}$
end loop

$g_k \neq 0$ for all $k \in \mathbb{N}$. Also for simplicity in its description, we state Algorithm 2 in such a way that it iterates *in order* through the sets of eigenvalues in (3.14). Note, however, that during a given iteration k, one may also consider computing all step sizes that would be obtained by any of the available pair of eigenvalues, and then choosing the pair that leads to the smallest corresponding step size. This would be consistent with Fletcher's approach in that, at each point in a 'sweep', the eigenvalue yielding the smallest step size is chosen. For this reason, this is the strategy that we have adopted in our numerical experiments. (Note that by ordering the eigenvalues from largest to smallest, employing them in order corresponds to choosing step sizes in increasing order of magnitude *if* all of the eigenvalues are positive.)

In our implementation described in the following section, we incorporated a nonmonotone line search into Algorithm 2, which, for ease of exposition, has also been suppressed in the presentation of the algorithm. In particular, we incorporated the Armijo (i.e., backtracking) line search of Zhang & Hager (2004) with moving average parameter $\eta_k = \eta \in (0,1)$ for all $k \in \mathbb{N}$, which is employed in every step of the algorithm. With this line search, we claim the following global convergence result for our algorithm; the proof is a special case of Zhang & Hager (2004, Theorem 2.2).

THEOREM 3.4 Suppose that f is bounded below and ∇f is Lipschitz continuous on

$$\{x \in \mathbb{R}^n : f(x) \leqslant f(x_0)\} + \left\{ d \in \mathbb{R}^n : \|d\|_2 \geqslant \sup_{k \in \mathbb{N}} \|\alpha_k g_k\|_2 \right\}. \tag{3.17}$$

Then, the iterate sequence $\{x_k\}$ generated by Algorithm 2 yields

$$\lim_{k \to \infty} g_k = 0.$$

That is, every convergent subsequence of $\{x_k\}$ approaches a point x_* with $\nabla f(x_*) = 0$.

4. Implementation

Algorithm 2 (which includes Algorithm 1 as a special case) was implemented in MATLAB along with two other algorithms for comparison purposes. In this section, we describe the implementations of these algorithms along with input parameter settings that were used in our experiments (see Section 5).

We used built-in MATLAB functions to compute matrix factorizations and eigenvalues in all of the implemented algorithms. In order to avoid the influence of numerical error and the computation of excessively small or large step sizes, we removed previously computed gradients (in a similar manner as in the inner **loop** of Algorithm 2) if more than one was currently being held and any of the corresponding computed eigenvalues were smaller than 10^{-12} or larger than 10^{12} in absolute value. Similarly, prior to computing corresponding step sizes, we projected any computed quadratic term coefficient so that it would have an absolute value at least 10^{-12} and at most 10^{12}, and we projected any computed cubic-term coefficient onto the widest possible subset of the positive reals so that the resulting step size would be at least 10^{-12} and at most 10^{12}. (For the quadratic-term coefficients, this projection was performed to maintain the sign of the originally computed coefficient.) Overall, this implied $\omega = 10^{-12}$ and $\Omega = 10^{12}$ for all algorithms. As described in Section 3, the eigenvalues, once computed, were ordered from largest to smallest, though the implementation of Algorithm 2 potentially used these eigenvalues out of order in order to ensure that, in any given iteration, the eigenvalue pair leading to the smallest step size was used (after which this pair was removed from the set for subsequent iterations).

The three implemented algorithms differed only in the manner in which step sizes were computed. Our implementation of Algorithm 2—hereafter referred to as CUBIC—employed the strategy described in Section 3 with $c \leftarrow 1$. The other two algorithms, on the other hand, were two variants of an algorithm derived from the ideas in Fletcher (2012). In particular, the algorithm we refer to as QUAD-RITZ computes step sizes as reciprocals of the Ritz values $\{\bar{q}_k, \ldots, \bar{q}_{k+m-1}\}$, whereas the algorithm we refer to as QUAD-HRITZ computes step sizes as reciprocals of the harmonic Ritz values $\{\hat{q}_k, \ldots, \hat{q}_{k+m-1}\}$ (recall (3.14)). In both QUAD-RITZ and QUAD-HRITZ, the standard approach of handling nonpositive curvature was employed; i.e., if a computed eigenvalue was negative, then the step size was simply set to Ω.

For simplicity and consistency, all algorithms employed the same nonmonotone line search in every step, using the step size computed in the main procedure as the initial step size for the line search.

As mentioned in Section 3, for this purpose, we implemented the Armijo (i.e., backtracking) line search of Zhang & Hager (2004). This strategy requires a sufficient decrease parameter, for which we used 10^{-12}, a backtracking parameter, for which we used 0.5, and a moving average parameter (see η_k as defined in Zhang & Hager, 2004), for which we used 0.5 for all $k \in \mathbb{N}$. For the history length parameter m, we experimented with values in the set $\{1, 3, 5\}$. (As discussed further in Section 5, results for larger values of m did not lead to improved performance beyond the values considered here. This is consistent with Fletcher's experience with his LMSD method, and in some previous studies of L-BFGS.) All algorithms terminated whenever either the ℓ_∞-norm of a computed gradient was less than or equal to $10^{-8} \max\{1, \|g_0\|_\infty\}$—indicating a successful run—or the maximum iteration count of 10^{10} was reached—indicating a failure.

5. Numerical experiments

We tested the algorithms CUBIC, QUAD-RITZ and QUAD-HRITZ by employing them to solve unconstrained problems from the CUTEst collection; see Gould *et al.* (2013) (and Gould *et al.*, 2003 for information about a previous version of the test set). We resized all unconstrained problems to the largest of their preset sizes, and from that set kept those (a) that had at least 50 variables, so as to have $m \ll n$; (b) for which at least one run of an algorithm required at least 5 s, so as to focus on the more difficult problems in the set; (c) for which at least one run of an algorithm involved the computation of a nonpositive quadratic-term coefficient, so as to focus on the issue of handling nonpositive curvature; and (d) for which at least one run of an algorithm led to a successful termination. The resulting set of 30 problems and their sizes can be found in the tables of results provided in Appendix B.

Over all runs of all algorithms on all of our test problems, we compiled the number of function and gradient evaluations required prior to termination. (Note that the number of iterations can be considered equal to the number of gradient evaluations.) To compare the results of the experiments, we consider the technique proposed in Morales (2002), which proceeds as follows. Letting func_A^j and func_B^j be the number of function evaluations required by algorithms A and B, respectively, on the jth problem in the set, we compute the logarithmic outperforming factor $r_{AB}^j := -\log_2(\text{func}_A^j/\text{func}_B^j)$. For example, $r_{AB}^j = 2$ indicates that algorithm A required 2^{-2} of the function evaluations required by algorithm B. Such a factor is similarly defined for comparing the numbers of gradient evaluations required by two algorithms. By computing all such factors for each problem in the test set, one can compare the performance of two algorithms side by side in a bar plot, where positive bars indicate better performance with respect to a particular measure for algorithm A and negative bars indicate better performance for algorithm B. In all cases, we restrict attention to $r_{AB}^j \in [-1, 1]$ since the particular magnitude of a factor beyond this interval is not of great interest; it is sufficient to know that an algorithm performed fewer than half (or more than double) the number of function or gradient evaluations as did another algorithm.

As a first comparison, we consider the performance of the algorithms for $m = 1$, for which we have the bar plots in Figs 3 and 4, corresponding to function and gradient evaluations, respectively. In each figure, the performance of each pair of algorithms is revealed in a side-by-side comparison with the name of algorithm 'A' indicated above the plot and the name of algorithm 'B' indicated below the plot. The profiles clearly indicate that the use of harmonic Ritz values led to better performance than the use of Ritz values in that CUBIC and QUAD-HRITZ consistently outperformed QUAD-RITZ with respect to both measures on most (if not all) problems in our test set.

One observes in Figs 3 and 4 that CUBIC and QUAD-HRITZ performed similarly on many problems, with better performance provided by QUAD-HRITZ on some problems in the set. This situation changed, however, when we considered $m = 3$ and $m = 5$, for which we have the bar plots in Figs 5–8.

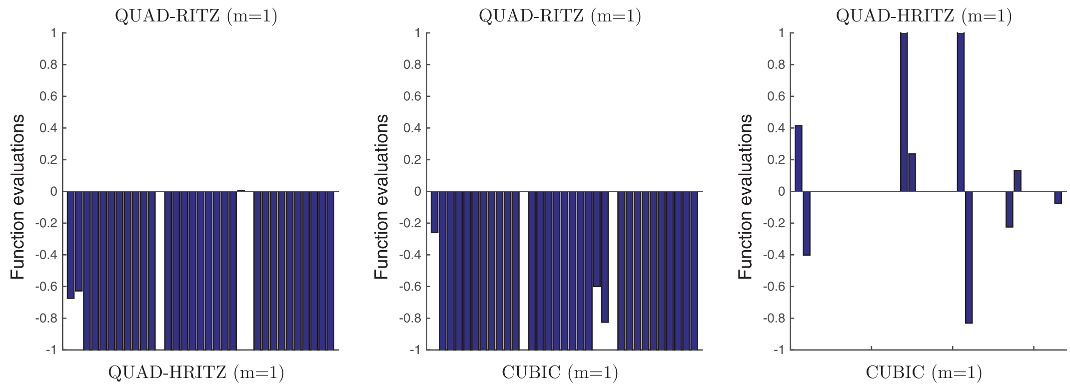

FIG. 3. Outperforming factors for function evaluations with $m = 1$.

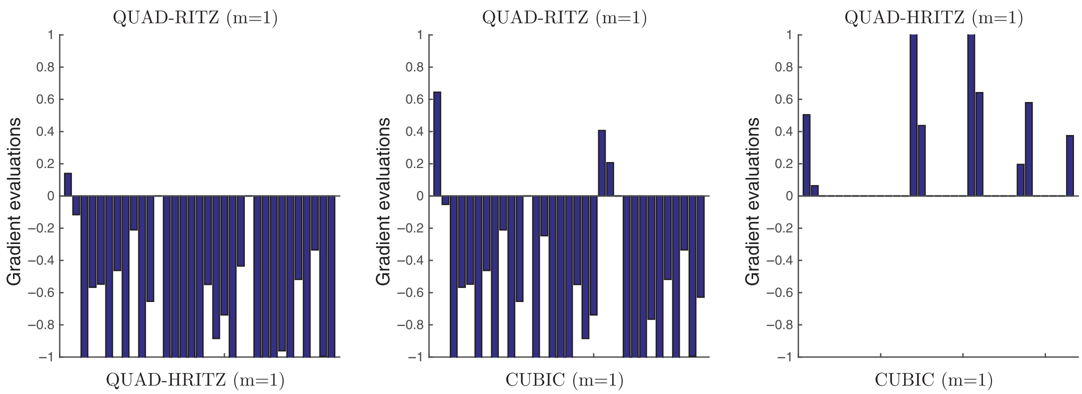

FIG. 4. Outperforming factors for gradient evaluations with $m = 1$.

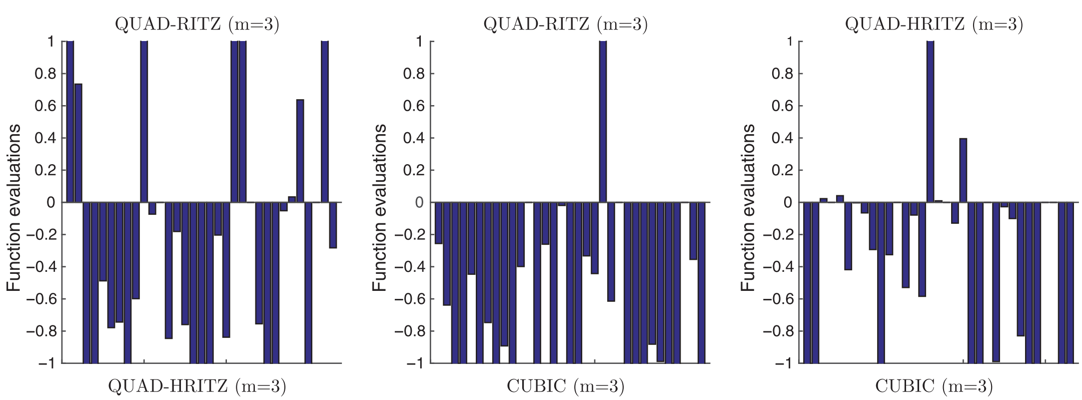

FIG. 5. Outperforming factors for function evaluations with $m = 3$.

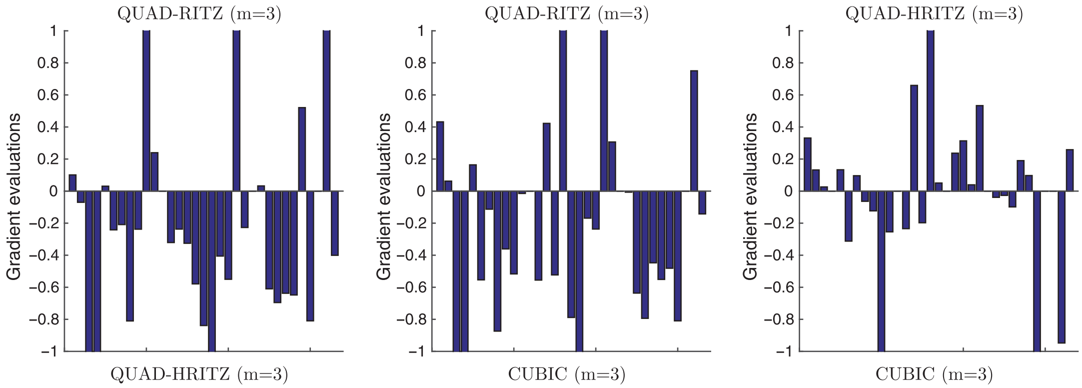

FIG. 6. Outperforming factors for gradient evaluations with $m = 3$.

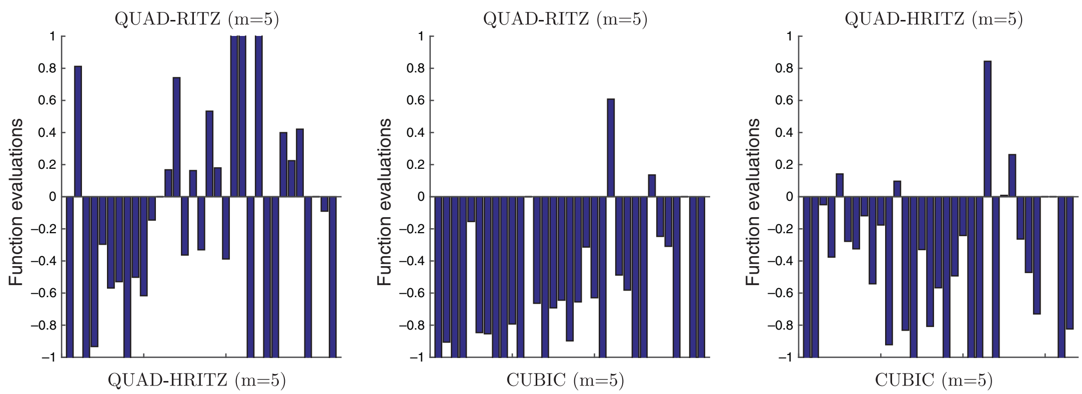

FIG. 7. Outperforming factors for function evaluations with $m = 5$.

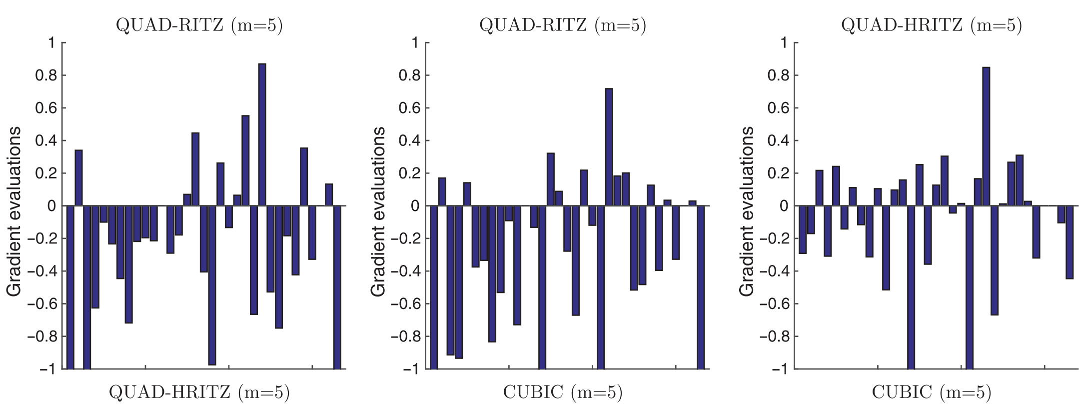

FIG. 8. Outperforming factors for gradient evaluations with $m = 5$.

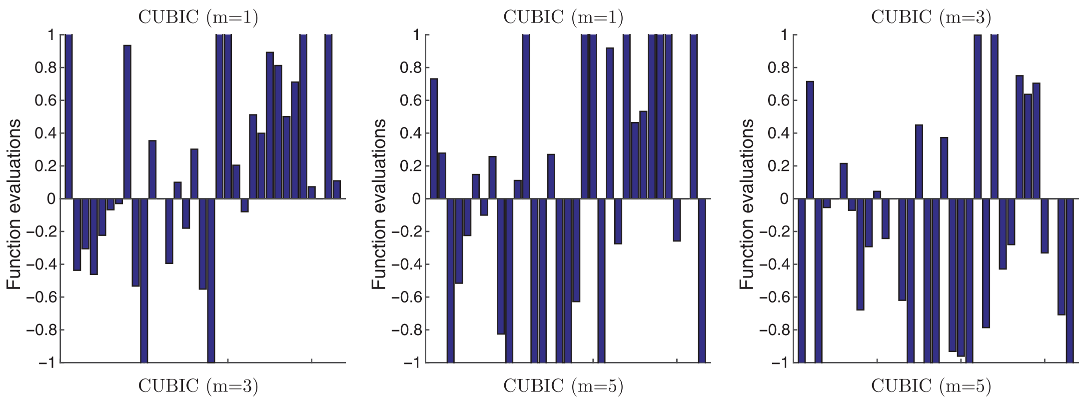

FIG. 9. Outperforming factors for function evaluations for CUBIC for $m \in \{1, 3, 5\}$.

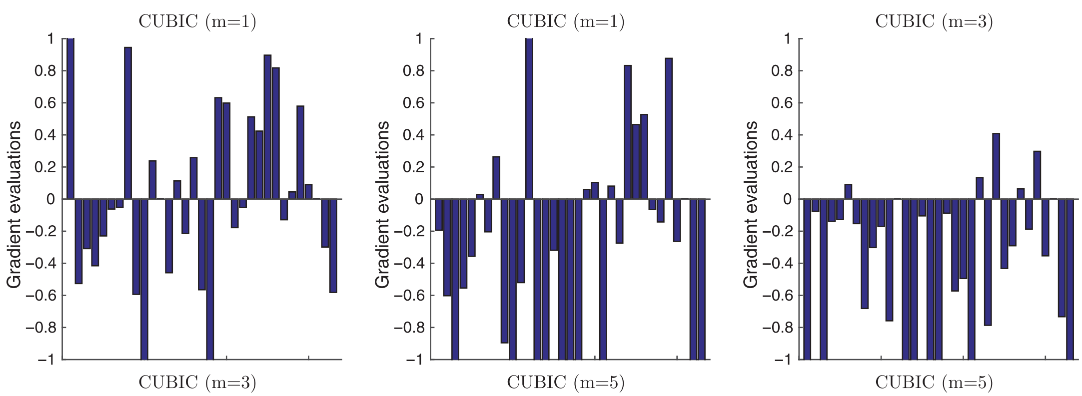

FIG. 10. Outperforming factors for gradient evaluations for CUBIC for $m \in \{1, 3, 5\}$.

Particularly in terms of function evaluations and often in terms of gradient evaluations, CUBIC outperformed both QUAD-RITZ and QUAD-HRITZ on a majority of problems in our set.

As a final comparison, we consider the performance of CUBIC for $m \in \{1, 3, 5\}$. The results are provided in the bar plots in Figs 9 and 10. Despite mixed results in terms of function evaluations, the factors computed based on gradient evaluations indicate better performance when $m = 5$. Combining this conclusion with those drawn from the results above, we claim that we obtained the best results in our experiments with CUBIC and $m = 5$. (We also experimented with larger values of m, but they did not lead to improved performance over those obtained with $m = 5$. We suspect that this was due to the presence of nonpositive curvature—i.e., nonpositive eigenvalues—much more often than would have been observed with a smaller value of m. This led us to conclude that while one may benefit by confronting a modest amount of nonpositive curvature with our proposed technique, an excessive amount of nonpositive curvature is not easily overcome.)

6. Conclusion

We have designed, analysed and experimented with an LMSD method for solving unconstrained nonconvex optimization problems. The unique feature of our algorithm is a novel approach for handling nonpositive curvature; in particular, we propose that when nonpositive curvature is encountered, step sizes can be computed by constructing local cubic models of the objective function, for which reasonable values of the quadratic- and cubic-term coefficients can be derived using previously computed gradient information. Our numerical experiments suggest that our approach yields superior performance in practice compared to algorithms that do not attempt to incorporate problem information when computing step sizes when nonpositive curvature is encountered.

Acknowledgements

The authors would like to thank the Associate Editor and the two anonymous referees whose comments and suggestions improved the paper, especially in terms of the aims and scope of our numerical results.

Funding

U.S. Department of Energy, Office of Science, Office of Advanced Scientific Computing Research, Applied Mathematics, Early Career Research Program (contract number DE–SC0010615); U.S. National Science Foundation (Grant No. DMS-1319356).

References

Barzilai, J. & Borwein, J. M. (1988) Two-point step size gradient methods. *IMA J. Numer. Anal.*, **8**, 141–148.

Biglari, M. & Solimanpur, M. (2013) Scaling on the spectral gradient method. *J. Optim. Theory Appl.*, **158**, 626–635.

Broyden, C. G. (1970) The convergence of a class of double-rank minimization algorithms. *J. Inst. Math. Appl.*, **6**, 76–90.

Cauchy, A. (1847) Méthode générale pour la résolution des systèms d'equations simultanées. *C. R. Séances Acad. Sci.*, **25**, 536–538.

Dai, Y.-H. & Liao, L.-Z. (2002) *R*-linear convergence of the Barzilai and Borwein gradient method. *IMA J. Numer. Anal.*, **22**, 1–10.

Dai, Y., Yuan, J. & Yuan, Y.-X. (2002) Modified two-point step size gradient methods for unconstrained optimization. *Comput. Optim. Appl.*, **22**, 103–109.

De Asmundis, R., Di Serafino, D., Hager, W. W., Toraldo, G. & Zhang, H. (2014) An efficient gradient method using the Yuan steplength. *Comput. Optim. Appl.*, **59**, 541–563.

De Asmundis, R., Di Serafino, D., Riccio, F. & Toraldo, G. (2013) On spectral properties of steepest descent methods. *IMA J. Numer. Anal.*, **33**, 1416–1435.

Fletcher, R. (1970) A new approach to variable metric algorithms. *Comput. J.*, **13**, 317–322.

Fletcher, R. (2012) A limited memory steepest descent method. *Math. Program.*, **135**, 413–436.

Goldfarb, D. (1970) A family of variable metric updates derived by variational means. *Math. Comput.*, **24**, 23–26.

Golub, G. H. & Van Loan, C. F. (1996) *Matrix Computations*, 3rd edn. Johns Hopkins University Press.

Gould, N. I. M., Orban, D. & Toint, Ph. L. (2003) CUTEr and SifDec: a constrained and unconstrained testing environment, revisited. *ACM Trans. Math. Softw.*, **29**, 373–394.

Gould, N. I. M., Orban, D. & Toint, Ph. L. (2013) CUTEst: A Constrained and Unconstrained Testing Environment with Safe Threads. *Technical Report RAL-TR-2013-005*. Rutherford-Appleton Laboratory.

Grippo, L., Lampariello, F. & Lucidi, S. (1986) A nonmonotone line search technique. *SIAM J. Numer. Anal.*, **23**, 707–716.

Kafaki, S. B. & Fatemi, M. (2013) A modified two-point step size gradient algorithm for unconstrained minimization. *Optim. Methods Softw.*, **28**, 1040–1050.

Li, D.-H. & Fukushima, M. (2001) A modified BFGS method and its global convergence in nonconvex minimization. *J. Comput. Appl. Math.*, **129**, 15–35.

Morales, J. L. (2002) A numerical study of limited memory BFGS methods. *Appl. Math. Lett.*, **15**, 481–487.
Nocedal, J. (1980) Updating quasi-Newton matrices with limited storage. *Math. Comput.*, **35**, 773–782.
Nocedal, J. & Wright, S. J. (2006) *Numerical Optimization*, 2nd edn. New York: Springer.
Paige, C. C., Parlett, B. N. & van der Vorst, H. (1995) Approximate solutions and eigenvalue bounds from Krylov subspaces. *Numer. Linear Algebra Appl.*, **2**, 115–133.
Raydan, M. (1993) On the Barzilai and Borwein choice of steplength for the gradient method. *IMA J. Numer. Anal.*, **13**, 321–326.
Raydan, M. (1997) The Barzilai and Borwein gradient method for the large scale unconstrained minimization problem. *SIAM J. Optim.*, **7**, 26–33.
Shanno, D. F. (1970) Conditioning of quasi-Newton methods for function minimization. *Math. Comput.*, **24**, 647–656.
Xiao, Y., Wang, Q. & Wang, D. (2010) Notes on the Dai-Yuan-Yuan modified spectral gradient method. *J. Comput. Appl. Math.*, **234**, 2986–2992.
Yuan, Y.-X. (2006) A new step size for the steepest descent method. *J. Comput. Math.*, **24**, 149–156.
Zhang, H. & Hager, W. W. (2004) A nonmonotone line search technique and its application to unconstrained optimization. *SIAM J. Optim.*, **14**, 1043–1056.

Appendix A. Proofs of theorems

In this appendix, we provide proofs for Theorems 3.1–3.3 as stated in Section 3.2.

Proof of Theorem 3.1. The matrix R_k in the (thin) QR-factorization of G_k is invertible if and only if the columns of G_k are linearly independent (see Golub and Van Loan, 1996), which is the first part of the theorem. Now, for the remainder of the proof, we may proceed under the assumption that the columns of G_k are linearly independent (which implies that R_k is invertible). Since R_k is invertible, we have

$$\begin{aligned} T_k &= [R_k \quad r_k]J_k R_k^{-1} \\ &= [Q_k^{\mathrm{T}} Q_k R_k \quad Q_k^{\mathrm{T}} Q_k r_k]J_k R_k^{-1} \\ &= Q_k^{\mathrm{T}}[G_k \quad g_k]J_k R_k^{-1} \\ &= R_k^{-T} G_k^{\mathrm{T}}[G_k \quad g_k]J_k R_k^{-1}, \end{aligned}$$

which implies that T_k is invertible if and only if $G_k^{\mathrm{T}}[G_k \; g_k]J_k$ is invertible. We find that

$$G_k^{\mathrm{T}}[G_k \quad g_k]J_k = G_k^{\mathrm{T}}[\alpha_{k-m}^{-1}(g_{k-m+1} - g_{k-m}) \quad \cdots \quad \alpha_{k-1}^{-1}(g_k - g_{k-1})],$$

which, along with the fact that $\alpha_k > 0$ for all $k \in \mathbb{N}$, implies that $G_k^{\mathrm{T}}[G_k \; g_k]J_k$ is invertible if and only if

$$G_k^{\mathrm{T}}[g_k - g_{k-m} \quad \cdots \quad g_k - g_{k-1}] = G_k^{\mathrm{T}}([g_k \quad \cdots \quad g_k] - G_k) \tag{A.1}$$

is invertible. Since G_k has linearly independent columns, we have by properties of determinants that

$$\begin{aligned} \det(G_k^{\mathrm{T}}([g_k \quad \cdots \quad g_k] - G_k)) &= \det(G_k^{\mathrm{T}}[g_k \quad \cdots \quad g_k] - G_k^{\mathrm{T}} G_k) \\ &= \det(G_k^{\mathrm{T}} G_k)\det((G_k^{\mathrm{T}} G_k)^{-1} G_k^{\mathrm{T}}[g_k \quad \cdots \quad g_k] - I), \end{aligned}$$

from which it follows that (A.1) is invertible if and only if

$$1 \neq \mathrm{trace}((G_k^{\mathrm{T}} G_k)^{-1} G_k^{\mathrm{T}} [g_k \quad \cdots \quad g_k]) = \mathrm{trace}(R_k^{-1} Q_k^{\mathrm{T}} [g_k \quad \cdots \quad g_k]),$$

which is true if and only if the elements of $R_k^{-1} Q_k^{\mathrm{T}} g_k$ do not sum to 1. □

Proof of Theorem 3.2. Letting $R_k^{(j)}$ denote the jth column of R_k for all $j \in \{1, \ldots, m\}$, we have

$$[R_k \quad r_k] J_k = [\alpha_{k-m}^{-1}(R_k^{(1)} - R_k^{(2)}) \quad \cdots \quad \alpha_{k-2}^{-1}(R_k^{(m-1)} - R_k^{(m)}) \quad \alpha_{k-1}^{-1}(R_k^{(m)} - r_k)]. \tag{A.2}$$

Since R_k is upper triangular, it follows that (A.2) is upper Hessenberg and that R_k^{-1} is upper triangular. Then, since the product of an upper Hessenberg and an upper triangular matrix is upper Hessenberg, it follows that T_k is upper Hessenberg. Thus, our construction of $\tilde{T}_k$ ensures that it is symmetric tridiagonal.

Now, consider $\tilde{P}_k = \tilde{T}_k^{\mathrm{T}} \tilde{T}_k + \zeta_k \zeta_k^{\mathrm{T}}$ where $\zeta_k^{\mathrm{T}} := [0 \ \rho_k] J_k R_k^{-1}$. For any pair of indices $\{i, j\} \subseteq \{1, \ldots, m\}$ such that $|i - j| > 2$, we have $[\tilde{T}_k^{\mathrm{T}} \tilde{T}_k]_{i,j} = 0$, from which it follows that $\tilde{T}_k^{\mathrm{T}} \tilde{T}_k$ is symmetric pentadiagonal. Moreover, by the structure of J_k and the fact that R_k^{-1} is upper triangular, it follows that $\zeta_k \zeta_k^{\mathrm{T}}$ is zero except for its (m, m) entry, which overall implies that $\tilde{P}_k$ is symmetric pentadiagonal. □

Proof of Theorem 3.3. For ease of exposition in this proof, we drop the tilde and iteration subscript from all quantities of interest; in particular, we let $T = \tilde{T}_k$, $P = \tilde{P}_k$ and $\zeta = \zeta_k$. Since T is invertible, there exists an orthogonal matrix V such that $T = V \Lambda V^{\mathrm{T}}$ where Λ is a diagonal matrix whose diagonal elements are the (nonzero) eigenvalues of T; in particular, for some nonnegative integers p and q with $p + q = m$, we have $\Lambda = \mathrm{diag}(a, -b)$ for some positive vectors $a \in \mathbb{R}^p$ and $b \in \mathbb{R}^q$. Without loss of generality, we assume that $a_1 \geqslant \cdots \geqslant a_p > 0$ and $b_1 \geqslant \cdots \geqslant b_q > 0$. Letting $w = V^{\mathrm{T}} \zeta$, we have

$$\begin{aligned} V^{\mathrm{T}}(P^{-1}T)^{-1} V &= V^{\mathrm{T}}(T^{-1}(T^{\mathrm{T}}T + \zeta\zeta^{\mathrm{T}}))V \\ &= \Lambda + V^{\mathrm{T}} T^{-1} \zeta \zeta^{\mathrm{T}} V \\ &= \Lambda + \Lambda^{-1} V^{\mathrm{T}} \zeta \zeta^{\mathrm{T}} V \\ &= \Lambda + \Lambda^{-1} w w^{\mathrm{T}}. \end{aligned}$$

Thus, denoting by $|\Lambda|$ the diagonal matrix whose diagonal elements are the absolute values of the elements of Λ, we have for some vectors $c \in \mathbb{R}^p$ and $d \in \mathbb{R}^m$ that

$$\begin{aligned} V^{\mathrm{T}}(P^{-1}T)^{-1} V &= |\Lambda|^{-1/2} \Lambda |\Lambda|^{1/2} + |\Lambda|^{-1/2} (|\Lambda|^{1/2} \Lambda^{-1} w w^{\mathrm{T}} |\Lambda|^{-1/2}) |\Lambda|^{1/2} \\ &= |\Lambda|^{-1/2} \left(\begin{bmatrix} \mathrm{diag}(a) & 0 \\ 0 & \mathrm{diag}(-b) \end{bmatrix} + \begin{bmatrix} c \\ -d \end{bmatrix} \begin{bmatrix} c \\ d \end{bmatrix}^{\mathrm{T}} \right) |\Lambda|^{1/2} \\ &= |\Lambda|^{-1/2} \left(\begin{bmatrix} \mathrm{diag}(a) + cc^{\mathrm{T}} & cd^{\mathrm{T}} \\ -dc^{\mathrm{T}} & \mathrm{diag}(-b) - dd^{\mathrm{T}} \end{bmatrix} \right) |\Lambda|^{1/2}. \end{aligned}$$

It follows that the eigenvalues of $(P^{-1}T)^{-1}$ are the same as those for

$$\Omega := \begin{bmatrix} \mathrm{diag}(a) + cc^{\mathrm{T}} & cd^{\mathrm{T}} \\ -dc^{\mathrm{T}} & \mathrm{diag}(-b) - dd^{\mathrm{T}} \end{bmatrix}.$$

Now, for $i \in \{1, \dots, p\}$, let e_j denote the jth unit vector in $\mathbb{R}^p$ and note that, for all $j \in \{1, \dots, i\}$,

$$\begin{bmatrix} e_j \\ 0 \end{bmatrix}^{\mathrm{T}} (\Omega - a_i I) \begin{bmatrix} e_j \\ 0 \end{bmatrix} = (a_j + c_j^2) - a_i \geqslant 0.$$

We may conclude from this fact that, for all $i \in \{1, \dots, p\}$, the matrix $\Omega - a_i I$ has at least i positive eigenvalue(s). Similarly, for a given $i \in \{1, \dots, q\}$ and with $\bar{e}_j$ denoting the jth unit vector in $\mathbb{R}^q$, we have for all $j \in \{1, \dots, i\}$ that

$$\begin{bmatrix} 0 \\ \bar{e}_j \end{bmatrix}^{\mathrm{T}} (\Omega + b_i I) \begin{bmatrix} 0 \\ \bar{e}_j \end{bmatrix} = (-b_j - d_j^2) + b_i \leqslant 0,$$

from which it follows that $\Omega + b_i I$ has at least i negative eigenvalue(s). Recalling that $(a, -b)$ are the eigenvalues of T, we conclude that the eigenvalues of $(P^{-1}T)^{-1}$, call them $(\bar{a}, -\bar{b})$, satisfy

$$-\bar{b}_1 \leqslant -b_1 \leqslant \cdots \leqslant -\bar{b}_q \leqslant -b_q < 0 < a_p \leqslant \bar{a}_p \leqslant \cdots \leqslant a_1 \leqslant \bar{a}_1,$$

as desired. □

Appendix B. Tables of numerical results

In this appendix, we provide details of the results of our numerical experiments that were summarized in Section 5. The following tables provide function and gradient evaluation counts for the implemented algorithms QUAD-RITZ, QUAD-HRITZ and CUBIC for the history length parameter values $m \in \{1, 3, 5\}$.

TABLE B1 *Results for $m = 1$*

Name	n	QUAD-RITZ		QUAD-HRITZ		CUBIC	
		#f	#g	#f	#g	#f	#g
CHAINWOO	10,000	809	412	507	454	676	644
CHNROSNB	50	2977	1486	1927	1371	1459	1433
DECONVU	61	276,033	126,801	23,625	23,553	23,625	23,553
DIXMAANE	9000	3597	1705	1205	1151	1205	1151
DIXMAANF	9000	3788	1789	1284	1224	1284	1224
DIXMAANG	9000	3755	1763	791	761	791	761
DIXMAANH	9000	2610	1265	939	918	939	918
DIXMAANI	9000	158,013	70,930	16,912	16,764	16,912	16,764
DIXMAANJ	9000	2402	1156	1033	999	1033	999
DIXMAANK	3000	10,612	4851	1275	1238	1275	1238
DIXON3DQ	10,000	1,907,708	865,306	222,481	222,086	222,481	222,086
EIGENALS	110	20,405	9231	3049	2947	3049	2947
EIGENBLS	110	24,276	11,095	1232	894	10,075	9351
EIGENCLS	462	49,256	22,791	9640	8186	11,355	11,086
ERRINROS	50	714,084	331,317	20,352	20,105	20,352	20,105
EXTROSNB	1000	1,031,208	474,887	126,174	126,096	126,174	126,096
FMINSRF2	15,625	6454	3104	1709	1681	1709	1681

(*continued*)

TABLE B1 *(Continued)*

Name	n	QUAD-RITZ		QUAD-HRITZ		CUBIC	
		#f	#g	#f	#g	#f	#g
FMINSURF	1024	2541	1298	817	778	817	778
GENHUMPS	5000	29,372	14,564	2135	1109	19,378	19,305
GENROSE	500	9483	4245	9517	3140	5353	4899
HYDC20LS	99	4,614,763	2,095,069	1,071,208	1,070,629	1,071,208	1,070,629
MODBEALE	2000	1273	527	204	164	204	164
MSQRTALS	529	248,340	113,968	28,812	28,688	28,812	28,688
MSQRTBLS	529	218,650	100,286	24,993	24,882	24,993	24,882
NONCVXU2	10,000	39,042	18,952	13,287	9734	11,375	11,151
NONCVXUN	10,000	75,236	36,361	15,263	10,987	16,727	16,418
NONDQUAR	10,000	612	368	266	257	266	257
SPMSRTLS	10,000	1451	753	632	597	632	597
TQUARTIC	10,000	11,279	2026	1282	1016	1282	1016
WOODS	10,000	2760	960	887	479	842	621

TABLE B2 *Results for $m = 3$*

Name	n	QUAD-RITZ		QUAD-HRITZ		CUBIC	
		#f	#g	#f	#g	#f	#g
CHAINWOO	10,000	139,022	72,636	440,326	77,900	116,431	97,997
CHNROSNB	50	1678	953	2792	908	1078	995
DECONVU	61	101,433	57,765	18,820	18,680	19,120	19,019
DIXMAANE	9000	2782	1822	875	863	875	863
DIXMAANF	9000	1499	932	1069	952	1100	1044
DIXMAANG	9000	1731	1070	1009	905	755	729
DIXMAANH	9000	1544	958	922	829	920	886
DIXMAANI	9000	98,504	59,120	33,824	33,723	32,322	32,270
DIXMAANJ	9000	1325	850	875	721	714	662
DIXMAANK	3000	1053	668	2530	2479	478	467
DIXON3DQ	10,000	1,779,714	1,079,416	1,421,808	1,421,753	1,387,582	1,387,527
EIGENALS	110	6016	3150	3348	2522	2320	2144
EIGENBLS	110	12,932	7548	11,406	6405	10,798	10,116
EIGENCLS	462	25,440	13,729	15,027	10,957	10,025	9556
ERRINROS	50	25,428	9363	10,147	6269	25,089	24,050
EXTROSNB	1000	265,288	147,208	85,540	82,318	86,104	85,247
FMINSRF2	15,625	17,509	2928	15,208	2211	13,905	2605
FMINSURF	1024	8793	1388	4919	948	6470	1178
GENHUMPS	5000	7414	2054	89,478	16,611	22,323	17,068
GENROSE	500	7754	3820	19,232	3264	5067	4724
HYDC20LS	99	3,989,081	2,379,958	1,527,159	1,527,100	1,527,159	1,527,100
MODBEALE	2000	901	221	534	226	269	220
MSQRTALS	529	134,835	83,020	54,501	54,410	53,480	53,414

(continued)

TABLE B2 *(Continued)*

Name	n	QUAD-RITZ		QUAD-HRITZ		CUBIC	
		#f	#g	#f	#g	#f	#g
MSQRTBLS	529	123,556	76,020	47,057	46,951	43,885	43,854
NONCVXU2	10,000	29,644	13,907	28,592	8941	16,091	10,200
NONCVXUN	10,000	54,336	24,814	55,618	15,836	27,385	16,938
NONDQUAR	10,000	4286	536	6664	769	1319	384
SPMSRTLS	10,000	1,307,875	729,339	607	554	8370	7667
TQUARTIC	10,000	6068	491	20,204	1593	4746	826
WOODS	10,000	2461	458	2023	347	908	415

TABLE B3 *Results for $m = 5$*

Name	n	QUAD-RITZ		QUAD-HRITZ		CUBIC	
		#f	#g	#f	#g	#f	#g
CHAINWOO	10,000	156,942	56,418	4233	689	1122	563
CHNROSNB	50	3314	839	5818	1062	1769	944
DECONVU	61	12,229	7398	4551	3381	4397	3928
DIXMAANE	9000	2089	1498	1094	971	843	784
DIXMAANF	9000	1223	867	996	809	1099	956
DIXMAANG	9000	1575	1006	1062	856	876	776
DIXMAANH	9000	1583	1005	1097	738	876	797
DIXMAANI	9000	53,132	35,845	21,932	21,797	20,201	20,120
DIXMAANJ	9000	1202	776	849	667	583	537
DIXMAANK	3000	854	442	557	386	493	415
DIXON3DQ	10,000	1,777,107	1,213,356	850,208	849,833	909,144	908,761
EIGENALS	110	2392	1076	2687	880	1510	982
EIGENBLS	110	8701	5418	14,548	4788	1764	903
EIGENCLS	462	22,116	7112	17,196	7464	13,688	8888
ERRINROS	50	9760	1427	10,927	1944	6244	1517
EXTROSNB	1000	22,726	10,282	18,073	7767	12,199	8480
FMINSRF2	15,625	9062	1506	10,264	1806	7293	1752
FMINSURF	1024	5143	908	3931	828	3325	836
GENHUMPS	5000	57,998	18,114	143,813	18,945	6110	1393
GENROSE	500	6639	3152	30,851	4620	10,119	5182
HYDC20LS	99	1,241,851	780,101	493,466	492,033	885,588	885,337
MODBEALE	2000	1106	254	2611	464	739	292
MSQRTALS	529	82,328	56,605	39,500	39,273	39,734	39,589
MSQRTBLS	529	72,615	50,086	30,137	29,798	36,146	35,852
NONCVXU2	10,000	24,633	9768	32,486	8602	27,062	10,662
NONCVXUN	10,000	50,520	19,570	59,028	14,601	42,572	14,875
NONDQUAR	10,000	2662	461	3564	589	2149	472
SPMSRTLS	10,000	3857	947	3462	907	1962	844
TQUARTIC	10,000	6336	487	5953	534	2906	497
WOODS	10,000	1804	255	377	94	213	69