Rescaling nonsmooth optimization using BFGS and Shor updates

Jiayi Guo [*] A.S. Lewis [†]

February 23, 2018

Abstract

The BFGS quasi-Newton methodology, popular for smooth minimization, has also proved surprisingly effective in nonsmooth optimization. Through a variety of simple examples and computational experiments, we explore how the BFGS matrix update improves the local metric associated with a convex function even in the absence of smoothness and without using a line search. We compare the behavior of the BFGS and Shor r-algorithm updates.

Key words: convex; BFGS; quasi-Newton; nonsmooth; Shor r-algorithm.

AMS 2000 Subject Classification: 90C30; 65K05.

1 Introduction

We consider unconstrained minimization methods for a function $f\colon \mathbf{R}^n \to \mathbf{R}$. Our aim is to explore basic theory, so for simplicity we assume throughout that f is convex and everywhere finite, even though many of the algorithms we consider are also interesting for functions that may be nonconvex or extended-valued.

Since the 1970's, an extensive literature has documented the powerful properties of the BFGS (Broyden-Fletcher-Goldfarb-Shanno) update in quasi-Newton minimization algorithms. The update accumulates information about the curvature of the objective, allowing, like Newton's method, a beneficial transformation of the space. Remarkably, this process seems to help reliably even when the objective is nonsmooth [13]. In this work we try to illuminate this phenomenon.

[*] ORIE, Cornell University, Ithaca, NY 14853, U.S.A. jg826@cornell.edu.

[†] ORIE, Cornell University, Ithaca, NY 14853, U.S.A. people.orie.cornell.edu/aslewis.
Research supported in part by National Science Foundation Grant DMS-1613996.

When studying the BFGS algorithm in the context of nonsmooth optimization, an interesting point of comparison is the Shor r-algorithm [18]. Shor's method also uses a quasi-Newton-like transformation, simpler than BFGS, but without satisfying the secant condition standard in smooth optimization [15]. The algorithm is challenging to analyze [5], and hard to implement systematically in practice, although there have been promising attempts [12]. One fundamental difficulty is how to incorporate into the method a systematic line search. By contrast, a simple line search (satisfying the standard weak Wolfe conditions) is easy to incorporate into a nonsmooth BFGS algorithm, and seems very successful in practice [13].

Unfortunately, we lack almost any theoretical insight into the benefits of the BFGS (or Shor) update in nonsmooth optimization. The interplay of a line search with quasi-Newton updates complicates the question still further. In this work, we therefore try to isolate the behavior of the BFGS update, in particular, and try to understand its beneficial effects *with no line search*.

On simple random nonsmooth convex optimization problems, a BFGS method can dispense almost entirely with the usual line search and seemingly still reliably succeed. More precisely, if the objective f is smooth at the current iterate $x \in \mathbf{R}^n$ (as holds generically) and the current BFGS matrix is H (n-by-n and positive-definite), then traditionally we calculate $x_+ = x - tH\nabla f(x)$, where the stepsize $t > 0$ is chosen through the line search, apply a standard BFGS update formula to H, and update $x = x_+$. However, a more rudimentary idea is simply to choose $t = 1$, and update H, but only update x if the step generates descent: $f(x_+) < f(x)$. We refer to this stripped-down method as *linesearch-free BFGS*.

In Figure 1, we illustrate the typical performance of the linesearch-free BFGS method on a simple example, where the objective function $f\colon \mathbf{R}^5 \to \mathbf{R}$ is the maximum of four random strictly convex quadratics. Somewhat surprisingly, the method minimizes f reliably, generating a sequence of iterates that appear to converge linearly to the minimizer. Experiments such as these prompt our quest for insight into the BFGS update for nonsmooth functions.

2 Space dilation via Shor and BFGS

We begin our exploration with a simpler method: the Shor r-algorithm for minimizing the function f. At each iteration we consider the current iterate $x \in \mathbf{R}^n$, and a current subgradient $g \in \partial f(x)$. The classical subgradient method takes a step from x in the direction $-g$. Shor [18, Section 3.6] proposed accelerating this idea by successively rescaling the space using a current n-by-n matrix V, initially equal to the identity matrix I. We update the iterate via

$$s = -V^T V g; \quad x_+ = x + ts; \tag{2.1}$$

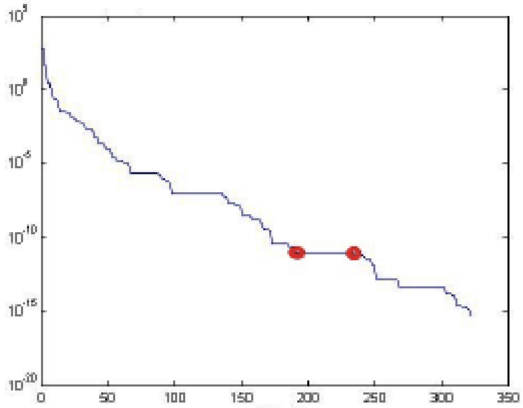

Figure 1: A typical run of the linesearch-free BFGS method for a nonsmooth f on $\mathbf{R}^5$, plotting the value $f(x_k) - \min f$ against the iteration count k. Flat segments (such as between the red dots) indicate BFGS updating without iterate updating.

the stepsize $t > 0$ being chosen through some kind of line search. At the new iterate x_+ we then find a new subgradient $g_+ \in \partial f(x_+)$, define a unit vector $e \in \mathbf{R}^n$ by

$$e = V(g - g_+); \quad e = \frac{e}{\|e\|};$$

update the matrix via

$$W = I - \frac{ee^T}{2\|e\|^2}; \quad V_+ = WV;$$

update $x = x_+$; $g = g_+$; $V = V_+$; and repeat. (The factor "2" that appears in the denominator in the definition of W has no special significance and could be replaced by any constant greater than 1.) Shor described his method as one of "space dilation": after making a current change of variables $x = V^T y$, the unit vector e lies in the direction of the difference of two successive subgradients of the objective function $y \mapsto f(V^T y)$, and the transformation W dilates the space in this direction.

Consider the canonical special case of minimizing a sublinear function:

$$f(x) = \max_{h \in Q} h^T x,$$

for a nonempty compact set of nonzero vectors $Q \subset \mathbf{R}^n$. Then $x = 0$ is nonoptimal if and only if there exists a descent direction: a vector $d \in \mathbf{R}^n$ and a scalar $\alpha < 0$ such that $h^T d \leq \alpha$ for all h in Q. This condition states that a hyperplane normal to d separates zero from Q (or equivalently its convex hull $\operatorname{conv} Q$).

We could apply Shor's method, seeking to minimize the function f starting (and remaining) at the point zero, and terminating once we find a descent direction. More precisely, at each iteration the current iterate is $x = 0$ and the current subgradient g lies in the set Q. We choose the stepsize $t = 1$, terminate if $f(s) < 0$, and otherwise choose a new subgradient $g_+ \in Q$ to maximize the inner product $s^T g_+$. (As we discuss and motivate in Section 5, this choice of g_+ correctly models the function in the search direction: $f(s) = s^T g_+$.) We then update the matrix V and the subgradient g, maintain $x = 0$, and repeat.

Following the change of variables we introduced above, if we define $h = Vg$ and $p = Vg_+$, we arrive at the following simple procedure for separating a set Q from zero, relying only on a linear optimization oracle over Q.

Algorithm 2.2 (Shor update method for $0 \in \mathbf{conv}\, Q$)

```
Choose $h \in Q$; $V = I$;
while not done do
  find a minimizer $p$ of $\langle \cdot, h \rangle$ over $Q$;
  if $p^T h > 0$ then
    terminate with $V^T h$ "normal to separating hyperplane";
  end if
  $e = h - p$; $W = I - \frac{ee^T}{2\|e\|^2}$; $Q = WQ$; $V = WV$; $h = Wp$;
end while
```

Geometrically, the procedure tests whether the current vector $h \in Q$ is normal to a hyperplane separating Q from zero, and if not applies to the set Q a simple linear transformation W, a symmetric rank-one perturbation of the identity.

In its intuitive simplicity and apparent versatility, the procedure has a certain appeal. Furthermore, experiments on small examples suggest it typically works: when zero lies outside the convex hull of Q, the procedure terminates, and otherwise the vector h converges to zero. Especially simple is the case when Q is a finite set of nonzero vectors, in which case we seek to separate the polytope $\mathrm{conv}\, Q$ from zero, or equivalently, find a solution d for the homogeneous system of inequalities $h^T d < 0$ for all $h \in Q$ (a core problem of linear programming). The oracle — linear optimization over Q — is then particularly easy.

Each iteration is computationally simple, involving just elementary operations on all the vectors in Q. The procedure is not as "elementary" as methods like the Perceptron Algorithm, that in particular preserve sparsity, being closer in spirit to rescaled perceptron methods [1]. As we discuss later, it also has some formal similarities with versions of the Ellipsoid Algorithm.

To illustrate, consider the following example in dimension $n = 5$. For each index j, denote the corresponding unit vector in $\mathbf{R}^5$ by e_j. Define vectors $a_j = 4^j e_j$, along with a convex combination $p = (\sum 4^{-j})^{-1} \sum e_j$. Fix a parameter $\epsilon > 0$, and let the set Q consist of the points $a_j - (1 + \epsilon)p$ (for each j) along with $-p$.

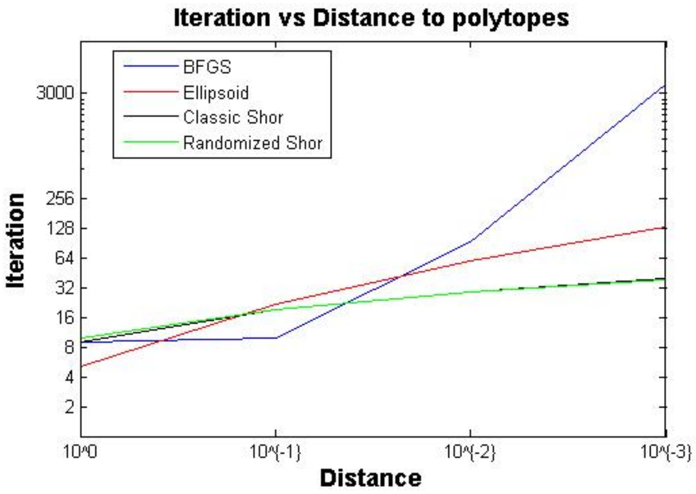

Figure 2: Separating a point from a polytope: mean of number of required iterations to terminate from a random start.

Geometrically, Q consists of the vertices of an irregular simplex. When ϵ is small the point zero is outside the simplex but close to one of the facets, making the problem ill-posed. Figure 2 plots the number of iterations needed by several algorithms to find a separating hyperplane, averaged over the starting point in Q, as a function of the parameter ϵ. The figure compares this Shor update method (labeled "Classic Shor") with several other algorithms discussed below: a randomized Shor method, a BFGS method, and a version of the ellipsoid algorithm. As we see from the plot, on this small and simple example, Shor updating is reliable, terminating after a couple of dozen iterations even with $\epsilon = 10^{-3}$. Figure 3 shows some typical trajectories.

A simple nonpolyhedral problem seeks to separate a point c from an ellipsoid in $\mathbf{R}^n$. If we describe the ellipsoid as AB, where $B \subset \mathbf{R}^n$ is the closed unit ball and A is an invertible n-by-n matrix, then we seek a normal vector $z \in \mathbf{R}^n$ to a separating hyperplane, or in other words a solution of the inequality $\|A^T z\| < c^T z$. If Q is the boundary of the ellipsoid $AB - c$, we arrive at following simple procedure (involving no matrix inversion): if it terminates, the output vector z solves our problem.

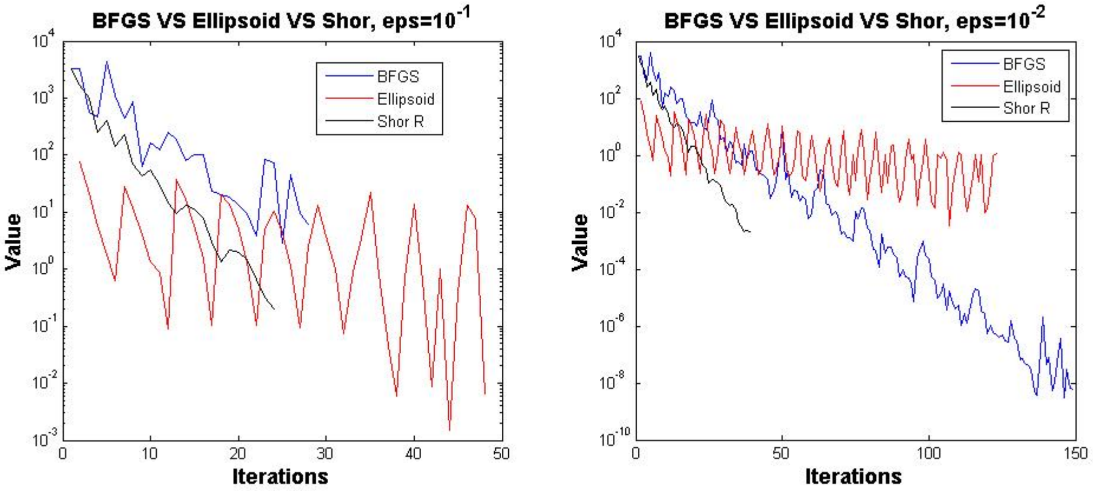

Figure 3: Separating a point from a polytope: typical trajectories.

```
**Algorithm 2.3 (Shor updating to separate point $c$ from ellipsoid $AB$)**
  Choose unit $x \in \mathbf{R}^n$; $V = I$;
  **while** not done **do**
    $h = Ax - c$; $y = -A^T h$; $y = \frac{y}{\|y\|}$; $p = Ay - c$;
    **if** $p^T h > 0$ **then**
      terminate with $z = V^T h$ "normal to separating hyperplane";
    **end if**
    $e = h - p$; $W = I - \frac{ee^T}{2\|e\|^2}$; $A = WA$; $V = WV$; $c = Wc$; $x = y$;
  **end while**
```

(In the notation of Algorithm 2.2, the current iterate h is $Ax - c$ for some unit vector x, and we compute p by minimizing $\langle p, h\rangle$ with $p = Ay - c$, over unit vectors y.)

We illustrate this idea in dimension $n = 5$, for a diagonal matrix A with diagonal $[1\ 10\ 10^2\ 10^3\ 10^4]$. We generate a hundred instances by choosing the vector $c = (1 + d)Au$, where $u \in \mathbf{R}^5$ is a random unit vector, and we set the scalar d (which controls the ill-posed of the instance) to be both 1 and 10^{-1} to illustrate the effect of ill-posedness. The figures below (Figure 4) plot histograms of the number of instances requiring certain numbers of iterations to terminate.

As we see from the plots, on these random examples on a low-dimensional ellipsoid, moderately ill-conditioned and well separated from zero, this Shor updating method works reliably. It typically finds a separating hyperplane after a couple of dozen iterations. Not surprisingly, the required number of iterations grows as the parameter d (and hence distance to ill-posedness) shrinks: examples with $d = 10^{-2}$ may need more than 100 iterations, and $d = 10^{-3}$ may need more than 1000. The

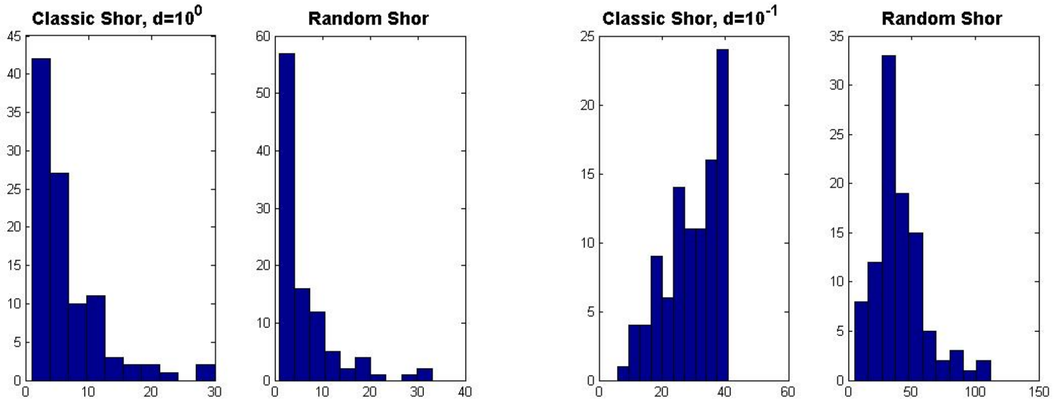

Figure 4: Shor updating to separate a point from an ellipsoid in $\mathbf{R}^5$. Histograms of number of required iterations to terminate, for 100 random examples.

method remains viable as the dimension grows. When the matrix A has diagonal entries $[1\ 10\ 10^2\ \cdots\ 10^9]$, with $d = 10^{-1}$, a hundred random instances all terminated in less than 200 iterations.

Bolstered by such random experiments, where Shor updating systematically succeeds, we might hope for a simple proof validating Algorithm 2.2, and thereby some insight into the Shor r-algorithm. Sadly, while the procedure is simple, its behavior is not: sporadically, *it can fail.* For example, numerical experiments with the ellipsoid separation procedure (Algorithm 2.3) revealed that on the small example in $\mathbf{R}^2$ defined by

$$A = \begin{bmatrix} 1 & 0 \\ 0 & 10 \end{bmatrix}, \quad v = -\begin{bmatrix} 10 \\ 39 \end{bmatrix}, \quad c = (1 + 10^{-2})A\frac{v}{\|v\|},$$

a thousand iterations do not suffice for termination. Furthermore, the failure seems unambiguous: after a few steps, iterations seem to behave cyclically, with a period of five iterations. In particular, the cosine of the angle between the vectors p and h is bounded above by $-1/100$, so the termination criterion always fails. Figure 5 plots this cosine for the first hundred iterations.

Sporadic failures notwithstanding, the numerical evidence suggests that the Shor update in Algorithm 2.2 improves the geometry of the problem in some average sense, perhaps analogous to randomized algorithms for convex programming like [1, 8]. To try to isolate this effect, we consider a simple modification of Algorithm 2.2. The original method remembers the new element $p \in Q$ at the end of an iteration, and uses it, transformed as $h = Wp$, to start the next iteration. The modified method below forgets p after the iteration, starting each new iteration afresh by simply choosing h to optimize over Q in a random direction.

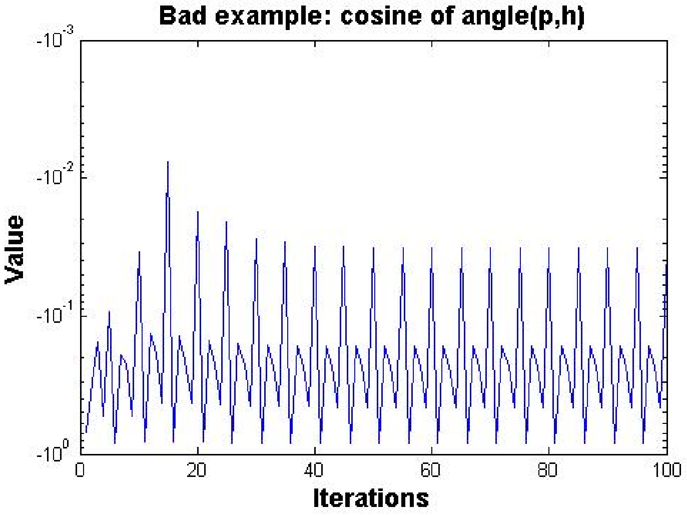

Figure 5: Cyclic behavior of the angle between p and h during Shor updating.

Algorithm 2.4 (Randomized Shor for $0 \in$ conv Q)

```
$V = I$;
while not done do
    choose random $u \in \mathbf{R}^n$;
    find a minimizer $h$ of $\langle \cdot, u \rangle$ over $Q$;
    find a minimizer $p$ of $\langle \cdot, h \rangle$ over $Q$;
    if $p^T h > 0$ then
        terminate with $V^T p$ “normal to separating hyperplane”;
    end if
    $e = h - p$; $W = I - \frac{ee^T}{2\|e\|^2}$; $Q = WQ$; $V = WV$;
end while
```

We could, for example, distribute the random vector u normally. In the special case of the ellipsoid separation procedure, we arrive at a Randomized Algorithm 2.3, where x, rather than equalling y, is just the normalized vector $A^T u$ for a random vector u. In the figure, we compare the results for the randomized procedure with those for the original “classic” procedure, and like that procedure, it seems reliable on small random examples. On the one hand, we observe no failures. On the other hand, shrinking the ill-posedness parameter d seems to slow the randomized procedure more than the original version. With $d = 10^{-2}$ (not shown in the figure), the original version terminates in every instance within around a hundred iterations, whereas the randomized version often takes thousands. In summary, reusing the previous element of the set Q at each iteration seems to accelerate the procedure.

We noted in the introduction that the BFGS method, as a general-purpose nonsmooth optimization tool, shows more promise than the Shor r-algorithm, and is

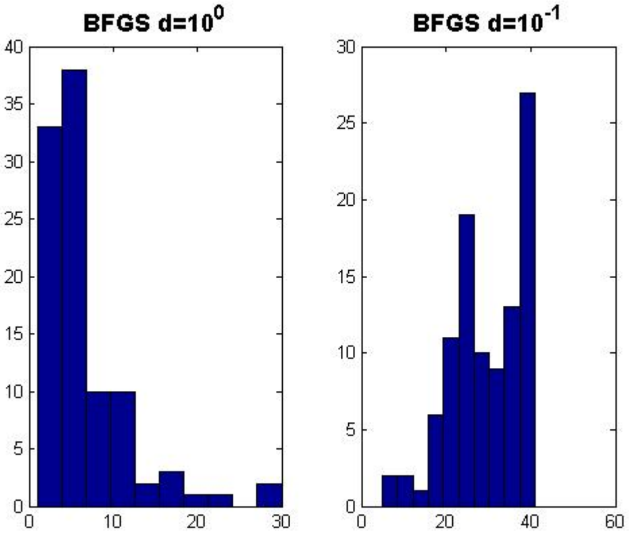

Figure 6: BFGS updating to separate a point from an ellipsoid in $\mathbf{R}^5$. Histograms of number of required iterations to terminate, for 100 random examples.

quite successful in practice [13]. A single modification to each of the space-dilation algorithms above transforms the motivation from the Shor update to the BFGS quasi-Newton update. Specifically, as we explain in Section 8, we simply change the updating transformation from

$$W = I - \frac{ee^T}{2\|e\|^2}$$

to

$$W = I - \Big(\frac{e}{h^T e} - \frac{h}{\|h\|\sqrt{h^T e}}\Big)h^T$$

A geometric interpretation of the resulting BFGS-based procedure is almost identical to that for the Shor updating procedure. The only difference is that the matrix W transforming the space, while still a rank-one perturbation of the identity, is now no longer symmetric. As with the Shor update, any convergence theory for this BFGS procedure seems elusive, but its simplicity and apparent effectiveness are intriguing.

On polyhedral separation examples (Figure 2), the BFGS method is successful but seems slower than the Shor method as the ill-posedness parameter ϵ shrinks. The previous ellipsoid separation examples suggest a similar comparison (Figure 6): the Shor method seems faster as the ill-posedness parameter shrinks (although the randomized version seems slower), and sporadic failure is a possibility.

3 The BFGS update

To begin a more careful discussion of the BFGS update, we first recall the classical idea of Newton's method as a steepest descent method in a local metric. Suppose the function f is $\mathcal{C}^2$-smooth, and consider a point $x \in \mathbf{R}^n$ at which the Hessian $\nabla^2 f(x)$ is positive definite. If we denote the gradient $\nabla f(x)$ by g, then the unit steepest descent step, with respect to the Euclidean norm, is the minimizer of the linear approximation $g^T s$ over the unit ball $\{s : \|s\| \leq 1\}$, namely $s = -\frac{1}{\|g\|} g$ (assuming $g \neq 0$). If instead we minimize over the ball $\{s : s^T \nabla^2 f(x) s \leq 1\}$ corresponding to a natural local metric associated with f at x, we instead arrive at the Newton step $s = -\nabla^2 f(x)^{-1} g$. In stark contrast to the steepest descent step, taking the Newton step from x and iterating — Newton's method — rapidly reduces the objective value, at least close to a minimizer $\bar{x}$ with $\nabla^2 f(\bar{x})$ positive definite. If the initial point is far from $\bar{x}$, a simple backtracking line search along the direction of the Newton step can guarantee progress into the neighborhood where the unit Newton step is acceptable.

Turning to quasi-Newton methods, the classical idea is to use an approximation H in the place of the inverse Hessian $\nabla^2 f(x)^{-1}$, updated after each step. In particular, the BFGS (Broyden-Fletcher-Goldfarb-Shanno) update, assuming, like Newton's method, a unit step, uses a matrix H in the set $\mathbf{S}^n_{++}$ of positive-definite n-by-n symmetric matrices. We update H as follows:

$$(3.1) \qquad s = -Hg, \quad x_+ = x + s, \qquad g_+ = \nabla f(x_+), \quad y = g_+ - g$$

$$(3.2) \qquad V = I - \frac{sy^T}{s^T y}, \qquad H_+ = VHV^T + \frac{ss^T}{s^T y}.$$

In this update, we assume that the quantity $s^T y$ is strictly positive: it must be nonnegative, by convexity, but unless f is strictly convex, it may be zero. For future reference, we make the following definition.

Definition 3.3 Given a $\mathcal{C}^1$-smooth convex function $f \colon \mathbf{R}^n \to \mathbf{R}$ and a point x in $\mathbf{R}^n$, denote the gradient $\nabla f(x)$ by g. The *unit-step BFGS update* is the map $\mathrm{BFGS}_{f,x} \colon \mathbf{S}^n_{++} \to \mathbf{S}^n_{++}$ defined by $\mathrm{BFGS}_{f,x}(H) = H_+$ for a matrix $H \in \mathbf{S}^n_{++}$, where the matrix H_+ is given by equations (3.1) and (3.2). If $s^T y = 0$ in equation (3.1), the matrix H_+ is undefined.

Like Newton's method, if the initial point x and approximation H are close to the minimizer $\bar{x}$ and inverse Hessian $\nabla^2 f(\bar{x})^{-1}$ respectively, then updating $x = x_+$, $g = g_+$, $H = H_+$, and iterating, rapidly reduces the objective value [7, Thm 8.6]. Far from $\bar{x}$, a line search again can guarantee progress into the neighborhood where the unit step is acceptable, resulting in an algorithm with good global *and* local convergence properties: see [15] for more details and an extended discussion of the enduringly popular BFGS method. Again like Newton's method, rather than thinking of H as an inverse Hessian approximation, we can instead associate it with a local metric at x, and thereby interpret the BFGS as a variable metric method. This viewpoint better suits our current development, where the objective function f may not be smooth.

For *nonsmooth* optimization, although supported by little theory, extensive computational experiments suggest that the BFGS method can also be surprisingly effective [13]. Under reasonable conditions, with a suitably randomized initial point, the function f is smooth at all points encountered by the method, so the update equations (3.1) and (3.2) make sense. In general, as in the smooth case, it is crucial to incorporate a suitable line search, scaling the step s defined in equation (3.1) at the outset of the update. In particular, the experiments in [13] rely on a weak Wolfe line search, ensuring both a sufficient decrease condition on the new objective value $f(x_+)$ and a curvature condition. Classically, these conditions serve multiple purposes. The sufficient decrease condition is important in convergence proofs (although in practice simply ensuring decrease typically seems to suffice). The curvature condition guarantees in particular the condition $s^T y > 0$, which in turn ensures that the update H_+ is positive definite (although for strictly convex objectives f this property is automatic). Well-known self-correcting properties of the BFGS update, allowing recovery from badly scaled approximations H, are thought to depend heavily on the line search [15].

Important as it is, the line search complicates the already challenging task of understanding how the BFGS update improves the local metric, especially in the nonsmooth case. The line search may sometimes be irrelevant, as occurs asymptotically in the smooth case, for example. We therefore ask:

- What can we learn simply from the unit-step BFGS update (3.1) and (3.2), *with no line search*?

- In particular, when does repeated application of the unit-step BFGS update at a fixed point generate a descent step?

- Can unit-step BFGS updating underly a nonsmooth minimization algorithm?

4 BFGS updating for smooth functions

Although our primary interest is in nonsmooth functions, we begin with the simplest smooth case. Consider a $\mathcal{C}^1$-smooth strictly convex function $f\colon \mathbf{R}^n \to \mathbf{R}$, a point $x \in \mathbf{R}^n$ at which the gradient $g = \nabla f(x)$ is nonzero, and an initial matrix $H \in \mathbf{S}^n_{++}$. Consider the following procedure, which repeatedly applies the unit-step BFGS update (Definition 3.3) at the fixed point x until it generates a descent step.

Algorithm 4.1 (BFGS updating for smooth function f)

```
while $f(x - Hg) \geq f(x)$ do
    $H = \mathrm{BFGS}_{f,x}(H)$;
end while
```

(Strict convexity of f ensures that the update is always well-defined.) When must this iteration terminate? We begin with the one-dimensional case.

Theorem 4.2 *BFGS updating (Algorithm 4.1) terminates for any $\mathcal{C}^1$-smooth strictly convex function $f\colon \mathbf{R} \to \mathbf{R}$ at any noncritical point x.*

Proof We argue by contradiction. Suppose without loss of generality $g = f'(x) = 1$ and the iteration does not terminate. The matrix H is now simply a scalar $h > 0$, and we have $s = -h$, $x_+ = x - h$, $g_+ = f'(x-h)$, $y = f'(x-h) - 1$, and $V = 0$. We then update:

$$h \leftarrow h_+ = \frac{h}{1 - f'(x-h)}.$$

By assumption, $f(x-h) \geq f(x)$, so $f'(x-h) < 0$. Hence $h_+ < h$ at every iteration, so h decreases to some limit $\bar{h} \geq 0$. By continuity we have $f'(x - \bar{h}) \leq 0$. If $f'(x-\bar{h}) = 0$, then we obtain a contradiction, since then the points $x - h$ approach the minimizer $x - \bar{h}$ so eventually $f(x-h) < f(x)$. Hence in fact we have $f'(x-\bar{h}) < 0$. But now we obtain the contradiction

$$h_+ = \frac{h}{1 - f'(x-h)} \to \frac{\bar{h}}{1 - f'(x-\bar{h})} < \bar{h}.$$

This completes the proof. □

Using standard theory from the classical quasi-Newton literature [7,10], we next work towards an analogous result for multivariate quadratic functions. Consider first the special case $f(x) = \frac{1}{2}\|x\|^2$ (for $x \in \mathbf{R}^n$). In that case a quick calculation shows that Algorithm 4.1 becomes the following.

Algorithm 4.3 (BFGS updating for $\frac{1}{2}\|\cdot\|^2$)

```
while $\|x - Hx\| \geq \|x\|$ do
    $s = -Hx$; $z = \frac{s}{\|s\|}$; $H = (I - zz^T)H(I - zz^T) + zz^T$;
end while
```

In fact the case of a general strictly convex quadratic function $f(x) = \frac{1}{2}\|Rx\|^2$ (for $x \in \mathbf{R}^n$), where the n-by-n matrix R is invertible, follows immediately from this special case. The change of variables $\hat{x} = Rx$ and $\hat{H} = RHR^T$ shows, after some algebra, that the BFGS updating algorithm is essentially identical to Algorithm 4.3:

```
**while** $\|\hat{x} - \hat{H}\hat{x}\| \geq \|\hat{x}\|$ **do**
    $s = -\hat{H}\hat{x}$; $z = \frac{s}{\|s\|}$; $\hat{H} = (I - zz^T)\hat{H}(I - zz^T) + zz^T$;
**end while**
```

To proceed, we begin with some geometry in the Euclidean space $\mathbf{S}^n$ of n-by-n symmetric matrices with the inner product defined by $\langle X, Y \rangle = \text{trace}(XY)$, for matrices $X, Y \in \mathbf{S}^n$. We start with a tool whose proof is immediate.

Lemma 4.4 *Consider any matrix $P \in \mathbf{S}^n$ satisfying $P^2 = P$. For any matrix $X \in \mathbf{S}^n$, the matrix $X_+ = PXP$ is orthogonal to the matrix $X_+ - X$.*

For the next step, we denote the smallest eigenvalue of a matrix $H \in \mathbf{S}^n$ by $\lambda_{\min}(H)$.

Lemma 4.5 *For any unit vector $z \in \mathbf{R}^n$, and any matrix $H \in \mathbf{S}^n$, the matrix*

$$H_+ = (I - zz^T)H(I - zz^T) + zz^T$$

satisfies the orthogonality condition $(H_+ - I) \perp (H_+ - H)$, and consequently

$$\|(H - I)z\|^2 \;\leq\; \|H - I\|^2 - \|H_+ - I\|^2,$$

and furthermore $\lambda_{\min}(H_+) \;\geq\; \min\{\lambda_{\min}(H), 1\}$.

Proof In Lemma 4.4, we consider the matrices $P = I - zz^T$ and $X = H - I$. Then we have

$$X_+ \;=\; (I - zz^T)(H - I)(I - zz^T) \;=\; H_+ - I,$$

so the orthogonality condition follows. Using this, and noting $H_+ z = z$, since

$$\|(H - I)z\|^2 \;=\; \|(H - H_+)z\|^2 \;\leq\; \|H - H_+\|^2 \;=\; \|H - I\|^2 - \|H_+ - I\|^2,$$

and the first inequality follows.

Turning to the second inequality, choose a unit vector $u \in \mathbf{R}^n$ satisfying $u^T H_+ u = \lambda_{\min}(H_+)$. Then we deduce

$$\begin{aligned}
\lambda_{\min}(H_+) &= \big(u - (z^T u)z\big)^T H \big(u - (z^T u)z\big) + (z^T u)^2 \\
&\geq \|u - (z^T u)z\|^2 \lambda_{\min}(H) + (z^T u)^2 \\
&= \big(1 - (z^T u)^2\big)\lambda_{\min}(H) + (z^T u)^2 \\
&= \big(1 - \lambda_{\min}(H)\big)(z^T u)^2 + \lambda_{\min}(H).
\end{aligned}$$

The result now follows. □

We can now complete our analysis in the quadratic case.

Theorem 4.6 *BFGS updating (Algorithm 4.1) terminates for any strictly convex quadratic function f at any noncritical point.*

Proof We apply Lemma 4.5. As we have argued, it suffices to consider Algorithm 4.3 at any nonzero point $x \in \mathbf{R}^n$, so suppose by way of contradiction that the procedure does not terminate. As the iterations progress, the nonnegative quantity $\|H - I\|$ is nonincreasing. The first important consequence is the uniform boundedness of the matrix H and hence that of the vector s. Secondly, we also deduce $(H - I)z \to 0$. Denoting the initial matrix H by H_0, we see by induction the inequality $\lambda_{\min}(H) \geq \min\{\lambda_{\min}(H_0), 1\} > 0$. Thus the matrix H^{-1} also stays uniformly bounded, so we deduce $\frac{1}{\|s\|}s + x = (I - H^{-1})z \to 0$. Consequently we see $s \to -x$, contradicting the assumption that the procedure does not terminate. □

The argument above in fact proves more.

Theorem 4.7 *The linesearch-free BFGS method converges to the minimizer of any strictly convex quadratic function.*

Proof As above, after a suitable change of variables it suffices to prove the result for the case $f(x) = \frac{1}{2}\|x\|^2$. Starting from any nonzero initial point $x \in \mathbf{R}^n$ and matrix $H \in \mathbf{S}^n_{++}$, we are therefore repeating the following procedure.

```
while $\|x - Hx\| \geq \|x\|$ do
  $s = -Hx$; $x_+ = x + s$; $z = \frac{s}{\|s\|}$; $H = (I - zz^T)H(I - zz^T) + zz^T$;
end while
$x = x_+$;
```

Exactly as before, we argue that the matrices H and H^{-1} remain uniformly bounded. By the definition of the sequence of iterates x, we know $\|x\|$ is nonincreasing, so the steps s are also uniformly bounded. We deduce $x_+ \to 0$, since as before we know

$$\frac{s + x}{\|s\|} \to 0. \tag{4.8}$$

If the iterates x do not converge to zero, then they are uniformly bounded away from zero, and hence eventually we always accept the step because $\|x_+\| < \|x\|$. But this is a contradiction, since $x_+ \to 0$. □

The argument shows a little more. After taking the inner product with the unit vector $\frac{s}{\|s\|}$, equation (4.8) shows $\frac{1}{\|s\|^2}s^T x \to -1$ and hence $\frac{1}{\|s\|^2}(\|x_+\|^2 - \|x\|^2) \to -1$. Thus eventually we always accept the step because $\|x_+\| < \|x\|$. We have thus shown that the linesearch-free BFGS method applied to a strictly convex quadratic always accepts the step eventually. The method then reduces to the classical method, and hence converges superlinearly to the minimizer [15].

5 BFGS updating for nonsmooth functions

In practice we can apply the classical BFGS method directly to nonsmooth functions, after a randomized initialization, as we noted in the introduction (see [13]). However, our aim here is to illuminate the effect of the BFGS update for nonsmooth functions as simply as possible, so we first consider more formally how we should define it.

To that end, consider a convex function $f\colon \mathbf{R}^n \to \mathbf{R}$, possibly nonsmooth. Given a current point x, subgradient $g \in \partial f(x)$, and matrix $H \in \mathbf{S}^n_{++}$, generalizing the classical BFGS method (with a unit step) leads to the following update:

$$s = -Hg, \qquad x_+ = x + s \tag{5.1}$$

$$g_+ \;\in\; \operatorname{argmax}\big\{ z^T s : z \in \partial f(x_+) \big\} \tag{5.2}$$

$$y = g_+ - g, \quad V = I - \frac{sy^T}{s^T y}, \qquad H_+ = VHV^T + \frac{ss^T}{s^T y}. \tag{5.3}$$

A priori it seems that we might choose the subgradient g_+ arbitrarily from $\partial f(x_+)$. The motivation for the particular choice in equation (5.2) deserves some explanation.

As we have discussed, practical BFGS methods, both in the classical smooth case and in the nonsmooth case, use a line search, suitably scaling the step s before updating $x \leftarrow x_+,\; g \leftarrow g_+,\; H \leftarrow H_+$ and repeating. The curvature condition in the line search depends crucially on the directional derivative $f'(x_+; s)$ of the objective f at the new point x_+ along the search direction s. By standard convex analysis [3], that directional derivative is given by

$$f'(x_+; s) \;=\; \max\big\{ z^T s : z \in \partial f(x_+) \big\} \;=\; g_+^T s.$$

Thus our choice of the new subgradient g_+ corresponds to the correct linear approximation to the objective f at the new point x_+ along the direction of the last step s. Worth noting too is that, for analogous reasons, this choice of subgradient is also reminiscent of the subgradients generated by bundle methods for convex optimization [11].

Repeating this generalized unit-step BFGS update at a fixed point x presents a fresh difficulty: not only must we choose a subgradient $g_+ \in \partial f(x_+)$ but we may also update the original subgradient $g \in \partial f(x)$. An analogous argument to the previous paragraph suggests the choice

$$g_{++} \;\in\; \operatorname{argmax}\big\{ z^T s : z \in \partial f(x) \big\}, \tag{5.4}$$

since this corresponds to the correct linear approximation to the objective f at the fixed point x along the direction of the last trial step s: $f'(x; s) = g_{++}^T s$. We are therefore led to the following generalized definition.

Definition 5.5 Consider a convex function $f\colon \mathbf{R}^n \to \mathbf{R}$ and a point x in $\mathbf{R}^n$. The *(nonsmooth) unit-step BFGS update* is the set-valued mapping

$$\mathrm{BFGS}_{f,x}\colon \partial f(x) \times \mathbf{S}^n_{++} \rightrightarrows \partial f(x) \times \mathbf{S}^n_{++}$$

defined, for any subgradient $g \in \partial f(x)$ and a matrix $H \in \mathbf{S}^n_{++}$, by

$$\mathrm{BFGS}_{f,x}(g, H) \;=\; \big\{(g_{++}, H_+) : (5.1),\ (5.2),\ (5.3),\ (5.4)\ \text{ hold}\big\}.$$

Notice that the set of updates $\mathrm{BFGS}_{f,x}(g, H)$ is empty if $s^T y = 0$ in equation (5.3). When the function f is smooth, the set $\mathrm{BFGS}_{f,x}(\nabla f(x), H)$ consists of just one element, namely the matrix we called $\mathrm{BFGS}_{f,x}(H)$ in our previous notation.

We can now pose our questions at the end of the introduction more precisely. In particular, consider the nonsmooth unit-step BFGS update algorithm for the objective function f at a fixed point x:

$$\textbf{while } f(x - Hg) \geq f(x), \quad (g, H) \leftarrow \mathrm{BFGS}_{f,x}(g, H). \tag{5.6}$$

- If x is not a minimizer, what conditions guarantee termination with descent: $f(x - Hg) < f(x)$?
- If x is a minimizer, what conditions guarantee that the step $-Hg$ converges to zero?

6 BFGS updating for sublinear functions

We now return to the interesting special case we discussed in Section 2, when the point of interest is $x = 0$, and the function f is sublinear. In that case the unit-step BFGS update simplifies. As a consequence of the following result, whose proof is an easy exercise, the distinction between the sets of acceptable subgradients g_+ and g_{++} vanishes in this case. To simplify the update in this case, we can choose $g_{++} = g_+$.

Proposition 6.1 *For any sublinear function $f\colon \mathbf{R}^n \to \mathbf{R}$ and any vector $s \in \mathbf{R}^n$, the maximum value of the linear function $\langle s, \cdot \rangle$ over the subdifferential $\partial f(0)$ is $f(s)$, and the set of maximizers is $\partial f(s)$.*

We can consider any sublinear function $f\colon \mathbf{R}^n \to \mathbf{R}$ as the support function δ^*_C of a nonempty compact set $C \subset \mathbf{R}^n$, namely $C = \partial f(0)$. Hence the nonsmooth unit-step BFGS update algorithm (5.6) for deciding whether or not the point zero minimizes a sublinear function $f = \delta^*_C$ is equivalent to the following algorithm for deciding whether or not zero lies in the compact convex set C.

Algorithm 6.2 (BFGS for $0 \in C$)

```
Choose $g \in C$, and $H \in \mathbf{S}^n_{++}$;
for $k = 0, 1, 2, \ldots$ do
  if $g = 0$ then
    terminate with "$0 \in C$";
  end if
  $s = -Hg$;
  Find a maximizer $g_+$ of $\langle \cdot, s \rangle$ over $C$;
  if $g_+^T s < 0$ then
    terminate with $s$ "normal to hyperplane separating 0 from $C$";
  end if
  $y = g_+ - g$; $V = I - \frac{sy^T}{s^T y}$; $H_+ = VHV^T + \frac{ss^T}{s^T y}$; $H = H_+$; $g = g_+$;
end for
```

Notice that if both the stopping conditions fail, so $g \neq 0$ and $g_+^T s \geq 0$, then

$$y^T s = g_+^T s - g^T s \geq g^T H g > 0,$$

so the BFGS update is well-defined.

We can translate the questions at the end of the previous section for this special case. Consider Algorithm 6.2 applied to a compact convex set C. What conditions ensure the following properties?

- $0 \notin C \;\Rightarrow$ correct termination.
- $0 \in C \;\Rightarrow$ either correct termination or convergence of the step s to zero.

The algorithm depends on being able to maximize linear functionals over the compact convex set $C \subset \mathbf{R}^n$, so is most realistic when C is the convex hull of a possibly simpler (even finite) compact set $D \subset \mathbf{R}^n$. In that case we can choose to restrict our attention to maximizers g_+ in D rather than C. Furthermore, if $0 \notin D$, we can omit the first termination criterion. We then arrive at the following algorithm for deciding whether or not zero lies in the convex hull of a compact set $D \subset \mathbf{R}^n$ not containing zero.

Algorithm 6.3 (BFGS for $0 \in \mathbf{conv}\, D$)

```
  Choose $g \in D$, and $H \in \mathbf{S}^n_{++}$;
  for $k = 0, 1, 2, \ldots$ do
    $s = -Hg$;
    Find a maximizer $g_+$ of $\langle \cdot, s \rangle$ over $D$;
    if $g_+^T s < 0$ then
      terminate with $s$ "normal to hyperplane separating 0 from conv $D$";
    end if
    $y = g_+ - g$; $V = I - \frac{sy^T}{s^T y}$; $H = VHV^T + \frac{ss^T}{s^T y}$; $g = g_+$;
  end for
```

Many authors (such as [2, p. 1051]) have noted the similarities between quasi-Newton algorithms like the BFGS method, and the Ellipsoid Algorithm and related space-dilation techniques (especially the Shor r-algorithm [18, Section 3.6]). The Ellipsoid Algorithm for minimizing, over the unit ball, the support function δ_C^*, when the set C is the convex hull of a compact set $D \subset \mathbf{R}^n$ not containing zero, takes the following form [4, p. 249]. We note the similarities with Algorithm 6.3.

Algorithm 6.4 (Ellipsoid algorithm for $0 \in \mathbf{conv}\, D$)

```
  $x = 0$; $H = I$;
  for $k = 0, 1, 2, \ldots$ do
    if $\|x\| > 1$ then
      $g = x$;
    else
      Find a maximizer $g$ of $\langle \cdot, x \rangle$ over $D$;
      if $g^T x < 0$ then
        terminate with "$x$ separates 0 from conv $D$";
      end if
    end if
    $s = -Hg$; $x = x + \frac{s}{(n+1)\sqrt{-s^T g}}$; $H = \frac{n^2}{n^2-1}\big(H + \frac{2ss^T}{(n+1)s^T g}\big)$;
  end for
```

7 Symmetry and the unit ball

We begin with the second of our two questions: how does Algorithm 6.2 behave when the compact convex set C contains zero? We recall the following measure of the symmetry of the set C:

$$\mathrm{sym}(C) \;=\; \max\{t : g \in C \;\Rightarrow -tg \in C\}.$$

This measure often appears in complexity analysis for convex optimization [9,14,17]. We also use the following standard result, whose proof entails simple linear algebra [15, equation (6.45)].

Lemma 7.1 *The matrices H and H_+ in Algorithm 6.2 satisfy*

$$\frac{\det H_+}{\det H} \;=\; -\frac{s^T g}{s^T y}.$$

In fact this result holds for any matrices $H, H_+ \in \mathbf{S}^n$ related via the BFGS update equations (5.1) and (5.3).

Our next result shows that when the set C contains zero in its interior, the determinant of the matrix H must converge to zero, and at a linear rate controlled by the symmetry measure.

Proposition 7.2 *If the compact convex set C contains zero, then the matrices H and H_+ in Algorithm 6.2 always satisfy*

$$\det H_+ \;\le\; \frac{\det H}{1 + \mathrm{sym}(C)}.$$

Proof Since $g \in C$, by definition we have $-\mathrm{sym}(C)g \in C$. By Lemma 7.1 we deduce

$$\frac{\det H}{\det H_+} \;=\; \frac{s^T(g - g_+)}{s^T g} \;=\; 1 + \frac{\max_C \langle \cdot, s\rangle}{-s^T g} \;\ge\; 1 + \frac{\langle -\mathrm{sym}(C)g, s\rangle}{-s^T g}.$$

The result follows. □

When the set C is simply the unit ball, Algorithm 6.2 becomes particularly simple. Numerical experiments suggest the following conjecture.

Conjecture 7.3 (BFGS for the unit ball) *Given any initial unit vector $g \in \mathbf{R}^n$ and matrix $H \in \mathbf{S}^n_{++}$, if we repeatedly set*

$$s = -Hg, \quad g_+ = \frac{s}{\|s\|}, \quad y = g_+ - g, \quad V = I - \frac{sy^T}{s^T y}, \quad H_+ = VHV^T + \frac{ss^T}{s^T y},$$

and update $g = g_+$ and $H = H_+$, then the trial step s converges to zero.

Figure 7 shows overlaid plots of $\|s\|$ against iteration count for a thousand randomly initiated runs in dimension $n = 5$. Such numerical results strongly suggest a linear convergence rate, and one that grows quite slowly with dimension n. Figure 8 plots against dimension n, on a log-log scale, the number of iterations (averaged over 200 random runs) to reduce $\|s\|$ by a factor 10^{-8} in Conjecture 7.3: the number grows roughly like $n^{1/\sqrt{2}}$.

As a first theoretical step we prove the following result. We denote the largest and smallest eigenvalues of H by $\lambda_{\max}(H)$ and $\lambda_{\min}(H)$ respectively, and we write $E \succ F$ for matrices $E, F \in \mathbf{S}^n$ to mean $E - F \in \mathbf{S}^n_{++}$.

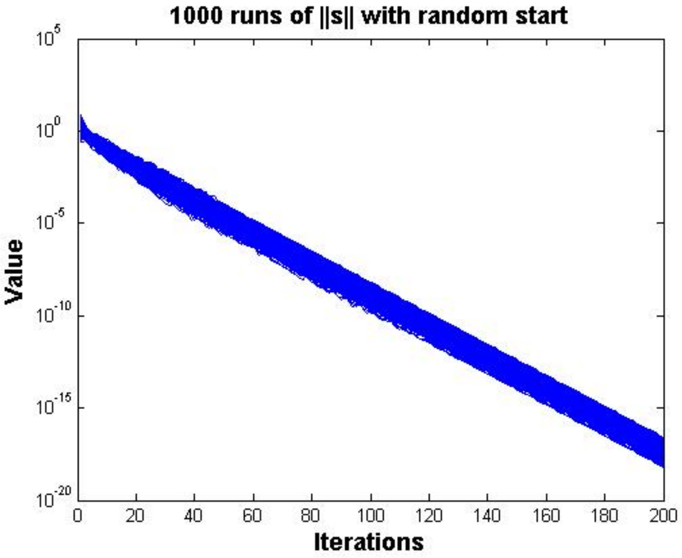

Figure 7: 1000 random runs of the iteration in Conjecture 7.3. Step size $\|s\|$ plotted against iteration count.

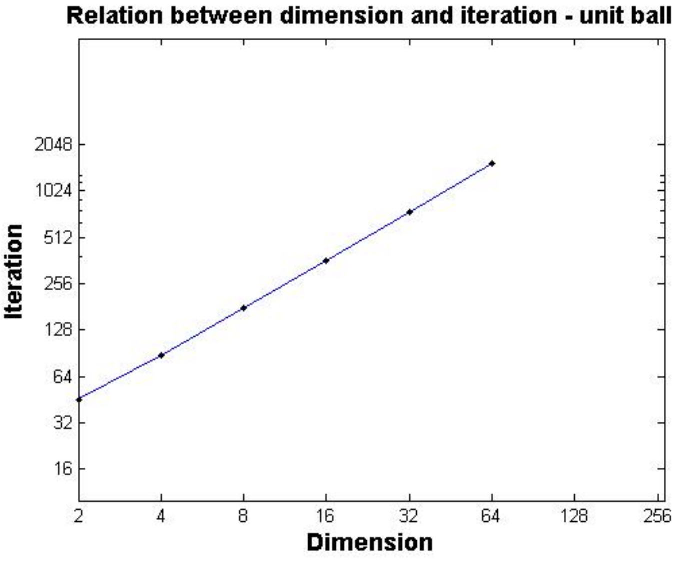

Figure 8: Mean number of iterations, over 200 random runs, to reduce $\|s\|$ by a factor 10^{-8} in Conjecture 7.3, plotted against dimension n.

Theorem 7.4 *The matrices H and H_+ in Conjecture 7.3 satisfy*

$$\det H_+ \le \frac{1}{2}\det H \quad \textit{and} \quad \lambda_{\max}(H_+) \le \lambda_{\max}(H).$$

Proof The first inequality follows from Proposition 7.2. Turning to the second, notice that the update of the matrix H to H_+ is positively homogeneous: if we replace H by γH for some positive scalar γ then H_+ is replaced by γH_+. After a suitable scaling of H, we can therefore assume that the step s is a unit vector. Similarly, if we replace H by $U^T H U$, for an orthogonal matrix U, and g by $U^T g$, then H_+ is replaced by $U^T H_+ U$. After such an orthogonal transformation, we can therefore also assume that s is just e_1, the first unit vector.

Partitioning vectors, we can write $s = [1\ 0]^T$ and $g = -[\alpha\ b]^T$, for some scalar $\alpha \in (0,1]$ and vector $b \in \mathbf{R}^{n-1}$ satisfying $\alpha^2 + \|b\|^2 = 1$. Since $s = -Hg$, we can also partition the matrix H^{-1} as

$$H^{-1} = \begin{bmatrix} \alpha & b^T \\ b & E \end{bmatrix},$$

where, using the Schur complement, the matrix $E \in \mathbf{S}^{n-1}$ satisfies $E \succ \frac{bb^T}{\alpha}$. We can write the BFGS update equivalently as

$$H_+^{-1} = H^{-1} + \frac{yy^T}{s^T y} + \frac{gg^T}{s^T g}$$

(see [15]), and we deduce

$$H_+^{-1} = \begin{bmatrix} 1+\alpha & b^T \\ b & E - \frac{bb^T}{\alpha(1+\alpha)} \end{bmatrix}.$$

For any positive scalar λ satisfying $\lambda < \lambda_{\min}(H^{-1}) = \big(\lambda_{\max}(H)\big)^{-1}$, we seek to show $\lambda < \lambda_{\min}(H_+^{-1}) = \big(\lambda_{\max}(H_+)\big)^{-1}$. Equivalently, via the Schur complement, we know

$$\alpha - \lambda > 0 \ \text{ and } \ E - \lambda I \succ \frac{bb^T}{\alpha - \lambda},$$

and we seek to prove

$$1 + \alpha - \lambda > 0 \ \text{ and } \ E - \frac{bb^T}{\alpha(1+\alpha)} - \lambda I \succ \frac{bb^T}{1+\alpha-\lambda}.$$

The first inequality is immediate. The second follows from the inequality

$$\frac{1}{\alpha - \lambda} \ge \frac{1}{\alpha(1+\alpha)} + \frac{1}{1+\alpha-\lambda},$$

which is equivalent to the inequality $\alpha(1+\alpha) \ge (\alpha-\lambda)(1+\alpha-\lambda)$, a consequence of the monotonicity of the left-hand side. □

8 Cholesky factors and line segments

The Shor r-algorithm (see equation (2.1)) uses the search direction $s = -V^TVg$, where g is a current subgradient. The analogue of the quasi-Newton matrix H is V^TV, and the method updates the factor V. As we commented at the end of Section 2, we can take a similar approach to the BFGS algorithm. Instead of updating the inverse Hessian approximation H directly through the BFGS formula (3.2), it can be useful (see [6, 16]) to update a factored form $H = T^TT$, where the matrix T is invertible. In that case we can write the update as $H_+ = T_+^TT_+$, where

$$T_+ = T(I - qs^T), \quad \text{for} \;\; q = \frac{y}{s^Ty} + \frac{g}{\sqrt{-s^Tgs^Ty}}.$$

Consider the BFGS algorithm for $0 \in C$, with this notation. After some algebra and the change of variables $h = Tg$, $p = Tg_+$ (where g_+ is the updated vector g), and $P = TC$, we arrive at the following algorithm for deciding whether or not zero lies in a compact convex set P.

Algorithm 8.1 (Cholesky BFGS for $0 \in P$)

```
  Choose $h \in P$;
  for $k = 0, 1, 2, \dots$ do
    if $h = 0$ then
      terminate with "$0 \in P$";
    end if
    Find a minimizer $p$ of $\langle \cdot, h \rangle$ over $P$;
    if $p^Th > 0$ then
      terminate with "$0 \notin P$";
    end if
    $e = h - p$; $\beta = h^Te$; $W = I - \frac{eh^T}{\beta} + \frac{hh^T}{\|h\|\sqrt{\beta}}$; $P = WP$; $h = Wp$;
  end for
```

As an example, consider the convex hull of nonzero vectors $a_i \in \mathbf{R}^n$ indexed by a finite set I. We can implement the algorithm above as follows.

Algorithm 8.2 (Cholesky BFGS for $0 \in \mathbf{conv}\{a_i : i \in I\}$)

```
  Choose $i \in I$;
  for $k = 0, 1, 2, \dots$ do
    Find $j \in I$ minimizing $a_i^Ta_j$;
    if $a_i^Ta_j > 0$ then
      terminate with "0 lies outside the convex hull";
    end if
    $e = a_i - a_j$; $\beta = a_i^Te$;
    for each $r \in I$ do
```

```
      $a_r = a_r - (a_i^T a_r)(\frac{e}{\beta} - \frac{a_i}{\|a_i\|\sqrt{\beta}})$;
    end for
    $i = j$;
  end for
```

To illustrate, consider how this method behaves for a set of just two distinct nonzero vectors $a_1 = c \neq d = a_2$ in $\mathbf{R}^n$. The algorithm becomes the following.

Algorithm 8.3 (Cholesky BFGS for $0 \in [c, d]$**)**

```
  for $k = 0, 1, 2, \ldots$ do
    if $c^T d > 0$ then
      terminate with "$0 \notin [c, d]$";
    end if
    $e = c - d$; $\beta = c^T e$;
    $d_+ = d - (c^T d)(\frac{e}{\beta} - \frac{c}{\|c\|\sqrt{\beta}})$; $c_+ = c - (c^T c)(\frac{e}{\beta} - \frac{c}{\|c\|\sqrt{\beta}})$;
    $c = d_+$; $d = c_+$;
  end for
```

We introduce a measure to track the conditioning of the line segments:

$$\gamma[c, d] \;=\; \frac{\sqrt{\|c\|^2\|d\|^2 - (c^T d)^2}}{\|c - d\|^2}.$$

This quantity is well-defined since the right-hand side is symmetric in c and d. It is also invariant under scaling and orthogonal transformations: $\gamma(\alpha[c, d]) = \gamma[c, d] = \gamma(U[c, d])$ for any nonzero scalar α and any n-by-n orthogonal matrix U.

Obviously the vectors in the algorithm all evolve in the two-dimensional space spanned by the original line segment $[c, d]$. Choosing a suitable basis, we therefore lose no generality in studying the special case $a_1 = c = [1\ 0]^T$ and $a_2 = d = [-p\ q]^T$ with $q \geq 0$, in which case we have

$$\gamma[c, d] = \frac{q}{(1 + p)^2 + q^2}. \tag{8.4}$$

We arrive at the following tool for recognizing when the algorithm will terminate.

Lemma 8.5 (Angle recognition) *Assuming the condition $c^T d \leq 0$, the line segment $[c, d]$ satisfies $\gamma[c, d] \leq \frac{1}{2}$. Under the additional assumption $0 \notin [c, d]$, the segment also satisfies $\gamma[c, d] > 0$.*

Proof Assuming the special case above, we have $p \geq 0$. We deduce

$$(1 + p)^2 + (q - 1)^2 \geq 1,$$

and the first claim now follows from equation (8.4). The second claim is easy. □

If the original line segment $[c, d]$ does not contain zero, eventually the termination criterion will hold, as a consequence of the following conditioning improvement.

Lemma 8.6 *If $c^T d \leq 0$, then $\gamma[c_+, d_+] \geq \gamma[c, d] + (\gamma[c, d])^3$.*

Proof We consider the special case above again, so by assumption, $p \geq 0$. Since (8.4) holds, in particular we have

$$\gamma[c, d] \leq \frac{q}{(1+p)^{3/4}}. \tag{8.7}$$

A quick calculation shows

$$\gamma[c_+, d_+] = \frac{q}{(1+p)^{3/2}}.$$

We deduce

$$\frac{\gamma[c_+, d_+]}{\gamma[c, d]} = \frac{(1+p)^2 + q^2}{(1+p)^{3/2}} = (1+p)^{1/2} + \frac{q^2}{(1+p)^{3/2}} \geq 1 + (\gamma[c, d])^2,$$

by inequality (8.7). The result follows. □

We can be more precise, using the following lemma.

Lemma 8.8 *For any $K > 0$, consider the finite sequence (β_k) defined (for integers $k \geq 0$) by $\beta_k = (K + 1 - k)^{-1/2}$ for all $k \leq K$. Suppose a second sequence (γ_k) satisfies $\gamma_0 \geq \beta_0$ and $\gamma_{k+1} \geq \gamma_k + \gamma_k^3$ for all $k \leq K - 1$. Then $\gamma_k \geq \beta_k$ for all $k \leq K$.*

Proof To prove the result by induction, we just need to show $\beta_{k+1} \leq \beta_k + \beta_k^3$ for all $k \leq K - 1$. Squaring both sides, we obtain

$$\beta_{k+1}^2 = \frac{\beta_k^2}{1 - \beta_k^2} \leq \beta_k^2 + 2\beta_k^4 + \beta_k^6,$$

or equivalently, $\beta_k^4 + \beta_k^2 \leq 1$. This last inequality is valid, since $\beta_k^2 \leq \frac{1}{2}$. □

Suppose the original segment $[c, d]$ does not contain zero, and denote the condition measure $\gamma[c, d]$ after $k = 0, 1, 2, \ldots$ iterations by γ_k. Then we have $\gamma_0 = \gamma[c, d]$ and $\gamma_{k+1} \geq \gamma_k + \gamma_k^3$ so we deduce by the previous lemma, $\gamma_k \geq (\gamma_0^{-2} - k)^{-1/2}$ for all $k \leq \gamma_0^{-2} - 1$. In particular, this inequality holds when k is the integral part $\lfloor \gamma_0^{-2} - 1 \rfloor$, if the algorithm has not already terminated. In that case, $k > \gamma_0^{-2} - 2$, so $\gamma_k \geq (\gamma_0^{-2} - k)^{-1/2} > 2^{-1/2} > \frac{1}{2}$, so the algorithm terminates.

We have proved the following result.

Theorem 8.9 *For any distinct nonzero vectors $c, d \in \mathbf{R}^n$, if the line segment $[c, d]$ does not contain zero, then, after a number of iterations not exceeding*

$$\frac{\|c - d\|^4}{\|c\|^2 \|d\|^2 - (c^T d)^2}$$

Algorithm 8.3 terminates correctly.

9 Conclusion

This work explores the relative effectiveness of the BFGS method and the Shor r-algorithm in the context of nonsmooth convex optimization. Incorporating line searches complicates the analysis, so here we try to separate their impact from the effect of the quasi-Newton or Shor update. In particular, we consider a simple linesearch-free BFGS algorithm.

Our experiments illustrate the effectiveness of improving the local metric in nonsmooth optimization. We focus especially on simple examples where the current subdifferential is a polytope, ball or ellipsoid, presenting both numerical and theoretical results. The algorithms simplify even further, conceptually, when rather than updating the approximate Hessian matrix, we instead work with its Cholesky factors. In summary, this exploration only heightens our appreciation for the mysterious power of the BFGS methodology.

References

[1] A. Belloni, R.M. Freund, and S. Vempala. An efficient rescaled perceptron algorithm for conic systems. *Math. Oper. Res.*, 34(3):621–641, 2009.

[2] R.G. Bland, D. Goldfarb, and M. J. Todd. The ellipsoid method: a survey. *Oper. Res.*, 29(6):1039–1091, 1981.

[3] J.M. Borwein and A.S. Lewis. *Convex Analysis and Nonlinear Optimization.* CMS Books in Mathematics, 3. Springer-Verlag, New York, 2000.

[4] S. Bubeck. Convex optimization: algorithms and complexity. *Foundations and Trends® in Machine Learning*, 8(3-4):231–357, 2015.

[5] J. V. Burke, A. S. Lewis, and M. L. Overton. The speed of Shor's r-algorithm. *IMA J. Numer. Anal.*, 28(4):711–720, 2008.

[6] D. Byatt, I. D. Coope, and C. J. Price. Performance of various BFGS implementations with limited precision second-order information. *ANZIAM J.*, 45(4):511–522, 2004.

[7] J.E. Dennis and J.J. Moré. Quasi-Newton methods, motivation and theory. *SIAM review*, 19(1):46–89, 1977.

[8] J. Dunagan and S. Vempala. A simple polynomial-time rescaling algorithm for solving linear programs. In *Proceedings of the 36th Annual ACM Symposium on Theory of Computing*, pages 315–320. ACM, New York, 2004.

[9] R.M. Freund. Complexity of convex optimization using geometry-based measures and a reference point. *Math. Program.*, 99(2, Ser. A):197–221, 2004.

[10] A. Griewank. Broyden updating, the good and the bad! *Doc. Math.*, Extra Volume (Optimization stories):301–315, 2012.

[11] J.-B. Hiriart-Urruty and C. Lemaréchal. *Convex Analysis and Minimization Algorithms. II*, volume 306 of *Grundlehren der Mathematischen Wissenschaften [Fundamental Principles of Mathematical Sciences]*. Springer-Verlag, Berlin, 1993.

[12] F. Kappel and A.V. Kuntsevich. An implementation of Shor's r-algorithm. *Comput. Optim. Appl.*, 15(2):193–205, 2000.

[13] A.S. Lewis and M.L. Overton. Nonsmooth optimization via quasi-Newton methods. *Math. Program.*, 141:135–163, 2013.

[14] Y. Nesterov and A. Nemirovskii. *Interior-Point Polynomial Algorithms in Convex Programming*, volume 13 of *SIAM Studies in Applied Mathematics*. Society for Industrial and Applied Mathematics (SIAM), Philadelphia, PA, 1994.

[15] J. Nocedal and S.J. Wright. *Numerical Optimization*. Springer Series in Operations Research and Financial Engineering. Springer, New York, second edition, 2006.

[16] M. J. D. Powell. Updating conjugate directions by the BFGS formula. *Math. Programming*, 38(1):29–46, 1987.

[17] J. Renegar. *A Mathematical View of Interior-Point Methods in Convex Optimization*. MPS/SIAM Series on Optimization. Society for Industrial and Applied Mathematics (SIAM), Philadelphia, PA; Mathematical Programming Society (MPS), Philadelphia, PA, 2001.

[18] N. Z. Shor. *Minimization Methods for Nondifferentiable Functions*, volume 3 of *Springer Series in Computational Mathematics*. Springer-Verlag, Berlin, 1985. Translated from the Russian by K. C. Kiwiel and A. Ruszczyński.