Randomized Hessian Estimation and Directional Search

D. Leventhal[*] A.S. Lewis[†]

September 24, 2008

Key words: derivative-free optimization, directional search, quasi-Newton, random search, steepest descent
AMS 2000 Subject Classification: 90C56, 90C53, 65K05

Abstract

We explore how randomization can help asymptotic convergence properties of simple directional search-based optimization methods. Specifically, we develop a cheap, iterative randomized Hessian estimation scheme. We then apply this technique and analyze how it enhances a random directional search method. Then, we proceed to develop a conjugate-directional search method that incorporates estimated Hessian information without requiring the direct use of gradients.

[*]School of Operations Research and Information Engineering, Cornell University, Ithaca, NY 14853, U.S.A. `leventhal@orie.cornell.edu`

[†]School of Operations Research and Information Engineering, Cornell University, Ithaca, NY 14853, U.S.A. `aslewis@orie.cornell.edu` `people.orie.cornell.edu/~aslewis`. Research supported in part by National Science Foundation Grant DMS-0504032.

1 Introduction and Motivation

Stochastic techniques in directional search algorithms have been well-studied in solving optimization problems, often where the underlying functions themselves are random or noisy. Some of these algorithms are based on directional search methods that obtain a random search direction which approximates a gradient in expectation: for some background on this class of algorithms, see [4, Ch. 6] or [13, Ch. 5]. For many algorithms, the broad convergence theory, combined with inherent computational simplicity, makes them particularly appealing, even for noiseless, deterministic optimization problems.

In this work, we avoid any direct use of gradient information, relying only on function evaluations. In that respect, the methods we consider have the flavor of derivative-free algorithms. Our goal, however, is not the immediate development of a practical, competitive, derivative-free optimization algorithm: our aim is instead primarily speculative. In contrast with much of the derivative-free literature, we make several impractical assumptions that hold throughout this paper. We assume that the function we seek to minimize is twice differentiable and that evaluations of that function are reliable, cheap, and accurate. Further, we assume that derivative information is neither available directly nor via automatic differentiation, but it is well-approximated by finite differencing. Again, for the purposes of speculation, we assume that any line search subproblem is relatively cheap to solve when compared to the cost of approximating a gradient. This last assumption is based on the fact that, asymptotically, the computational cost of a line search should be independent of the problem dimension, being a one-dimensional optimization problem, while the number of function evaluations required to obtain a gradient grows linearly with the problem dimension, based on previous assumptions. Within this narrow framework, we consider the question as to whether, in principle, randomization can be incorporated to help simple iterative algorithms achieve good asymptotic convergence.

Keeping this narrow framework in mind, this paper is organized as follows. In the remainder of this section, we consider a randomized directional search algorithm that chooses a search direction uniformly at random from the unit sphere and apply it to convex quadratic functions, comparing convergence results with a traditional gradient descent algorithm. In Section 2, we introduce a technique of randomized Hessian estimation and prove some basic properties. In Section 3, we consider algorithmic applications of our randomized Hessian estimation method. In particular, we show how Hessian estimates can be used to accelerate the uniformly random search algorithm introduced in this section and, additionally, how randomized Hessian estimation can also be used to develop a conjugate direction-like algorithm.

To illustrate the use of randomization, consider the following basic algorithm: at each iteration, choose a search direction uniformly at random on the unit sphere and perform an exact line search. Note that this algorithm has been widely studied, with analysis appearing in [5] and [12], among others. Further, it was shown to be linearly convergent for twice differentiable functions under conditions given in [11].

Consider applying this algorithm to the problem of minimizing a convex quadratic function $f(x) = x^T Ax$ where A is a symmetric, positive-definite, $n \times n$ matrix. If the current iterate is x, then the new iterate is given by

$$x_+ = x - \frac{d^T Ax}{d^T Ad} d \tag{1.1}$$

and the new function value is

$$f(x_+) = f(x) - \frac{(d^T Ax)^2}{d^T Ad}.$$

Assuming $x \neq 0$ (which is the optimal solution), the function value is reduced by the ratio

$$\begin{aligned}
\frac{f(x_+)}{f(x)} &= 1 - \frac{(d^T Ax)^2}{(d^T Ad)(x^T Ax)} \\
&= 1 - \frac{(d^T Ax)^2}{(d^T Ad)((Ax)^T A^{-1}(Ax))} \\
&\leq 1 - \frac{1}{\kappa(A)}\Big(d^T \frac{Ax}{\|Ax\|}\Big)^2,
\end{aligned}$$

where $\kappa(A) = \|A\|\|A^{-1}\|$ denotes the condition number of A. Since the distribution of d is invariant under orthogonal transformations, we have

$$\mathbf{E}\big[\Big(d^T \frac{Ax}{\|Ax\|}\Big) \mid x\big] = \mathbf{E}[d_1^2] = \frac{1}{n}\mathbf{E}\big[\Big(\sum_i d_i^2\Big)\big] = \frac{1}{n},$$

by symmetry. We deduce

$$\mathbf{E}[f(x_+) \mid x] \le \Big(1 - \frac{1}{n\kappa(A)}\Big) f(x) \tag{1.2}$$

with equality when A is a multiple of the identity matrix, in which case $\kappa(A) = 1$.

Compare this with the steepest descent algorithm. A known result about the steepest descent algorithm in [1] says that, given initial iterate x and letting $\hat{x}$ be the new iterate constructed from an exact line search in the negative gradient direction,

$$f(\hat{x}) \le \Big(\frac{\kappa(A) - 1}{\kappa(A) + 1}\Big)^2 f(x) = \Big(1 - O(\frac{1}{\kappa(A)})\Big) f(x).$$

Further, for most initial iterates x, this inequality is asymptotically tight if this procedure is iteratively repeated. Consider the following asymptotic argument, applying the assumptions made earlier in this section. Suppose derivative information is only available through–and well-approximated by–finite differencing but we can perform an exact (or almost-exact) line search in some constant number, $O(1)$, of function evaluations. It follows that each iteration of random search takes $O(1)$ function evaluations. However, since derivative information is only available via finite differencing, computing a gradient takes $O(n)$ function evaluations. Letting $\bar{x}$ be the iterate after performing $O(n)$ iterations of random search, we obtain that

$$E\Big[\frac{f(\bar{x})}{f(x)} \mid x\Big] \le \Big(1 - \frac{1}{n\kappa(A)}\Big)^{O(n)} = 1 - O(\frac{1}{\kappa(A)}).$$

Essentially, the expected improvement of random search is on the same order of magnitude as steepest descent when measured on a cost per function evaluation basis. This simple example suggests that randomization techniques may be an interesting ingredient in the design and analysis of iterative optimization algorithms.

2 Randomized Hessian Estimation

In this section, we will consider arbitrary twice-differentiable functions $f : \mathbf{R}^n \to \mathbf{R}$. As in the previous section, assume these functions can be evaluated exactly, but derivative information is only available through finite differencing. In particular, for any vector $v \in \mathbf{R}^n$, suppose we can use finite differencing to well-approximate the second derivative of f at x in the direction v via the formula

$$v^T \nabla^2 f(x) v \approx \frac{f(x+\epsilon v) - 2f(x) + f(x-\epsilon v)}{\epsilon^2} \tag{2.1}$$

for some sufficiently small $\epsilon > 0$. In particular, note that by choosing $\frac{1}{2}n(n+1)$ suitable directions v, we could effectively approximate the entire Hessian $\nabla^2 f(x)$.

In Section 1, we considered a framework in which computational costs of an algorithm are measured by the number of function evaluations required and we will continue with that throughout the paper. In particular, it was shown that under this framework, the steepest descent algorithm, asymptotically, achieves improvement on the same order of magnitude as a uniformly random search algorithm when applied to convex quadratics. Ideally, we would like to extend these methods of analysis to algorithms that incorporate additional information about a function's behavior. For example, instead of calculating a complete Hessian matrix at each iteration, Newton-like methods rely on approximations to the Hessian matrix which are iteratively updated, often from successively generated gradient information. To consider a similar approach in the context of random search, suppose we begin with an approximation to the Hessian matrix, denoted B, and some unit vector $v \in \mathbf{R}^n$. Consider the new matrix B_+ obtained by making a rank-one update so that the new matrix B_+ matches the true Hessian in the direction v, i.e.,

$$B_+ = B + (v^T(\nabla^2 f(x) - B)v)vv^T. \tag{2.2}$$

This rank-one update results in the new matrix B_+ having the property that $v^T B_+ v = v^T \nabla^2 f(x) v$. Note that if this update is performed using the approximate second derivative via Equation 2.1, then this only costs 3 function evaluations. The following result on the space of symmetric $n \times n$

matrices, $\mathbf{S}^n$, equipped with the usual trace inner product and the induced Frobenius norm, will be our primary tool.

Theorem 2.3 *Given any matrices $H, B \in \mathbf{S}^n$, if the random vector $d \in \mathbf{R}^n$ is uniformly distributed on the unit sphere, then the matrix*

$$B_+ = B + (d^T(H-B)d)dd^T$$

satisfies

$$\|B_+ - H\| \le \|B - H\|$$

and

$$E[\|B_+ - H\|^2] \le \Big(1 - \frac{2}{n(n+2)}\Big)\|B - H\|^2.$$

Proof Since we can rewrite the update in the form

$$(B_+ - H) = (B - H) - \Big(d^T(B-H)d\Big)dd^T,$$

we lose no generality in assuming $H = 0$. Again, we lose no generality in assuming $\|B\| = 1$, and proving

$$\|B_+\| \le 1 \ \text{ and } \ \mathbf{E}[\|B_+\|^2] \ \le \ 1 - \frac{2}{n(n+2)}.$$

From the equation

$$B_+ = B - (d^T Bd)dd^T,$$

we immediately deduce

$$\|B_+\|^2 = \|B\|^2 - (d^T Bd)^2 = 1 - (d^T Bd)^2 \le 1.$$

To complete the proof, we need to bound the quantity $\mathbf{E}[(d^T Bd)^2]$. We can diagonalize the matrix $B = U^T(\text{Diag }\lambda)U$ where U is orthogonal and the vector of eigenvalues $\lambda \in \mathbf{R}^n$ satisfies $\|\lambda\| = 1$ by assumption. Using the fact that the distribution of d is invariant under orthogonal transformations, we obtain

$$\begin{aligned}
\mathbf{E}[(d^T Bd)^2] &= \mathbf{E}[(d^T U^T(\text{Diag }\lambda)Ud)^2] = \mathbf{E}[(d^T(\text{Diag }\lambda)d)^2] \\
&= \mathbf{E}[(\sum_{i=1}^n \lambda_i d_i^2)^2] = \mathbf{E}[\sum_i \lambda_i^2 d_i^4 + \sum_{i \ne j} \lambda_i \lambda_j d_i^2 d_j^2] \\
&= \mathbf{E}[d_1^4] + (\sum_{i \ne j} \lambda_i \lambda_j)\mathbf{E}[d_1^2 d_2^2]
\end{aligned}$$

by symmetry. Since we know that

$$0 \le (\sum_i \lambda_i)^2 = \sum_i \lambda_i^2 + \sum_{i \ne j} \lambda_i \lambda_j = 1 + \sum_{i \ne j} \lambda_i \lambda_j,$$

it follows that

$$\mathbf{E}[(d^T B d)^2] \ge \mathbf{E}[d_1^4] - \mathbf{E}[d_1^2 d_2^2].$$

Standard results on integrals over the unit sphere in $\mathbf{R}^n$ gives the formula

$$\int_{\|x\|=1} x_1^\nu \, d\sigma \;=\; 2\pi^{\frac{n-1}{2}} \frac{\Gamma\left(\frac{\nu+1}{2}\right)}{\Gamma\left(\frac{\nu+n}{2}\right)},$$

where $d\sigma$ denotes an $(n-1)$-dimensional surface element, and Γ denotes the Gamma function. We deduce

$$\mathbf{E}(d_1^4) \;=\; \frac{\int_{\|x\|=1} x_1^4 \, d\sigma}{\int_{\|x\|=1} d\sigma} = \frac{\Gamma\left(\frac{5}{2}\right)}{\Gamma\left(\frac{n}{2}+2\right)} \cdot \frac{\Gamma\left(\frac{n}{2}\right)}{\Gamma\left(\frac{1}{2}\right)} = \frac{\frac{3}{2} \cdot \frac{1}{2}}{\left(\frac{n}{2}+1\right) \cdot \frac{n}{2}} = \frac{3}{n(n+2)}.$$

Furthermore,

$$1 = \Big(\sum_i d_i^2\Big)^2 = \sum_i d_i^4 + \sum_{i \ne j} d_i^2 d_j^2,$$

so using symmetry again shows

$$1 = n\mathbf{E}[d_1^4] + n(n-1)\mathbf{E}[d_1^2 d_2^2].$$

From this we deduce

$$\mathbf{E}[d_1^2 d_2^2] = \frac{1 - n\mathbf{E}[d_1^4]}{n(n-1)} = \frac{1}{n(n+2)}.$$

Therefore, this shows that

$$\mathbf{E}[(d^T B d)^2] \ge \frac{3}{n(n+2)} - \frac{1}{n(n+2)} = \frac{2}{n(n+2)},$$

so

$$\mathbf{E}[\|B_+\|^2] \le 1 - \frac{2}{n(n+2)}$$

as required. $\square$

To continue, note that iterating this procedure generates a sequence of Hessian approximations that converges almost surely to the true Hessian, as shown next.

Corollary 2.4 *Given any matrices $H, B_0 \in \mathbf{S}^n$, consider the sequence of matrices $B_k \in \mathbf{S}^n$ for $k = 0, 1, 2, \ldots$, defined iteratively by*

$$B_{k+1} = B_k + \Big((d^k)^T (H - B_k) d^k\Big) d^k (d^k)^T,$$

where the random vectors $d^0, d^1, d^2, \ldots \in \mathbf{R}^n$ are independent and uniformly distributed on the unit sphere. Then the errors $\|B_k - H\|$ decrease monotonically, and $B_k \to H$ almost surely.

Proof Again we lose no generality in assuming $H = 0$. By the previous result, it follows that the random variables $\|B_k\|^2$ form a supermartingale which is bounded above by $\|B_0\|^2$. By classical supermartingale convergence results (see [2], for example), it follows that $\|B_k\|^2 \to Y$ almost surely for some random variable $Y \geq 0$. It remains to be shown that $Y = 0$ almost everywhere.

Clearly the random variables B_k are uniformly bounded. Furthermore, if we define a measure on the space of symmetric $n \times n$ matrices, γ_k, by

$$\gamma_k(U) = \mathrm{pr}\{B_k \in U\}$$

for any measurable set U, then the previous result implies

$$\begin{aligned} \mathbf{E}[\|B_{k+1}\|^2] &= \int \mathbf{E}\Big[\|B_{k+1}\|^2 \mid B_k = X\Big]\, d\gamma_k(X) \\ &\leq \int \Big(1 - \frac{2}{n(n+2)}\Big) \|X\|^2\, d\gamma_k(X) \\ &= \Big(1 - \frac{2}{n(n+2)}\Big) \mathbf{E}[\|B_k\|^2]. \end{aligned}$$

Now by monotone convergence we have

$$\mathbf{E}[Y] = \mathbf{E}[\lim_k \|B_k\|^2] = \lim_k \mathbf{E}[\|B_k\|^2] = 0,$$

so $Y = 0$ almost everywhere as required. $\square$

In a more realistic framework for optimization, we wish to approximate a limiting Hessian. Consider a sequence of points $x_k \in \mathbf{R}^n$ converging to some limit $\bar{x}$, and suppose we use the above technique to iteratively update the approximation B_k to the Hessian $\nabla^2 f(x_k)$. The next result shows convergence of the Hessian approximations B_k to the limiting Hessian, $\nabla^2 f(\bar{x})$.

Theorem 2.5 *Consider a sequence of random matrices $H_k \in \mathbf{S}^n$ for $k = 1,2,3,\ldots$, with each $\mathbf{E}(\|H_k\|^2)$ finite, and a matrix $\bar{H} \in \mathbf{S}^n$ such that $\mathbf{E}(\|H_k - \bar{H}\|^2) \to 0$. Consider a sequence of random matrices $B_k \in \mathbf{S}^n$ for $k = 0,1,2,\ldots$, with $\mathbf{E}(\|B_0\|^2)$ finite, related by the iterative formula*

$$B_{k+1} = B_k + \Big((d^k)^T(H_k - B_k)d^k\Big)d^k(d^k)^T,$$

where the random vectors $d^0, d^1, d^2, \ldots \in \mathbf{R}^n$ are independent and uniformly distributed on the unit sphere. Then $\mathbf{E}(\|B_k - \bar{H}\|^2) \to 0$.

Proof By Corollary 2.4, we know for each $k = 0,1,2,\ldots$ the inequality

$$\|B_{k+1} - H_k\|^2 \le \|B_k - H_k\|^2$$

holds. Hence by induction it follows that $\mathbf{E}(\|B_k\|^2)$ is finite for all $k \ge 0$. We can therefore work with the L_2-norm on the space of random matrices defined by $\|X\|_2 = \sqrt{\mathbf{E}(\|X\|^2)}$.

Define a number

$$r \;=\; \sqrt{1 - \frac{2}{n(n+2)}} \;\in\; (0,1).$$

By Theorem 2.3, we have

$$\mathbf{E}[\|B_{k+1} - H_k\|^2 \mid B_k, H_k] \;\le\; r^2\|B_k - H_k\|^2.$$

Once again, define a measure γ_k by

$$\gamma_k(S) = \mathrm{pr}\{(B_k, H_k) \in S\}$$

for any measurable set S. Then we have

$$\begin{aligned}
\|B_{k+1} - H_k\|_2^2 &= \mathbf{E}[\|B_{k+1} - H_k\|^2] \\
&= \int \mathbf{E}[\|B_{k+1} - H_k\|^2 \mid (B_k, H_k) = (B, H)]d\gamma_k(B, H) \\
&\le \int \|B - H\|^2 \, d\gamma_k(B, H) \\
&= r^2\mathbf{E}[\|B_k - H_k\|^2] \\
&= r^2\|B_k - H_k\|_2^2.
\end{aligned}$$

Applying the triangle inequality gives

$$\|B_{k+1} - \bar{H}\|_2 \leq r\|B_k - \bar{H}\|_2 + (1+r)\|H_k - \bar{H}\|_2.$$

Now fix any number $\epsilon > 0$. By assumption, there exists an integer $\bar{k}$ such that for all integers $k \geq \bar{k}$ we have

$$\|H_k - \bar{H}\|_2 \leq \frac{\epsilon(1-r)}{2(1+r)}.$$

Hence, for all $k \geq \bar{k}$, we deduce

$$\|B_{k+1} - \bar{H}\|_2 \leq r\|B_k - \bar{H}\|_2 + \frac{\epsilon(1-r)}{2}.$$

For such k, if $\|B_k - \bar{H}\|_2 \leq \epsilon$, then

$$\|B_{k+1} - \bar{H}\|_2 \leq \frac{\epsilon(1+r)}{2} < \epsilon,$$

whereas if $\|B_k - \bar{H}\|_2 > \epsilon$, then

$$\|B_{k+1} - \bar{H}\|_2 < r\|B_k - \bar{H}\|_2 + \frac{1-r}{2}\|B_k - \bar{H}\|_2 = \frac{1+r}{2}\|B_k - \bar{H}\|_2.$$

Consequently, $\|B_k - \bar{H}\|_2 \leq \epsilon$ for all large k. Since $\epsilon > 0$ was arbitrary, the result follows. □

3 Applications to Algorithms

Random Search, Revisited

Return to the convex quadratic function $f(x) = x^T Ax$ considered in Section 1 where A is a positive definite, $n \times n$ matrix. Recall that if we consider the iterative algorithm given by Equation 1.1, letting d be a unit vector uniformly distributed on the unit sphere and letting

$$x_+ = x - \frac{d^T Ax}{d^T Ad} d,$$

then it was shown in Inequality 1.2 that

$$\mathbf{E}[f(x_+) \mid x] \leq \Big(1 - \frac{1}{n\kappa(A)}\Big) f(x).$$

Now, suppose that H is a positive-definite estimate of the matrix A and consider the Cholesky factor matrix C such that $CC^T = H^{-1}$. Suppose that instead of performing an exact line search in the uniformly distributed direction d, we instead perform the line search in the direction Cd. From this we obtain

$$\begin{aligned}
\frac{f(x_+)}{f(x)} &= 1 - \frac{(d^T C^T A x)^2}{(d^T C^T A C d)(x^T A x)} \\
&= 1 - \frac{\Big(d^T (C^T A x)\Big)^2}{(d^T (C^T A C) d)\left((C^T A x)^T (C^T A C)^{-1} (C^T A x)\right)} \\
&\leq 1 - \frac{1}{\kappa(C^T A C)} \left(d^T \frac{C^T A x}{\|C^T A x\|}\right)^2,
\end{aligned}$$

allowing us to conclude that

$$\mathbf{E}[f(x_+) \mid x] \leq \Big(1 - \frac{1}{n\kappa(C^T A C)}\Big) f(x).$$

This provides the same convergence rate as performing the random search algorithm given by Equation 1.1 on the function $g(x) = x^T (C^T A C) x$.

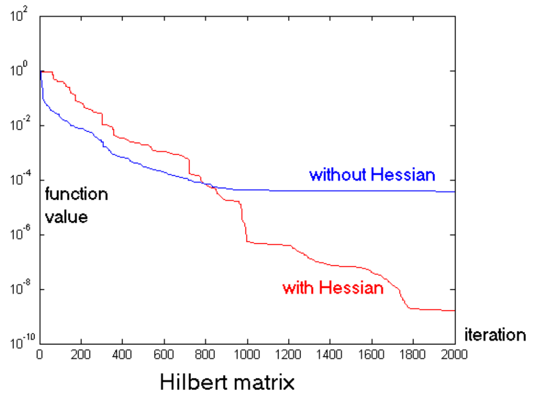
Consider an implementation of this algorithm using the Hessian approximation technique described in Section 2. Given a current iterate x_{k-1} and Hessian approximation B_{k-1}, we can proceed as follows. First, form the new Hessian approximation B_k given by Equation 2.2. Then, if B_k is positive-definite, obtain the Cholesky factorization $B_k^{-1} = C_k C_k^T$; otherwise, we can obtain the projection of B_k onto the positive semi-definite cone, denoted B_k^+, and perform the Cholesky factorization $C_k C_k^T = (B_k^+ + \epsilon I)^{-1}$ for some $\epsilon > 0$. Finally, we can find the next iterate x_k by a line search in the direction $C_k d_k$ where d_k is uniformly distributed on the unit sphere.

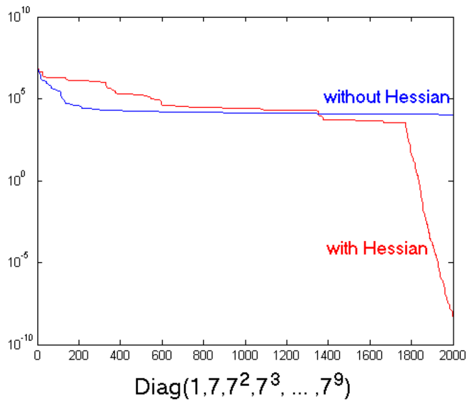
By Corollary 2.4, it follows that $B_k \longrightarrow A$ almost surely and therefore $C_k \longrightarrow A^{-\frac{1}{2}}$ almost surely as well. Therefore, it follows that

$$\frac{\mathbf{E}[f(x_{k+1}) \mid x_k]}{f(x_k)} \leq 1 - \frac{1}{n\kappa(C_k^T A C_k)} \longrightarrow 1 - \frac{1}{n}.$$

Thus, the uniformly random search algorithm incorporating the Hessian update provides linear convergence with asymptotic rate $1 - \frac{1}{n}$, *independent of the conditioning of the original matrix.*

Below are two examples of the algorithm's behavior with a convex quadratic function. The first example uses a Hilbert Matrix of size 10 (with condition number on the order of 10^{13}) while the second uses the matrix $A = Diag(1, 7, 7^2, \ldots, 7^9)$. In each case, the initial iterate is random and $B_0 = I$.





Random directional search on ill-conditioned convex quadratics in 10 variables, with and without Hessian approximation.

A Conjugate Directions Algorithm

Coordinate descent algorithms have a long and varied history in differentiable minimization. In the worst case, examples of continuously differentiable functions exist in [10] where a coordinate descent algorithm will fail to converge to a first-order stationary point. On the other hand, for twice-differentiable, strictly convex functions, variants of coordinate descent methods were shown to be linearly convergent in [7]. In either case, the simplicity of such algorithms, along with the lack of a need for gradient information, often makes them appealing.

Let us briefly return to the example of a convex quadratic function $f(x) = \frac{1}{2}x^T Ax + b^T x$. Consider algorithms, similar to coordinate descent algorithms, that choose search directions by cycling through some fixed set $W = \{w_1, \ldots, w_n\}$, performing an exact line search at each iteration. If the search directions in W happen to be A-conjugate, satisfying $w_i^T Aw_j = 0$ for all $i \neq j$, then we actually reach the optimal solution in n iterations. Alternatively, if our set of search directions fails to account for the function's second-order behavior, convergence can be significantly slower. Explicitly generating a set of directions that are conjugate with respect to the Hessian requires knowledge of the function's Hessian information. Methods were proposed in [9], and expanded upon in [14], [3], and [8] among others, that begin as coordinate descent algorithms and iteratively adjust the search directions, gradually making them conjugate with respect to the Hessian matrix. Further, these adjustments are based on the results of previous line searches without actually requiring full knowledge of the Hessian or any gradients.

We propose an alternative approach for arbitrary twice-differentiable functions. If an estimate of the Hessian were readily available, we could take advantage of it by generating search directions iteratively that are conjugate with respect to the estimate. This suggests that we can design an algorithm using the Hessian estimation technique in Section 2 to dynamically generate new search directions that have the desired conjugacy properties. We can formalize this in the following algorithm.

Algorithm 3.1 *Let f be a twice-differentiable function, x_0 an initial starting point, B_0 and initial Hessian estimate and $\{v_{-n}, v_{-(n-1)}, \ldots, v_{-1}\}$ an initial set of search directions. For $k = 0, 1, 2, \ldots$*

1. *Compute the vector v_k that is B_k-conjugate to $v_{k-1}, \ldots, v_{k-n+1}$.*
2. *Compute x_{k+1} as a result of a (two-way) line search in the direction v_k.*
3. *Compute B_{k+1} according to Equation 2.2, letting d_k be uniformly distributed on the unit sphere and computing*

$$B_{k+1} = B_k + (d_k(\nabla^2 f(x_{k+1}) - B_k)d_k)d_k d_k^T.$$

One simple initialization scheme takes $B_0 = I$ and $\{v_{-n}, \ldots, v_{-1}\} = \{e_1, \ldots, e_n\}$, the standard basis vectors.

Since B_k is our Hessian approximation at the current iterate x_k, a reasonable initial step size is given by $x_{k+1} = x_k - t_k v_k$, where $t_k = \frac{v_k^T \nabla f(x_k)}{v_k^T B_k v_k}$, corresponding to an exact line search in the direction v_k of the quadratic model. The advantage to this approach is that each iteration requires only directional derivatives and, being highly iterative, this interpolates nicely with the Hessian update derived in Section 2. Specifically, when using the fixed step size mentioned above, each iteration takes exactly five function evaluations: $f(x_k), f(x_k \pm \epsilon v_k)$, and $f(x_k \pm \epsilon d_k)$ where v_k and d_k are the search direction and the random unit vector, respectively.

The essence of this algorithm lies in using our randomized Hessian estimation technique to update a quadratic model and then performing a line search. Since we are relying solely on function evaluations, this algorithm has the "flavor" of derivative-free optimization. However, it should be noted that a different perspective can be taken with regards to this algorithm, permitting a comparison with Newton-like methods.

Typical Newton-like methods maintain, along with the current iterate x_k, a (positive-definite) Hessian estimate B_k and proceed by performing some type of line search in the direction $-B_k^{-1} \nabla f(x_k)$. For simplicity, consider a step size of 1, i.e., $x_{k+1} = x_k - B_k^{-1} \nabla f(x_k)$. Recall that computing $B_k^{-1} \nabla f(x_k)$, equivalent to solving the system $B_k y = \nabla f(x_k)$ for y, can be done indirectly by searching in n different B_k-conjugate directions.

Specifically, suppose we have a set of directions $\{v_1, \ldots, v_n\}$ that are B_k-conjugate, satisfying $v_i^T B_k v_j = 0$ for all $i \neq j$. and take $x^0 = x_k$, our current iterate. For $i = 1, \ldots, n$, let $x^i = x^{i-1} - \frac{v_i^T \nabla f(x_k)}{v_i^T B_k v_i} v_i$. Then it follows that

$$x^n = x^0 - \sum_{i=1}^{n} \frac{v_i^T \nabla f(x_k)}{v_i^T B_k v_i} v_i = x_k - B_k^{-1} \nabla f(x_k),$$

the Newton-like step. Given this interpretation of Newton-like methods, consider the version of Algorithm 3.1 where, at each iteration, the step size is fixed beforehand at $t_k = \frac{v_k^T \nabla f(x_k)}{v_k^T B_k v_k}$. Then one can interpret Algorithm 3.1 as an iterated version of a Newton-like method. Specifically, while the Newton-like method *indirectly* involves computing the quantities $v_i^T \nabla f(x_k)$

and $v_i^T B_k v_i$ with the iterate x_k and Hessian estimate B_k fixed, Algorithm 3.1 allows for a dynamically changing gradient and Hessian approximation at each conjugate direction step.

Given this connection between Algorithm 3.1 and traditional Newton-like methods, it seems natural to expect superlinear convergence under similar assumptions. As we demonstrate in the remainder of this section, superlinear convergence is obtained for strictly convex quadratic functions. Before doing so, we define some notation. For a matrix, A, the *spectral norm* of A is the quantity $\|A\|_2 := \max_{\|x\|=1\|} \|Ax\|$ and the *Frobenius norm* of A is given by $\|A\|_F := \sqrt{\sum_{i,j} a_{ij}^2}$. If A is invertible, then $\|A^{-1}\|_2$ can also be expressed as the smallest constant K such that $\|Ax\|_2 \geq \frac{1}{K}\|x\|_2$ for all vectors x. Let $\mathrm{Diag}(A)$ be the matrix whose diagonal matches that of A whose non-diagonal entries are zero. For positive-definite A, the *energy norm* of x is defined by $\|x\|_A := \sqrt{x^T A x}$. Additionally, these norms satisfy

$$\|A\|_2 \leq \|A\|_F \leq \sqrt{n}\|A\|_2, \tag{3.2}$$

$$\|Ax\|^2 \leq \lambda_{\max}(A)\|x\|_A^2 \leq \lambda_{\max}(A)^2\|x\|_2^2 \tag{3.3}$$

and

$$\lambda_{\min}(A)\|x\|_2^2 \leq \|x\|_A^2 \tag{3.4}$$

where $\lambda_{\max}(A)$ and $\lambda_{\min}(A)$ are the maximum and minimum eigenvalues of A, respectively. For a strictly convex quadratic function $f(x) = \frac{1}{2}x^T Ax + b^T x$ with minimizer $x^* = -A^{-1}b$, the energy norm satisfies

$$\|x - x^*\|_A^2 = f(x) - f(x^*). \tag{3.5}$$

Theorem 3.6 *Consider the strictly convex quadratic function $f(x) = \frac{1}{2}x^T Ax + b^T x$ where A is a positive definite matrix. Then for any initial point x_0, initial Hessian estimate B_0 and initial search directions, Algorithm 3.1 is $n-$step superlinearly convergent when implemented with an exact line search.*

Proof Define $\epsilon_k = \|B_k - A\|_F$ and note that, by Inequality 3.2, $\|B_k - A\|_2 \leq \epsilon_k$. Now consider n consecutive iterations of the algorithm, beginning at iterate x_k and ending at iterate x_{k+n}. Without loss of generality, assume the respective search directions satisfy $\|v_i\| = 1$ for $i = k, k+1, \dots, k+n-1$. Recall that by design of the algorithm, these search directions satisfy $v_i^T B_j v_j = 0$ for any $j \in \{k, k+1, \dots, k+n-1\}$ and $i \in \{j-n+1, \dots, j-1\}$. Note that this implies that for any $i < j \in \{k, k+1, \dots, k+n-1\}$,

$$|v_i^T A v_j| = |v_i^T B_j v_j + v_i^T (A - B_j) v_j| \leq \|v_i\| \|A - B_j\|_2 \|v_j\| \leq \epsilon_k \tag{3.7}$$

by Inequality 3.2 and the definition of ϵ_k.

Next, we construct a matrix M_k such that these search directions are M_k-conjugate and $\|A - M_k\| = O(\epsilon_k)$. Let $V_k = [v_k, v_{k+1}, \dots, v_{k+n-1}]$ be the matrix whose columns are the n consecutive search directions. First, notice that if ϵ_k is sufficiently small, this matrix is invertible and the quantity $\|V_k^{-1}\|_2$ is uniformly bounded. To see this, consider any $y \in \mathbf{R}^n$ such that $\|y\| = 1$. Then,

$$\begin{aligned}
\|V_k y\|_2^2 &= y^T V_k^T V_k y \\
&= (y^T V_k^T A^{\frac{1}{2}}) A^{-1} (A^{\frac{1}{2}} V_k y) \\
&= \|A^{\frac{1}{2}} V^T y\|_{A^{-1}}^2 \\
&\geq \lambda_{\min}(A^{-1}) \|A^{\frac{1}{2}} V_k y\|^2 \quad \text{(by Inequality 3.4)} \\
&= \frac{1}{\lambda_{\max}(A)} y^T V_k^T A V_k y \\
&= \frac{1}{\lambda_{\max}(A)} [y^T \mathrm{Diag}(V_k^T A V_k) y + y^T [V_k^T A V_k - \mathrm{Diag}(V_k^T A V_k)] y] \\
&\geq \frac{\lambda_{\min}(A) - \sqrt{n}\epsilon_k}{\lambda_{\max}(A)},
\end{aligned}$$

with the last inequality coming from the fact that

$$y^T \mathrm{Diag}(V_k^T A V_k) y = \sum_{i=1}^{n} y_i^2 v_{k+i-1}^T A v_{k+i-1},$$

Inequality 3.4 and Inequality 3.7. From the above bound and the alternative definition of $\|V_k^{-1}\|_2$, it follows that

$$\|V_k^{-1}\|_2^2 \leq \frac{\lambda_{\max}(A)}{\lambda_{\min}(A) - \sqrt{n}\epsilon_k}. \tag{3.8}$$

Now consider the matrix M_k defined by

$$M_k = A - V_k^{-T}(V_k^T A V_k - \mathrm{Diag}(V_k^T A V_k))V_k^{-1} = V_k^{-T}\mathrm{Diag}(V_k^T A V_k)V_k^{-1}.$$

Further, observe that

$$\begin{aligned}\|A - M_k\|_2 &= \|V_k^{-T}(V_k^T A V_k - \mathrm{Diag}(V_k^T A V_k))V_k^{-1}\|_2 \\ &\leq \|V_k^{-1}\|_2^2 \|(V_k^T A V_k - \mathrm{Diag}(V_k^T A V_k))\|_F \\ &\leq \|V_k^{-1}\|_2^2 n\epsilon_k,\end{aligned}$$

with the first inequality coming from the triangle inequality for the spectral norm, the fact that the spectral norm is invariant under matrix transposition and Inequality 3.2 while the last inequality comes from Inequality 3.7. In particular, the matrix M_k satisfies $\|A - M_k\|_2 = O(\epsilon_k)$ and, for $i \neq j \in \{k, k+1, \ldots, k+n-1\}$, both $v_i^T M_k v_i = v_i^T A v_i$ and $v_i^T M_k v_j = 0$.

At each iteration $i = k, k+1, \ldots, k+n-1$, Algorithm 3.1 obtains the new point by way of exact line search, getting

$$\begin{aligned}x_{i+1} &= x_i - \frac{v_i^T(Ax_i + b)}{v_i^T A v_i} v_i \\ &= x_i - \frac{v_i^T(Ax_k + b + \sum_{j=k}^{i-1}\alpha_j A v_j)}{v_i^T A v_i} v_i \\ &= x_i - \frac{v_i^T(Ax_k + b)}{v_i^T A v_i} v_i - \frac{\sum_{j=k}^{i-1}\alpha_j v_i^T A v_j}{v_i^T A v_i} v_i,\end{aligned}$$

where α_j is defined by $\alpha_j = -\frac{v_j^T \nabla f(x_j)}{v_j^T A v_j}$. Expanding this out over n consecutive iterations, we obtain

$$x_{k+n} = x_k - \sum_{i=k}^{k+n-1} \frac{v_i^T \nabla f(x_k)}{v_i^T A v_i} v_i - \sum_{i=k}^{k+n-1}\sum_{j=k}^{i-1} \frac{\alpha_j v_i^T A v_j}{v_i^T A v_i} v_i.$$

In particular, this implies

$$\begin{aligned}\|x_{k+n} - x^*\| &\leq \Big\| x_k - \sum_{i=k}^{k+n-1} \frac{v_i^T \nabla f(x_k)}{v_i^T A v_i} v_i - x^* \Big\| \\ &\quad + \sum_{i=k}^{k+n-1}\sum_{j=k}^{i-1} \Big| \frac{v_j^T \nabla f(x_j)}{v_j^T A v_j} \; \frac{v_i^T A v_j}{v_i^T A v_i} \Big|. \end{aligned} \tag{3.9}$$

Recall that since $v_k, \ldots, v_{k+n-1}$ are conjugate with respect to M_k and $v_i^T A v_i = v_i^T M_k v_i$, it follows that

$$\left\| x_k - \sum_{i=k}^{k+n-1} \frac{v_i^T \nabla f(x_k)}{v_i^T A v_i} v_i - x^* \right\| = \| x_k - M_k^{-1} \nabla f(x_k) - x^* \|. \tag{3.10}$$

Next, recall that since the algorithm is implemented with an exact line search, the objective function is non-increasing at each iteration. Specifically, for all j, $f(x_{j+1}) \leq f(x_j)$. By Equation 3.5, it can be seen that

$$\|x_{j+1} - x^*\|_A^2 = f(x_{j+1}) - f(x^*) \leq f(x_j) - f(x^*) = \|x_j - x^*\|_A^2.$$

This implies that the sequence $\{x_k\}_{k \geq 0}$ is bounded. Additionally, along with Inequality 3.3, this implies that for $j \geq k$ we have,

$$\begin{aligned} |v_j^T \nabla f(x_j)| &\leq \|v_j\| \|\nabla f(x_j)\| \\ &= \|A(x_j - x^*)\| \\ &\leq \sqrt{\lambda_{\max}(A)} \|x_j - x^*\|_A \\ &\leq \sqrt{\lambda_{\max}(A)} \|x_k - x^*\|_A \\ &\leq \lambda_{\max}(A) \|x_k - x^*\|. \end{aligned}$$

Combining the above inequality, Inequality 3.4, Inequality 3.7, Inequality 3.9 and Equation 3.10, we conclude that

$$\|x_{k+n} - x^*\| \leq \|x_k - M_k^{-1} \nabla f(x_k) - x^*\| + \frac{n^2 \lambda_{\max}(A)}{\lambda_{\min}^2(A)} \|x_k - x^*\| \epsilon_k. \tag{3.11}$$

Further, observe that since $\nabla f(x_k) = A x_k + b = A(x_k - x^*)$, it follows that

$$\|x_k - M_k^{-1} \nabla f(x_k) - x^*\| = \|(I - M_k^{-1} A)(x_k - x^*)\|. \tag{3.12}$$

With the above results, we are ready to prove the superlinear convergence of the algorithm. By Theorem 2.3 and Corollary 2.4, it follows that $B_k \to A$ almost surely, implying $\epsilon_k \to 0$ almost surely. Therefore, for all sufficiently large k, the matrix V_k is invertible implying that the matrix M_k is well-defined and that $M_k \to A$ almost surely. From that, Equation 3.12, and the fact that $\{x_k - x^*\}_{k \geq 0}$ is bounded, it follows that

$$\|x_k - M_k^{-1} \nabla f(x_k) - x^*\| \to 0$$

almost surely. Combining this result, the fact that $\epsilon_k \longrightarrow 0$ almost surely and Inequality 3.11, it follows that $\|x_{k+n} - x^*\| \longrightarrow 0$ almost surely, proving that the algorithm converges almost surely.

Finally, consider scaling Inequality 3.11 by $\|x_k - x^*\|$, obtaining

$$\frac{\|x_{k+n} - x^*\|}{\|x_k - x^*\|} \leq \frac{\|x_k - M_k^{-1}\nabla f(x_k) - x^*\|}{\|x_k - x^*\|} + \frac{\frac{n^2\lambda_{\max}(A)}{\lambda_{\min}^2(A)}\|x_k - x^*\|\epsilon_k}{\|x_k - x^*\|}.$$

Since

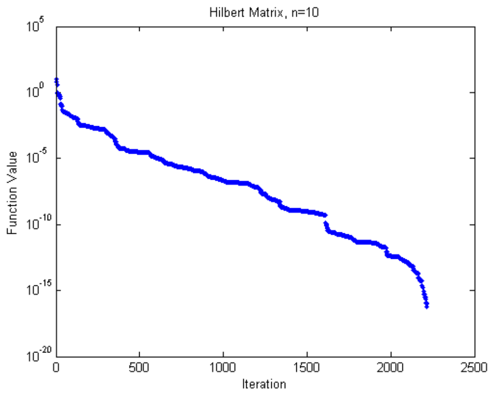
$$\frac{\|x_k - M_k^{-1}\nabla f(x_k) - x^*\|}{\|x_k - x^*\|} = \frac{\|(I - M_k^{-1}A)(x_k - x^*)\|}{\|x_k - x^*\|},$$

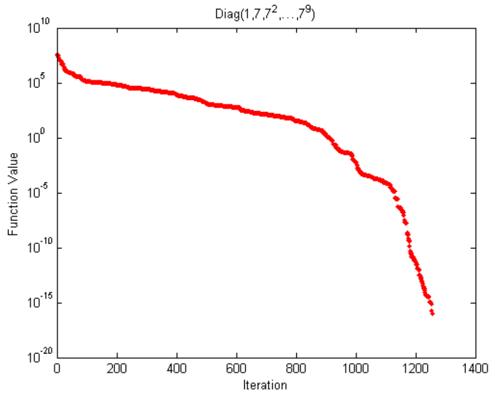
it follows that the first term converges to zero almost surely since $M_k \longrightarrow A$ almost surely. Further, since $\epsilon_k \longrightarrow 0$ almost surely, the second term converges to zero almost surely. These two facts together imply that

$$\frac{\|x_{k+n} - x^*\|}{\|x_k - x^*\|} \longrightarrow 0$$

almost surely: by definition, this means the algorithm is n-step superlinearly convergent almost surely. $\square$

In the following example, we again consider two convex quadratic functions $x^T Ax$ where A is a Hilbert matrix of dimension 10 and $A = \mathrm{Diag}(1, 7, 7^2, \ldots, 7^9)$, respectively. The above algorithm was implemented with an exact line search and exact directional second derivatives.





4 Conclusion

Randomization provides an interesting perspective for a variety of algorithms. Consider the perspective adhered to in this paper in which our cost measure is the number of function evaluations required, assuming line searches are relatively cheap being a one-dimensional optimization problem, and with derivative information only available through (and well-approximated by) finite differencing. It was then shown in Section 1 that random search is comparable to steepest descent. Then, using the Hessian estimation technique introduced in Section 2, Section 3 demonstrated how these techniques can be used to accelerate random search. Finally, we devised a conjugate directions algorithm that incorporates second derivative information without directly requiring gradient information while sharing certain behaviors with more traditional Newton-like methods.

We make no claim that the conceptual techniques described above, in their pure form, are competitive with already-known derivative-based or derivative-free algorithms. We simply intend to illustrate how incorporating randomization provides a novel approach to the design of algorithms, even in very simple optimization schemes, suggesting that it may deserve further consideration. Note that all the algorithms considered in this paper, at each iteration, require only directional derivative or directional second-order information, creating a connection between the realms of derivative-free and derivative-based algorithms when this derivative information is well-approximated by finite differencing.

References

[1] H. Akaike. On a successive transformation of probability distribution and its application to the analysis of the optimum gradient method. *Annals of the Institute of Statistical Mathematics*, 11:1–16, 1959.

[2] P. Billingsley. *Probability and Measure.* John Wiley & Sons, 1986.

[3] K.W. Brodlie. A new direction set method for unconstrained minimization without evaluating derivatives. *Journal of the Institute of Mathematics and Its Applications*, 15:385–396, 1975.

[4] Yu. Ermoliev and R.J.-B. Wets. *Numerical Techniques for Stochastic Optimization.* Springer-Verlag, 1988.

[5] M. Gaviano. Some general results on convergence of random search algorithms in minimization problems. In *Towards Global Optimisation*, pages 149–157, 1975.

[6] J.E. Dennis Jr. and J.J. More. A characterization of superlinear convergence and its application to quasi-Newton methods. *Mathematics of Computation*, 28(126), 1974.

[7] Z.Q. Luo and P. Tseng. On the convergence of the coordinate descent method for convex differentiable minimization. *Journal of Optimization Theory and Applications*, 72:7–35, 1992.

[8] L. Nazareth. Generation of conjugate directions for unconstrained minimization without derivatives. *Mathematics of Computation*, 30(133):115–131, 1976.

[9] M.J.D. Powell. An efficient method for finding the minimum of a function of several variables without calculating derivatives. *The Computer Journal*, 7:155–162, 1964.

[10] M.J.D. Powell. On search directions for minimization algorithms. *Mathematical Programming*, 4:193–201, 1973.

[11] G. Rappl. On linear convergence of a class of random search algorithms. *ZAMM - Journal of Applied Mathematics and Mechanics*, 69(1):37–45, 1989.

[12] F.J. Solis and R. J-B. Wets. Minimization by random search techniques. *Mathematics of Operations Research*, 6(1):19–30, 1981.

[13] J.C. Spall. *Introduction to Stochastic Search and Optimization: Estimation, Simulation and Control.* John Wiley & Sons, New York, 2003.

[14] W.I. Zangwill. Minimizing a function without calculating derivatives. *The Computer Journal*, 10(3):293–296, 1967.