QN OPTIMIZATION WITH HESSIAN SAMPLES

JOY AZZAM *, DANIEL P. HENDERSON †, BENJAMIN W. ONG ‡, AND ALLAN A. STRUTHERS §

Abstract. This article explores how to effectively incorporate curvature information generated using SIMD-parallel forward-mode Algorithmic Differentiation (AD) into unconstrained Quasi-Newton (QN) minimization of a smooth objective function, f. Specifically, forward-mode AD can be used to generate block Hessian samples $Y = \nabla^2 f(x)\, S$ whenever the gradient is evaluated. Block QN algorithms then update approximate inverse Hessians, $H_k \approx \nabla^2 f(x_k)$, with these Hessian samples. Whereas standard line-search based BFGS algorithms carefully filter and correct secant-based approximate curvature information to maintain positive definite approximations, our algorithms directly incorporate Hessian samples to update indefinite inverse Hessian approximations without filtering. The sampled directions supplement the standard QN two-dimensional trust-region sub-problem to generate a moderate dimensional subproblem which can exploit negative curvature. The resulting quadratically-constrained quadratic program is solved accurately with a generalized eigenvalue algorithm and the step advanced using standard trust region step acceptance and radius adjustments. The article aims to avoid serial bottlenecks, exploit accurate positive and negative curvature information, and conduct a preliminary evaluation of selection strategies for S.

Key words. Optimization, Randomized algorithms, Samples, Quasi-Newton

AMS subject classifications. 68W20, 68W2, 65F35, 90C53

1. Literature Discussion.

Schnable [16] first discusses incorporating parallel function evaluations to improve Hessian approximation in optimization algorithms. Together with his colleagues, Byrd, Schnable, and Shultz [6, 5] supplement several standard Quasi Newton optimization schemes with a small number of finite difference second derivative approximations and conclude that the supplemental information improves all the variants considered. Their study includes the underlying QN update, the update order within each step, and a variety of strategies to choose a few supplemental directions. In their conclusions Byrd, Schnable, and Shultz [5] suggest supplementing the extremely simple Symmetric-Rank-One (SR1) QN update with additional second derivative information: the motivation for this suggestion is that Conn, Gould and Toint find in [7] that SR1 is better than Powell-Symmetric-Broyden (PSB), Davidon-Fletcher-Powell (DFP), and Broyden–Fletcher–Goldfarb–Shanno (BFGS). The review article by Schnable [17] summarizes this and other early parallel optimization approaches. More recently, Gau and Goldfarb [8] implemented and tested line-search based algorithms using block BFGS updates on subsets of previous directions, and a family of Quasi-Newton algorithm [9] which avoids a line-search for a restricted class of cost functions. In yet another approach, Berahasa, Jahanib, Richtarik and Takac [2] develop zero-memory Block-BFGS and block-SR1 algorithm using only one (the most recent) set of AD generated Hessian samples. Their BFGS variant is implemented with a Line-Search (requiring a positive definite approximate Hessian) while their SR1 variant is implemented with a Conjugate Gradient based approximate solution of a full dimensional trust region sub problem.

The review article [11] and the articles in the associated special edition of *Parallel Computing* emphasize improving the parallel efficiency of the underlying linear alge-

*Michigan Technological University, Houghton, MI (atazzam@mtu.edu).
†Michigan Technological University, Houghton, MI (dphender@mtu.edu).
‡Michigan Technological University, Houghton, MI (ongbw@mtu.edu), http://mathgeek.us/.
§Michigan Technological University, Houghton, MI (struther@mtu.edu).

bra and introducing parallelism through simultaneous (with different starting points and/or different but possibly related QN updates) line searches. The second-order section of the extensive review article [3] provides a more recent update with a focus on algorithms for very high dimensional problems.

A number of recent developments make it appropriate to revisit the topics in [5] with modern computational tools. The GPU-enabled forward-mode Algorithmic Differentiation (AD), implemented in the open-source computational tool Julia [14] and other software projects, can efficiently replace the finite difference approximations used in [5] and greatly increases the number of simultaneous Hessian samples. The generalized eigenvalue based trust-region sub-problem solver developed by Adachi et al. [1] can replace the line search with an accurate and robust moderate-dimensional trust-region sub-problem (TRSP), implemented in Julia [15].

The goal of this manuscript is to incorporate SIMD-parallel Hessian samples into QN updates to generate provably convergent QN like algorithms. The proposed algorithms avoid serial bottlenecks by using indefinite approximate Hessians. The standard two-dimensional TRSP minimizes a quadratic model over the span of the steepest-descent and Newton directions. This 2D search space is extended (through a specific Hessian re-sampling strategy) to include additional supplementary directions with accurate curvature information on the resulting moderate dimensional sub-space. With standard trust-region controls this gives a provably convergent algorithm with rapid asymptotic convergence to non-degenerate local minimizers.

The article explores two simple indefinite updates (a block variant of SR1 and a block variant of Powell Symmetric Broyden) which can be directly implemented on accurate AD curvature information. In contrast, line-search based block methods (such as a block BFGS or DFP) need to carefully filter and correct approximate curvature information to maintain positive definite Hessian approximations. The article explores how the selection strategy for and number of supplementary directions affects the algorithms based on these two standard indefinite QN updates.

Section 2 introduces essential assumptions and notation. Section 3 discusses the choices made to evaluate curvature information using Algorithmic Differentiation (AD). Section 4 discusses the advantages of indefinite QN updates for trust-region optimization, explains why block BFGS and DFP are not suitable, and presents the simple and highly-parallel block SR1 and PSB updates used. Section 5 describes the trust-region sub-problem underlying the algorithm. Section 6 explains how new supplemental directions are chosen. Section 7 describes the assembled algorithm including: a simple mean curvature estimate used to initialize H; a simple initial trust region radius Δ_0 based on the curvature in the steepest descent direction; standard trust region control; cost evaluation; and pseudo code. Section 8 describes our numerical experiments. Section 9 summarizes the results and future plans.

2. Notation and Assumptions. We assume throughout we are seeking the unconstrained minimum of a C^2 function $f : \mathbb{R}^n \to \mathbb{R}$, with gradient $g(x) = \nabla f(x)$ given by $g : \mathbb{R}^n \to \mathbb{R}^n$, and that we can efficiently sample the Jacobian of g (which is the Hessian of f) $J(x) = \nabla g(x) = \nabla^2 f(x)$ using forward-mode AD. In this context, sample the Hessian means that whenever we compute $g(x) = \nabla f(x)$ we can efficiently and simultaneously compute $Y = J(x)\,S \in \mathbb{R}^{n\times w}$ for a block of w directions, $S \in \mathbb{R}^{n\times w}$. We write $\texttt{orth}(M)$ for an orthogonalization of M (implemented as `Matrix(qr(M).Q)` in Julia), $M_\delta^\dagger$ (implemented as $\texttt{pinv}(M;\,\text{rtol} = \delta)$ in Julia) for the δ thresholded pseudo-inverse of M, and $M \sim \mathcal{N}_{0,1}^{n\times w}$ (implemented as $\texttt{randn}(\text{n,w})$) for an $n \times w$ matrix with elements drawn from the standard normal distribution. We

denote the current search point by x_k with: objective function value $f_k = f(x_k)$; gradient $g_k = g(x_k) = \nabla f(x_k)$; Jacobian $J_k = J(x_k) = \nabla^2 f(x_k)$; and $h_k = h(x_k)$ where $h(x) = J(x)\, g(x) = \nabla^2 f(x) \nabla f(x)$. Symmetric Quasi-Newton (QN) approximations B_k and H_k are updated (using indefinite updates which can incorporate negative curvature information) to satisfy $B_k \approx J_k$ and $H_k \approx J_k^{-1}$. The Frobenius inner product of matrices A and B is denoted by $\langle A, B \rangle_F$ and the Frobenius norm $||A||_F^2 = \langle A, A \rangle_F$ is used throughout.

3. Algorithmic Differentiation and Hessian Samples. In our context, sampling the Hessian means that whenever we compute $g(x) = \nabla f(x)$, we can efficiently and simultaneously compute $Y = \nabla^2 f(x)\, S \in \mathbb{R}^{n \times w}$ for a block of w directions $S \in \mathbb{R}^{n \times w}$. The Julia `ForwardDiff` package [14] modifies the code of g to code $\texttt{gAD}(x, S) : \mathbb{R}^n \times \mathbb{R}^{n \times w} \to \mathbb{R}^n \times \mathbb{R}^{n \times w}$ for the simultaneous combined computation. In practice, `gAD` is embarrassingly SIMD parallel. Provided w does not exceed the available processor resources, evaluating $\texttt{gAD}(x, S)$ takes only $2 - 3$ times as long as evaluating $g(x)$. Since modern GPUs have over 256 cores organized into SIMD warps of 8, 16 or 32 threads, values of $w \leqslant 256$ are feasible on most commodity modern hardware with much larger values feasible on specialized hardware.

The algorithm we will generate centers around generating accurate curvature information at a single point for the steepest descent direction, in addition to other sampled directions. To accomplish this, the curvature information Incorporated in Algorithm 3.1 is generated by two sequential `gAD` calls. A first call to `gAD` computes the gradient and a first Hessian sample $(g, Y_1) \leftarrow \texttt{gAD}(x, S_1)$. The gradient, g, is included in a second direction set S_2 so that the Hessian sample Y_2 computed by $\texttt{gAD}(x, S_2)$ contains $h(x) = J(x)\, g(x) = \nabla^2 f(x)\, \nabla f(x)$. This can be organized in a number of ways. Algorithm 3.1 gives the simple (and almost certainly non-optimal) choices made for a combined gradient and Hessian Sample operation $\texttt{gHS}(x, S)$.

Algorithm 3.1 Gradient and Hessian Sample: $(g, h, Y) \leftarrow \texttt{gHS}(x, S)$

Require: $x \in \mathbb{R}^n$ and $S \in \mathbb{R}^{n \times (2w-1)}$.

1: Compute $(g, Y_1) \leftarrow \texttt{gAD}(x, S[:, 1:w])$ {Input the first w columns of S to `gAD`}
2: Compute $(g, Y_2) \leftarrow \texttt{gAD}(x, [\, S[:, w+1\!:\!\text{end}\,],\, g\,])$ {Input the last $w-1$ columns of S and g to `gAD`}
3: Assemble $Y \leftarrow [\, Y_1,\, Y_2[:, 1:\text{end}-1]\,]$
4: **return** $(g, Y_2[:, \text{end}], Y)$ {By construction, $Y_2[:, \text{end}] = \nabla^2 f(x)\, g$}

4. Quasi-Newton Updates. Several Quasi-Newton updates (with a variety of update details) are tested in [6]. The subsequent article [5] which focuses on the now dominant BFGS algorithm notes that an indefinite SR1 update might work well, as evidenced in an almost contemporaneous article [7]. In this preliminary study we use the indefinite block SR1 and PSB algorithms to incorporate a block of accurate Hessian samples $Y = \nabla^2 f(x_{k+1})\, S$ after successful steps. We explore the effect of including a standard secant curvature estimates $(\nabla f(x_{k+1}) - \nabla f(x_k)) \approx \nabla f^2(x_k)\, (x_{k+1} - x_k)$ before the block update and including the prior step in the block update. The Hessian is not updated on a failed step since the gradient is not evaluated and there is no new curvature information.

Conn, Gould and Toint [7] present evidence that when the underlying Hessian is indefinite, SR1 (with suitable globalization strategies for indefinite Hessian approximations) can outperform updates (like BFGS) designed to maintain positive definite

Hessian approximations. We use a globalization strategy (the moderate dimensional trust region presented in Section 5) which can exploit negative curvature and select QN updates suitable for indefinite approximations.

Block versions of updates which implicitly use the current Hessian (such as BFGS and DFP) require care when the Hessian is indefinite. We explain the issues for the block BFGS update in Algorithm 4.1 [5, 6] which incorporates a block of curvature estimates $\widehat{V} \approx \nabla^2 f(x)\, U$ into an inverse Hessian approximation $H^{-1} \approx \nabla^2 f(x)$. There are two reasons for the initial filtering step in line of Algorithm 4.1. Finite

Algorithm 4.1 block BFGS Update: $H_{\text{BFGS}} \leftarrow \texttt{BFGS}(H, U, \widehat{V}, \delta)$.

Require: SPD $H \in \mathbb{R}^{n\times n}$ and $U, \widehat{V} \in \mathbb{R}^{n\times 2w}$ with $U^\top \widehat{V}$ approximately symmetric
1: Filter and correct $\widehat{V}$ to V consistent with $V = A\,U$ for some SPD A.
2: Compute $T \leftarrow \left(U^\top V\right)^\dagger_\delta$.
3: **return** $U\,T\,U^\top + \left(I - U\,T\,V^\top\right) H \left(I - V\,T\,U^\top\right)$.

difference approximations and/or multiple secant estimates generate approximations $\widehat{V} \approx \nabla^2 f(x)\, U$, however, $\widehat{V}$ needs to be corrected to ensure $U^\top \widehat{V}$ is symmetric. When $\nabla^2 f(x)$ is not positive definite, negative curvature directions in $\widehat{V} \approx \nabla^2 f(x)\, U$ need to be filtered to ensure H_{BFGS} is SPD. These corrections and filters [5, 6, 2] are inherently serial. In contrast, there is no need to correct accurate AD Hessian samples $V = \nabla^2 f(x)\, U = J(x)\, U$ from $\texttt{gHS}(x, S)$ or filter negative curvature directions for trust region based algorithms which can use indefinite Hessian approximations H. Note the BFGS update is a critical point of

$$\underset{A\,V=U, A=A^\top}{\arg\min} \left\langle\, (H-A)\,\mathcal{J}^{-1}\,,\, (H-A)\,\right\rangle_F \tag{4.1}$$

for any invertible $\mathcal{J}$ consistent with the sample in the sense that $U^\top V = U^\top \mathcal{J}\, U$. Running BFGS without filtering produces meaningless updates since if $U^\top V$ is not SPD (4.1) gives a saddle point of an unbounded minimization problem.

We test two indefinite block QN updates with identical trust-region sub-solvers and controls. Algorithm 4.2 specifies block Symmetric Rank 1 (block SR1) which is a direct block generalization [2] of the rank one SR1 update: block SR1 is the algebraically minimal update H_{SR1} which satisfies the block inverse secant condition $H_{\text{SR1}} V = U$. Algorithm 4.3 specifies block Powell-Symmetric-Broyden (block PSB) which is a direct block generalization of the rank two Powell-Symmetric-Broyden (PSB) update [6, 5]: block PSB is the minimal Frobenius norm change satisfying the block inverse secant condition $H_{\text{PSB}} V = U$, i.e.,

$$H_{\text{PSB}} = \underset{AV=U, A=A^\top}{\arg\min} ||H - A||_F = \underset{AV=U, A=A^\top}{\arg\min} \left\langle\, H-A\,,\, H-A\,\right\rangle_F .$$

Algorithms 4.2 and 4.3 do not need to filter accurate curvature information from gHS.

Algorithm 4.2 block SR1 Update: $H_{\text{SR1}} \leftarrow \texttt{SR1}(H, U, V, \delta)$.

Require: $H \in \mathbb{R}^{n\times n}$ with $H = H^\top$; $U, V \in \mathbb{R}^{n\times 2w}$ with $U^\top V = V^\top U$; $\delta > 0$.
1: Compute $T \leftarrow ((U - H\,V)^\top V)^\dagger_\delta$.
2: **return** $H + (U - H\,V)\,T\,(U - H\,V)^\top$.

Algorithm 4.3 block PSB Update: $H_{\text{PSB}} \leftarrow \texttt{PSB}(H, U, V, \delta)$.

Require: $H \in \mathbb{R}^{n\times n}$ with $H = H^\top$; $U, V \in \mathbb{R}^{n\times 2w}$ with $U^\top V = V^\top U$; $\delta > 0$.

1: Compute $T_1 \leftarrow \left(V^\top V\right)^\dagger_\delta$.
2: Compute $T_2 \leftarrow V T_1 (U - HV)^\top$.
3: **return** $H + T_2 + T_2^\top - T_2\, V\, T_1\, V^\top$.

The pseudo-inverse tolerance $\delta = 10^{-12}$ simply restricts excessively large updates.

Byrd and Schnabel [5, 6] considered various additional updates and discovered that including the approximate secant curvature information

$$y_{k+1} = (\nabla f_{k+1} - \nabla f_k) \approx \nabla^2 f(x_{k+1}) p_k = \nabla^2 f(x_{k+1})(x_{k+1} - x_k) \tag{4.2}$$

improved their algorithms. They recommend a preliminary QN update with the approximate curvature information (4.2) before a second update to incorporate the additional curvature information. We perform numerical experiments which replicate this observation and evaluate a possible block replacement.

5. Trust Region Sub-Problem. Trust region algorithms are based on approximate solutions of the n dimensional quadratically constrained quadratic program

$$p_k = \underset{|p|\leqslant\Delta_k}{\arg\min} \frac{1}{2} p^\top H_k^{-1} p + \nabla f_k^\top p \quad \text{where} \quad H_k^{-1} \approx \nabla^2 f(x_k). \tag{5.1}$$

If H_k is full rank (5.1) is equivalent (with $p_k = H_k q_k$) to

$$q_k = \underset{|H_k q|\leqslant\Delta_k}{\arg\min}\, m_k(q) \quad \text{where} \quad m_k(q) = \frac{1}{2} q^\top H_k\, q + (H_k \nabla f_k)^\top q. \tag{5.2}$$

The standard two-dimensional subspace approximation (discussed on p76 of [13]) minimizes (5.1) for $p = a_1 \nabla f_k + a_2 H_k \nabla f_k$.

The sampling procedure $\texttt{gHS}(x_k, S_k)$ generates accurate curvature information $[\, Y_k \,,\, h_k \,] = \nabla^2 f(x_k) [\, S_k \,,\, \nabla f_k \,]$ in the directions specified by the columns of S_k and ∇f_k and the inverse Hessian approximation, H_k, is immediately updated (using either Algorithm 4.1 or Algorithm 4.2) to match with

$$U = \left[\, S_k \,,\, \frac{\nabla f_k}{\|\nabla f_k\|} \,\right] \quad \text{and} \quad V = \left[\, Y_k \,,\, \frac{h_k}{\|\nabla f_k\|} \,\right]$$

The scaling weights all the columns of V equally and maintains a well-conditioned computation when ∇f_k is large or small. Thus,

$$Y_k = H_k^{-1} S_k \quad \text{and} \quad h_k = H_k^{-1} \nabla f_k. \tag{5.3}$$

The standard 2D approximation (p in the span of ∇f_k and $H_k \nabla f_k$) is expanded with the columns of S_k to give the explicit representation $p = M_k\, a$ where

$$M_k = \left[\, \frac{\nabla f_k}{\|\nabla f_k\|} \,,\, \frac{H_k \nabla f_k}{\|\nabla f_k\|} \,,\, S_k \,\right] \in \mathbb{R}^{n\times(2w+1)}, \quad a \in \mathbb{R}^{2w+1}.$$

In terms of $q = H_k^{-1} p$, the equivalent representation (since H_k is exact on the sample (5.3)) is $H_k^{-1} M_k$. We use the orthogonal representation $Q_k = \texttt{orth}\left(H_k^{-1} M_k\right)$ where

$$H_k^{-1} M_k = \left[\, H_k^{-1} \frac{\nabla f_k}{\|\nabla f_k\|} \,,\, \frac{\nabla f_k}{\|\nabla f_k\|} \,,\, H_k^{-1} S_k \,\right] = \left[\, \frac{h_k}{\|\nabla f_k\|} \,,\, \frac{\nabla f_k}{\|\nabla f_k\|} \,,\, Y_k \,\right]. \tag{5.4}$$

Thus, our new trust-region approximation (with Q_k from (5.4) and m_k from (5.2)) is

$$a_k = \underset{a\in\mathbb{R}^{2w+1}:|H_k Q_k\, a|\leqslant\Delta_k}{\arg\min} m_k(Q_k\, a) \tag{5.5}$$

giving the trial step $x_{k+1} = x_k + p_k = x_k + H_k\, Q_k\, a_k$. The Julia `trs` package [12] (developed for [15] based on [1]) computes accurate eigen-value based solutions of

$$a_k = \underset{a^\top C a\,\leqslant\,\Delta_k^2}{\arg\min} \frac{1}{2} a^\top P a + b^\top a.$$

We use the robust small-scale solver (based on a dense generalized eigenvalue decomposition) and compute accurate solutions of (5.5) with

$$a_k = \texttt{trs\_small}(P, b, \Delta_k, C) \tag{5.6}$$

where the arguments are

$$P = Q_k^\top\, H_k\, Q_k, \quad b = Q_k^\top\, H_k\, \nabla f_k, \quad \text{and} \quad C = Q_k^\top\, H_k^2\, Q_k.$$

6. Supplemental Directions. Lastly, we need to address how to select the supplementary directions, $S \in \mathbb{R}^{n\times(2w-1)}$, in Algorithm 3.1. The six supplemental direction variants considered are:

(6.1a) $S_{k+1} = \texttt{orth}\,(M)\,,$ where $M \sim \mathcal{N}_{0,1}^{n\times(2w-1)}$;

(6.1b) $S_{k+1} = \texttt{orth}\,\big(M - S_k(S_k^\top M)\big)\,,$ where $M \sim \mathcal{N}_{0,1}^{n\times(2w-1)}$;

(6.1c) $S_{k+1} = \texttt{orth}\,\big(Y_k - S_k(S_k^\top Y_k)\big)\,;$

(6.1d) $S_{k+1} = \texttt{orth}\,([\ \texttt{orth}\,(M)\ ,\ p_k\ ])\,,$ where $M \sim \mathcal{N}_{0,1}^{n\times(2w-2)}$;

(6.1e) $S_{k+1} = \texttt{orth}\,\big(\big[\ \texttt{orth}\,\big(M - S_k(S_k^\top M)\big)\ ,\ p_k\ \big]\big)\,,$ where $M \sim \mathcal{N}_{0,1}^{n\times(2w-2)}$;

(6.1f) $S_{k+1} = \texttt{orth}\,\big(\big[\ \texttt{orth}\,\big(Y_k[:,\ 1:\text{end}-1] - S_k(S_k^\top Y_k[:,\ 1:\text{end}-1])\big)\ ,\ p_k\ \big]\big)\,.$

The idea behind (6.1b) was to prevent immediate re-sampling (which will happen in the simple randomization (6.1a)) by orthogonalizing against the immediate previous directions. The idea behind (6.1c) was to guide the algorithm to accurately resolve eigen-space associated with the larger Hessian eigenvalues. As noted in section 4, Byrd and Schnabel [5, 6] perform a preliminary secant update to incorporate the approximate secant curvature information along the previous step $p_k = x_{k+1} - x_k$ from (4.2). A simple block alternative is to include in the p_k in S_{k+1} which ensures that the accurate curvature $\nabla^2 f(x_{k+1})\, p_k$ is incorporated in the inverse Hessian. Equations (6.1d)–(6.1f) are simply variants of (6.1a)–(6.1c) which include p_k.

7. Algorithmic Overview, Motivation, and Details. The primary goals when designing the algorithm was to extract maximal benefit from AD generated curvature information while avoiding linear solves in the potentially large ambient dimension n. Secondary goals (which drove many of the details) were a clean flow of information and provably better objective function reduction (at each step) than familiar convergent benchmarks. Algorithm 7.1 has pseudo code for the assembled algorithm. Some comments are in order.

Line 4: QN algorithms are commonly initialized with a multiple of the identity. The mean of the eigenvalues of $S_0^\top \nabla f^2(x_0)\, S_0 = S_0^\top Y_0$ where $Y_0 = \texttt{gHS}(x_0, S_0)$ provides a natural estimate for this multiplier. This initialization is immediately updated with the QN update to give H_0 satisfying $Y_0 = H_0 S_0$.

Lines 11 & 12: We adopt the simple standard strategy and terminology from [13] for updating the radius Δ_k and accepting or rejecting the trial step $x_k + H_k\, Q_k\, a_k$. Model quality is assessed by measuring the ratio of the actual decrease of the objective function, $f_k - f(x_k + H_k\, Q_k\, a_k)$, to the model decrease, $m_k(0) - m_k(Q_k\, a_k)$, viz.,

$$\rho = \frac{f_k - f(x_k + H_k\, Q_k\, a_k)}{m_k(0) - m_k(Q_k\, a_k)}. \tag{7.1}$$

The trust-region radius is updated (our experiments use the large maximum trust region radius $\Delta_{\max} = 100$) as follows

$$\Delta_{k+1} = \begin{cases} 0.25\Delta_k & \text{if } \rho < 0.25 \\ \min\left(2\Delta_k, \Delta_{\max}\right) & \text{if } \rho > 0.75 \text{ and } ||H_k Q_k a_k|| = \Delta_k \\ \Delta_k & \text{otherwise} \end{cases} . \tag{7.2}$$

We reject the step if $\rho \leqslant 0$ (this is the standard [13] trust-region control with rejection parameter $\eta = 0$): we retain all previous values with the updated subscript $k+1$ except the radius Δ_k and recompute (5.6) with the $\Delta_{k+1} = 0.25\Delta_k$ since $\rho < 0.25$. We accept the step if $\rho > 0$. We set $x_{k+1} = x_k + H_k Q_k a_k$ and $f_{k+1} = f(x_{k+1})$ then resample Hessians and update QN approximations.

8. Numerical Experiments. We test our algorithms on the Rosenbrock function

$$f(x) = \sum_{i=1}^{n} \left[a\left(x_{i+1} - x_i^2\right)^2 + (x_i - 1)^2 \right],$$

which is a popular test problem for gradient-based optimization algorithms. The experiments are initialized with $x_0 \in \mathbb{R}^n$ having each component drawn from the uniform distribution on $[-1, 1]$. The global minimum for the Rosenbrock function lies in a narrow valley with many saddle points (in dimension 60 Kok and Sandrock [10] find $53,165$ saddles and predict over 145 million in dimension 100) which makes the minimization challenging for many algorithms. We report results for $n = 100$ (a relatively small dimensional problem that still illustrates the behavior of our algorithms), and $a = 100$, the standard torture test for optimization algorithms. We treat gradient evaluations as the primary expense in each optimization step and evaluate our algorithms by counting the number of `gHS` evaluations. Each such evaluation involves two sequential calls to the underlying AD code `gAD` with each such call evaluating w simultaneous Hessian samples in about $2 - 3$ times an evaluation of g. Given sufficient SIMD processors and neglecting linear algebra, each algorithmic step takes roughly 5 times the evaluation time for a single gradient. Plots use the number of `gHS` evaluations.

Representative results for single runs are presented in Figures 1 to 4. At times the algorithm converges to the secondary local min described in [10]. These runs are discarded. Figure 1 should be compared to Figure 2 to see the comparatively slow convergence of the PSB update. The block PSB update is very conservative in the sense that it gives the smallest change (in the Frobenius norm) consistent with the new Hessian sample which may cause PSB to struggle with the rapidly changing Rosenbrock Hessian. SR1 is the primary focus from here on. As would be expected Figure 2 shows SR1 converging faster for larger sample sizes w. The improvement appears to decrease as w increases.

Algorithm 7.1 Trust Region Quasi-Newton Optimization with Hessian Samples

Require: Convergence tolerance $\epsilon > 0$; Pseudo-Inverse tolerance $\delta > 0$; Initial point $x_0 \in \mathbb{R}^n$; function handle f; Preliminary QN update flag, *pflag*.

Compute $f_0 \leftarrow f(x_0)$; Draw $M \sim \mathcal{N}_{0,1}^{n\times(2w-1)}$; Set $S_0 \leftarrow \texttt{orth}\,(M)$.
Compute $(\nabla f_0, h_0, Y_0) \leftarrow \texttt{gHS}(x_0, S_0)$. {see Algorithm 3.1}
Construct $U_0 = \left[\, S_0 \,,\ \dfrac{\nabla f_0}{\|\nabla f_0\|} \,\right]$ and $V_0 = \left[\, Y_0 \,,\ \dfrac{h_0}{\|\nabla f_0\|} \,\right]$.
Compute initial mean curvature estimate $\alpha = \texttt{mean}(\texttt{eig}(S_0^\top Y_0))$.
Initialize $\Delta_0 = xxx$ using the curvature in the steepest descent direction. {The 1.1 is included to ensure that the minmum }
Initialize $H_0 \leftarrow \texttt{QN}\left(\alpha^{-1} I, U_0, V_0, \delta\right)$. {See Algorithms 4.1 and 4.2}
Set $Q_0 = \texttt{orth}\left(\left[\, \dfrac{h_0}{\|\nabla f_0\|} \,,\ \dfrac{\nabla f_0}{\|\nabla f_0\|} \,,\ Y_0 \,\right]\right)$. {See (5.4)}
Compute $P \leftarrow Q_0^\top H_0 Q_0$; $\quad b \leftarrow Q_0^\top H_0 \nabla f_0$; $\quad C \leftarrow Q_0^\top H_0^2 Q_0$. {See (5.6)}
repeat $\{k = 0, 1, \ldots\}$
  Compute $a_k \leftarrow \texttt{trs\_small}(P, b, \Delta_k, C)$. {from Julia `TRS` package}
  Compute $p_k \leftarrow H_k Q_k a_k$, $\quad f_{k+1} \leftarrow f(x_k + p_k)$ and ρ given by (7.1).
  Update Δ_{k+1} according to (7.2).
  if $\rho \leqslant 0$ **then**
    Set $x_{k+1} \leftarrow x_k$; $\quad f_{k+1} \leftarrow f_k$; $\quad \nabla f_{k+1} \leftarrow \nabla f_k$; $\quad H_{k+1} \leftarrow H_k$. {Reject Step}
  else
    Set $x_{k+1} \leftarrow x_k + p_k$. {Accept Step}
    if *pflag* **then**
      $H_k \leftarrow \texttt{QN}(H_k, p_k, \nabla f_{k+1} - \nabla f_k, \delta)$
    end if
    Pick new supplemental directions, S_{k+1}, using one of (6.1a)–(6.1f) .
    Compute $(\nabla f_{k+1}, h_{k+1}, Y_{k+1}) \leftarrow \texttt{gHS}(x_{k+1}, S_{k+1})$.
    Construct $U_{k+1} = \left[\, S_{k+1} \,,\ \dfrac{\nabla f_{k+1}}{\|\nabla f_{k+1}\|} \,\right]$ and $V_{k+1} = \left[\, Y_{k+1} \,,\ \dfrac{h_{k+1}}{\|\nabla f_{k+1}\|} \,\right]$.
    Update $H_{k+1} \leftarrow \texttt{QN}\,(H_k, U_{k+1}, V_{k+1}, \delta)$.
    Set $Q_{k+1} = \texttt{orth}\left(\left[\, \dfrac{h_{k+1}}{\|\nabla f_{k+1}\|} \,,\ \dfrac{\nabla f_{k+1}}{\|\nabla f_{k+1}\|} \,,\ Y_{k+1} \,\right]\right)$. {See (5.4)}
    Compute (5.6)
    $P \leftarrow Q_{k+1}^\top H_{k+1} Q_{k+1}$; $\quad b \leftarrow Q_{k+1}^\top H_{k+1} \nabla f_{k+1}$; $\quad C \leftarrow Q_{k+1}^\top H_{k+1}^2 Q_{k+1}$.
  end if
until $||\nabla f_{k+1}|| \leqslant \epsilon$ {Convergence}
return x_{k+1}

Figure 3 compares several SR1 variants with and without a preliminary secant update (*pflag* $= \{0, 1\}$ in Algorithm 7.1) and sample directions chosen from {(6.1a) and (6.1d)}. The sample size is held fixed at $w = 4$. The blue curves (sample directions do not include p_k, (6.1a)) are consistent with the observations in [5, 6] that including the approximate secant curvature information in a preliminary QN update is advantageous. Incorporating p_k in our selection of sample directions, (6.1d) is highly beneficial (red curves), and eliminates the need for the preliminary QN update, which is a serial bottleneck.

Lastly, Figure 4 shows the effect of varying the sample selection strategy between (6.1d)–(6.1f) The simple purely randomized supplemental directions from (6.1d) gave

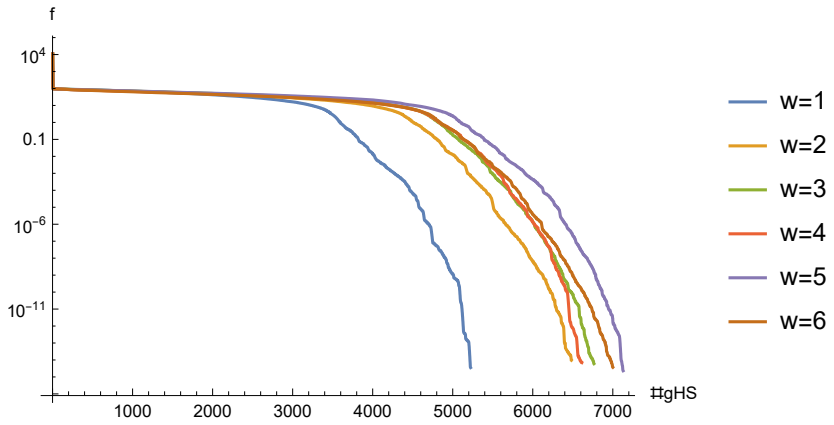

FIG. 1. *Block PSB with preliminary secant update (i.e., with pflag=1 in Algorithm 7.1). Supplemental directions chosen using (6.1d). Surprisingly, as we increase the sample size, the performance of the algorithm degrades.*

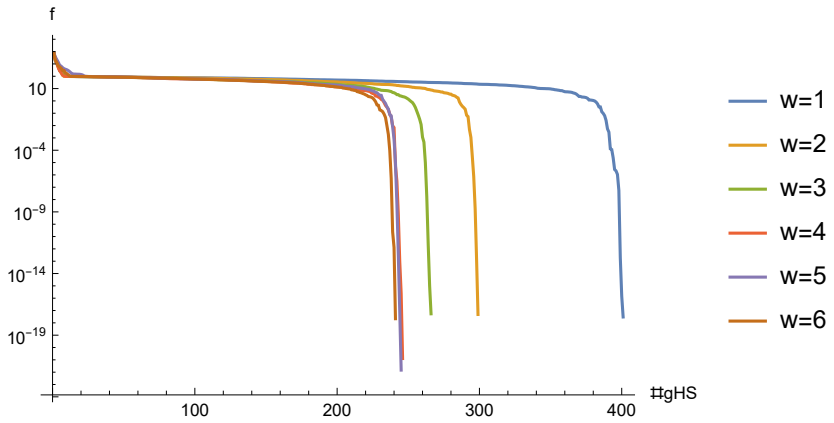

FIG. 2. *Block SR1 with preliminary secant update (i.e., with pflag=1 in Algorithm 7.1). Supplemental directions chosen using (6.1d). Comparing Figure 2 and Figure 1, block SR1 converges much more rapidly than block PSB. Also, the convergence of block SR1 is superior for larger sample sizes w, though the improvement appears to decrease as w increases.*

good results. The intuition behind (6.1e) was to prevent immediate re-sampling by orthogonalizing against the immediate previous directions. It did not lead to significant improvement. The intuition behind (6.1f) was to guide the algorithm to accurately resolve eigen-space associated with the larger Hessian eigenvalues. This variant does appear to resolve these eigenspaces but unfortunately it does not improve the performance of the algorithm.

9. Conclusions and Future Work. The goal was a straightforward algorithm that would focus on potential benefits AD generated Hessian samples in optimzation algorithms. The algorithms presented are intended to evaluate potential benefits of incorporating block Hessian samples in various ways into a fairly standard optimization framework. Practical implementations would require limited memory updates to reduce storage requirements. Carefully eliminating some sampled directions from the trust region sub problem and/or exploiting the structure of a limited memory update

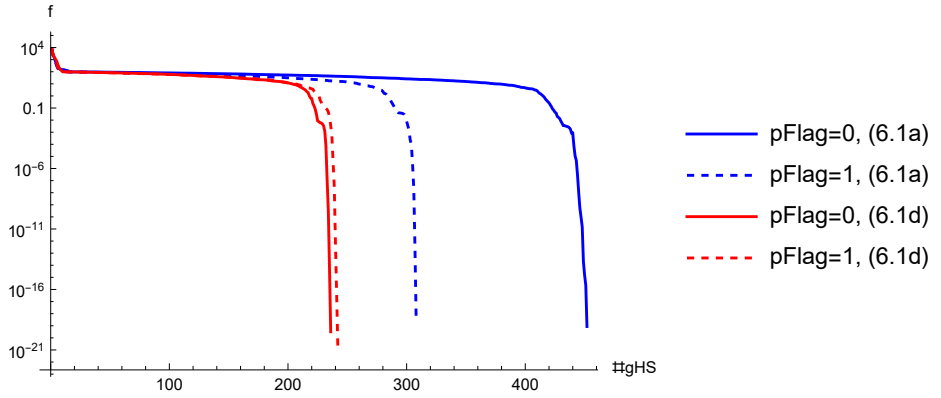

FIG. 3. *Convergence of SR1 variants with pflag=*$\{0,1\}$*, and sample directions chosen using* $\{$(6.1a), (6.1d)$\}$*. Sample size is held fixed at* $w = 4$*. The blue curves (sample directions do not include* p_k*,* (6.1a)*) are consistent with the observations in [5, 6] that including the approximate secant curvature information in a preliminary QN update is advantageous. Our framework allows us to incorporate Hessian information in the direction of* p_k*, i.e.,* (6.1d)*. The red curves show that using* (6.1d) *to select sample directions is highly beneficial, and eliminates the need to include approximate secant curvature information in the Hessian, which is a serial bottleneck.*

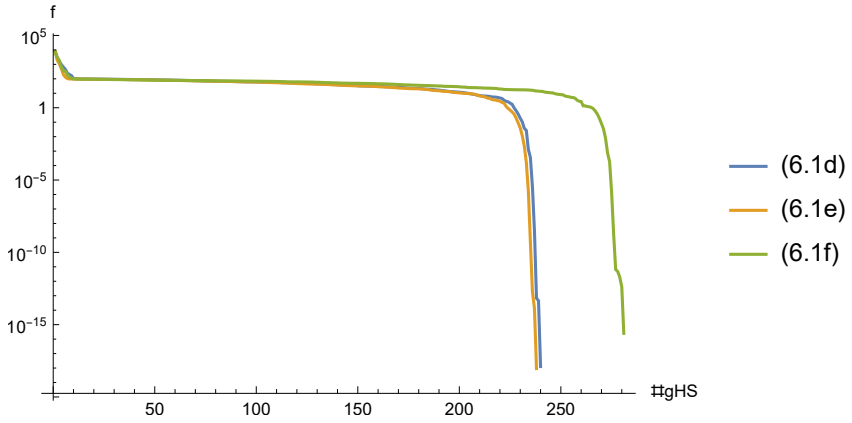

FIG. 4. *SR1 with with* $w = 4$*, samples from* (6.1d)–(6.1f)*, and no preliminary secant update (i.e., with pflag=0 in Algorithm* 7.1*). The simple purely randomized supplemental directions from* (6.1d) *gave good results.*

(as shown for LSR1 by Brust et al. [4]) would reduce the computational intensity of the trust-region sub-problem solver. We do not address these issues in this article and restrict attention to significantly fewer than the 64 Hessian samples which are feasible on common GPU hardware.

10. Distribution of Responsibilities. The article implements matrix approximation ideas from Dr. Azzam's thesis in optimization. Dr. Struthers designed the algorithm and drafted the article with significant input from Drs. Ong and Azzam. Mr. Henderson created the Julia test problems and code, and generated numerical results. All authors made significant editorial contributions.

REFERENCES

[1] Satoru Adachi, Satoru Iwata, Yuji Nakatsukasa, and Akiko Takeda, *Solving the trust-region subproblem by a generalized eigenvalue problem*, SIAM Journal on Optimization, 27 (2017), pp. 269–291.
[2] Albert S. Berahas, Majid Jahani, Peter Richtárik, and Martin Takáč, *Quasi-newton methods for machine learning: Forget the past, just sample*, 2021.
[3] Léon Bottou, Frank E. Curtis, and Jorge Nocedal, *Optimization methods for large-scale machine learning*, SIAM Review, 60 (2018), pp. 223–311.
[4] Johannes Brust, Jennifer B. Erway, and Roummel F. Marcia, *On solving l-sr1 trust-region subproblems*, Comput. Optim. Appl., 66 (2017), p. 245–266.
[5] R.H. Byrd, R.B. Schnabel, and G.A. Shultz, *Parallel quasi-newton methods for unconstrained optimization*, Mathematical Programming, 42 (1988), pp. 273–306. cited By 46.
[6] R. H. Byrd, R. B. Schnabel, and G. A. Shultz, *Using parallel function evaluations to improve hessian approximation for unconstrained optimization*, Ann. Oper. Res., 14 (1988), p. 167–193.
[7] Andrew Conn, Nicholas Gould, and Philippe Toint, *Testing a class of methods for solving minimization problems with simple bounds on the variables*, Mathematics of Computation, 50 (1988).
[8] Wenbo Gao and Donald Goldfarb, *Block bfgs methods*, SIAM Journal on Optimization, 28 (2018), pp. 1205–1231.
[9] Wenbo Gao and Donald Goldfarb, *Quasi-newton methods: superlinear convergence without line searches for self-concordant functions*, Optimization Methods and Software, 34 (2019), pp. 194–217.
[10] Schalk Kok and Carl Sandrock, *Locating and Characterizing the Stationary Points of the Extended Rosenbrock Function*, Evolutionary Computation, 17 (2009), pp. 437–453.
[11] A. Migdalas, G. Toraldo, and V. Kumar, *Nonlinear optimization and parallel computing*, Parallel Computing, 29 (2003), pp. 375–391. Parallel computing in numerical optimization.
[12] Rontsis N., Goulart P. J., and Y. Nakatsukasa, *TRS.jl*, 2021.
[13] Jorge Nocedal and Stephen J. Wright, *Numerical Optimization*, Springer, New York, NY, USA, second ed., 2006.
[14] J. Revels, M. Lubin, and T. Papamarkou, *Forward-mode automatic differentiation in Julia*, arXiv:1607.07892 [cs.MS], (2016).
[15] Nikitas Rontsis, Paul Goulart, and Yuji Nakatsukasa, *An active-set algorithm for norm constrained quadratic problems*, Mathematical Programming, (2021), pp. 1–37.
[16] R.B. Schnabel, *Concurrent function evaluations in local and global optimization*, Computer Methods in Applied Mechanics and Engineering, 64 (1987), pp. 537–552. cited By 17.
[17] Robert B. Schnabel, *A view of the limitations, opportunities, and challenges in parallel nonlinear optimization*, Parallel Computing, 21 (1995), pp. 875–905.