SIAM J. MATRIX ANAL. APPL.
Vol. 45, No. 2, pp. 905–929

ARE SKETCH-AND-PRECONDITION LEAST SQUARES SOLVERS NUMERICALLY STABLE?*

MAIKE MEIER†, YUJI NAKATSUKASA†, ALEX TOWNSEND‡, AND MARCUS WEBB§

In memory of Prof. N.J. Higham FRS, 1961–2024

Abstract. Sketch-and-precondition techniques are efficient and popular for solving large least squares (LS) problems of the form $Ax = b$ with $A \in \mathbb{R}^{m\times n}$ and $m \gg n$. This is where A is "sketched" to a smaller matrix SA with $S \in \mathbb{R}^{\lceil cn\rceil\times m}$ for some constant $c > 1$ before an iterative LS solver computes the solution to $Ax = b$ with a right preconditioner P, where P is constructed from SA. Prominent sketch-and-precondition LS solvers are Blendenpik and LSRN. We show that the sketch-and-precondition technique in its most commonly used form is not numerically stable for ill-conditioned LS problems. For provable and practical backward stability and optimal residuals, we suggest using an unpreconditioned iterative LS solver on $(AP)z = b$ with $x = Pz$. Provided the condition number of A is smaller than the reciprocal of the unit roundoff, we show that this modification ensures that the computed solution has a backward error comparable to the iterative LS solver applied to a well-conditioned matrix. Using smoothed analysis, we model floating-point rounding errors to argue that our modification is expected to compute a backward stable solution even for arbitrarily ill-conditioned LS problems. Additionally, we provide experimental evidence that using the sketch-and-solve solution as a starting vector in sketch-and-precondition algorithms (as suggested by Rokhlin and Tygert in 2008) should be highly preferred over the zero vector. The initialization often results in much more accurate solutions—albeit not always backward stable ones.

Key words. least squares, numerical stability, sketching, preconditioner

MSC codes. 65F10, 65F20

DOI. 10.1137/23M1551973

1. Introduction.

Randomized numerical linear algebra is a growing subfield of matrix computations that has produced major advances in low-rank approximation [22], iterative methods [32], and projections [5]. Sketch-and-precondition techniques are a class of randomized algorithms for solving overdetermined least squares (LS) problems of the form

$$\min_{x\in\mathbb{R}^n} \|Ax - b\|_2, \qquad A \in \mathbb{R}^{m\times n}, \quad b \in \mathbb{R}^{m\times 1}, \tag{1.1}$$

where $m > n$. One first sketches A to a smaller matrix SA with a random sketch matrix $S \in \mathbb{R}^{\lceil cn\rceil\times m}$ for some constant $c > 1$, then a right preconditioner, P, is constructed from SA. Finally, one solves $Ax = b$ using an iterative LS solver with the right preconditioner P. There are many details of sketch-and-precondition based on how to sketch and construct P as well as which iterative LS solver to employ. One of the most prominent sketch-and-precondition techniques is known as Blendenpik [1] (see Algorithm 1.1). In exact arithmetic, Blendenpik has a complexity of $\mathcal{O}(mn \log m)$

*Received by the editors February 14, 2023; accepted for publication (in revised form) January 18, 2024; published electronically April 24, 2024.
https://doi.org/10.1137/23M1551973
†Mathematical Institute, University of Oxford, Oxford, OX2 6GG England (meier@maths.ox.ac.uk, nakatsukasa@maths.ox.ac.uk).
‡Department of Mathematics, Cornell University, Ithaca, NY 14853 USA (townsend@cornell.edu).
§Department of Mathematics, University of Manchester, Manchester, England (marcus.webb@manchester.ac.uk).

Algorithm 1.1. A sketch-and-precondition LS solver for (1.1) known as Blendenpik. Here, HHQR refers to the Householder QR algorithm.

1: Draw a random sketching matrix $S \in \mathbb{R}^{s\times m}$, where $m \gg s > n$
2: Compute $B = SA$
3: Compute the triangular factor R of a QR factorization of B using HHQR
4: Solve $Ax = b$ with LSQR and right preconditioner $P = R^{-1}$

operations, which is better than the $\mathcal{O}(mn^2)$ QR-based direct solver. Consequently, for large LS problems, Blendenpik can be substantially faster than the LS solver implemented in LAPACK [1]. However, are sketch-and-precondition techniques—such as Blendenpik—numerically stable?

Surprisingly, we find that sketch-and-precondition techniques such as Blendenpik [1] and LSRN [24] are numerically unstable in their standard form (see Figure 1.1). For moderately ill-conditioned problems ($1 \ll \kappa_2(A) \ll u^{-1}$, where $\kappa_2(A) = \sigma_{\max}(A)/\sigma_{\min}(A)$ is the condition number of A and u is the unit roundoff), sketch-and-precondition iterations stagnate in terms of residual and backward error, potentially before optimal levels are reached. The main focus of this work is deriving a provable method to resolve these numerical instabilities. We suggest a modification to the sketch-and-precondition framework to obtain a new algorithm, which we coin sketch-and-apply. We show using classical stability analysis and experimentally that sketch-and-apply attains backward stable solutions under modest conditions. However, sketch-and-apply requires $\mathcal{O}(mn^2)$ operations, the same complexity as the classical LS solver based on QR. This work thus highlights the significant open problem: is there a fast (randomized) LS solver with guaranteed backward stability?

Throughout the paper, we assume that $A \in \mathbb{R}^{m\times n}$ and $SA \in \mathbb{R}^{s\times n}$ (for some s such that $m > s > n$) are of full rank so that (1.1) has a unique solution.

1.1. Numerical stability of LS solvers. The stability and accuracy of solutions to and (classical) solvers for the LS problem (1.1) are well studied; see, e.g., [18, Ch. 20], [2, Chap. 1], and [3, 11, 30]. One can consider various metrics for the quality of a computed solution x to (1.1) with exact solution x^*, among which are the residual $\|Ax - b\|_2 = \|r\|_2$, the normal residual $\|A^T r\|_2$, the forward error $\|x - x^*\|_2$, and the (normwise) backward error. The backward error is the smallest perturbation to A and b such that x exactly solves the perturbed system. That is, we define the backward error as [34]

$$\eta_F(x) := \min\{\|[\Delta A, \Delta b]\|_F : \|(A + \Delta A)x - (b + \Delta b)\|_2 = \min\}. \tag{1.2}$$

It can be computed using [34] (see also [18, Thm. 20.5]),

$$\eta_F(x) = \min\left\{\phi,\ \sigma_{\min}\left(\begin{bmatrix} A & \phi(I_m - rr^\dagger)\end{bmatrix}\right)\right\}, \quad \phi = \frac{\|r\|_2}{\sqrt{1 + \|x\|_2^2}}, \quad r = b - Ax,$$

where $\sigma_{\min}(B)$ indicates the smallest singular value of B, and $r^\dagger = r^T/\|r\|^2$ is the pseudoinverse of r. A solution is backward stable if its backward error is a modest multiple of machine precision. An algorithm is backward stable if it always computes a backward stable solution. It is an important quality measure for a solution; if the backward error is smaller than the errors in the data A and b (e.g., measurement errors), then the solution is as accurate as we can possibly achieve [16].

We furthermore often report on the residual, an important metric for practitioners. The optimal residual, $\min \|Ax-b\| = \|Ax^*-b\|$, depends on the specific problem, and may be large (inconsistent system) or small (nearly consistent system). We aim to achieve solutions with a residual close to optimal and a backward error close to the unit roundoff. Throughout this work, we show that sketch-and-precondition algorithms cannot guarantee this.

1.2. Sketch-and-precondition algorithms. Sketch-and-precondition algorithms use sketching to construct a preconditioner to be used in an iterative LS method. Given an embedding matrix $S \in \mathbb{R}^{s\times m}$, $n < s \ll m$, and the resulting sketch SA, one would usually compute a QR decomposition of the sketch and use the inverse of the R-factor as a right preconditioner in LSQR. This is the basis of the Blendenpik algorithm [1]. Another popular sketch-and-precondition technique is LSRN [24], which uses the singular value decomposition (SVD) of SA instead of the QR decomposition. The computational cost of Blendenpik is $\mathcal{O}(mn\log n)$ to compute the sketch, $\mathcal{O}(n^3)$ for the QR decomposition, and $\mathcal{O}(mn)$ cost per iteration in the iterative solver.

The most commonly used iterative solver for (1.1) is LSQR [27], which is competitive when A is large, sparse, and well-conditioned. This is also the recommended solver in Blendenpik and LSRN. The number of iterations required to reach a desired accuracy typically depends on the condition number of A [2, Chap. 7.4]. As a result, a preconditioner is necessary when A is ill-conditioned to ensure a reasonable speed of convergence. With careful sketching, the condition number of AP with $P = R^{-1}$ from Algorithm 1.1 ($SA = QR$) is small with high probability in exact arithmetic [28, Lem. 1] (see Lemma 2.1). In finite precision arithmetic, the condition number of the computed AP remains modest in size, provided that $\kappa_2(A)u \ll 1$ (see Theorem 3.4). However, as we will see, although AP is well-conditioned, the application of P can cause numerical errors. The crux of the numerical instability in sketch-and-precondition is the repeated application of $P = R^{-1}$, which is about as ill-conditioned as A. Indeed, it is recommended in the Blendenpik paper that LSQR is avoided when $\kappa_2(R) > 1/(5u)$ [1]. However, we find that for moderately ill-conditioned systems (such as $\kappa_2(A) \approx \sqrt{u^{-1}}$), numerical errors affect the accuracy of the solutions commensurately (see subsection 1.5).

Sketch-and-precondition in its standard form (see Algorithm 1.1) has $x_0 = 0$ as the initial guess in LSQR. However, as suggested in Rokhlin and Tygert's paper [28], a more natural guess is readily available: the solution to the sketched LS problem, i.e.,

$$x_0 = \operatorname*{arg\,min}_{x\in\mathbb{R}^n} \|SAx - Sb\|_2. \tag{1.3}$$

This can be computed directly with the QR decomposition of SA; the resulting algorithm is displayed in pseudocode in Algorithm 1.2. Although originally proposed in [28] as part of the sketch-and-precondition framework, most implementations (e.g., [1, 24]) do not mention this choice of initial guess as part of their algorithms.[1]

The sketch-and-solve solution (1.3) typically attains a residual within a small multiple of the optimal residual (see subsection 2.1.1). Although the attainable accuracy of sketch-and-precondition algorithms with a random or trivial initial guess may stagnate before a desired accuracy when dealing with ill-conditioned problems (see Figure 1.1(b)), our experiments show Algorithm 1.2 attains optimal residuals in most cases,

[1] In the C implementation of Blendenpik [1], SAS initialization is available as an option, but the user needs to append 'improve_start_point' to the parameters.

Algorithm 1.2. A sketch-and-precondition LS solver for (1.1) with a sketch-and-solve solution as initial guess. Here, HHQR refers to the Householder QR algorithm.

1: Draw a random sketching matrix $S \in \mathbb{R}^{s\times m}$, where $m \gg s > n$
2: Compute $B = SA$ and $c = Sb$
3: Compute both Q and R of the QR factorization of B using HHQR
4: Compute initial guess $x_0 = R^{-1}Q^T c$
5: Solve $Ax = b$ with LSQR and right preconditioner $P = R^{-1}$ and initial guess x_0

and is significantly better than the standard, trivial initial guess $x_0 = 0$. Furthermore, as the QR decomposition (or SVD) of the sketch SA is necessary for any sketch-and-precondition solver, the sketch-and-solve initial guess is obtained practically for free. We urge practitioners to adopt this as standard practice for these types of algorithms. It should be noted, however, that there are instances where the solution found by sketch-and-precondition with sketch-and-solve initialization does not attain backward stable solutions (see Figure 1.1(a)). To further resolve the numerical instabilities, we introduce the *sketch-and-apply* algorithm.

1.3. Sketch-and-apply. The sketch-and-apply algorithm is a modification of the sketch-and-precondition algorithm for which we can ensure that the computed residual is close to optimal and the backward error is approximately machine precision regardless of the initial guess.

The modification of sketch-and-precondition is simple. Instead of using $P = R^{-1}$ as a preconditioner, we explicitly apply P by computing AP and then employ an unpreconditioned iterative LS solver on $(AP)z = b$ with $x = Pz$. We therefore call this a sketch-and-apply technique. Of course, in exact arithmetic sketch-and-precondition and sketch-and-apply compute the same solution; however, for ill-conditioned LS problems in floating-point arithmetic, we find a significant difference. To our knowledge, this is the first algorithm for LS problems based on randomized sketching that is demonstrated to be backward stable (with a mild assumption to be made precise in Theorem 3.5).

By computing AP explicitly, we remove all ill-conditioning from the iterative solver and instead use an unpreconditioned solver on a well-conditioned system. This results in accurate and backward stable solutions. In particular, we prove that if $\kappa_2(A) \ll u^{-1}$, our sketch-and-apply technique computes a backward stable solution provided that LSQR on a well-conditioned matrix computes a backward stable solution (see Theorem 3.5).

Unfortunately, while sketch-and-precondition techniques cost $\mathcal{O}(mn \log m)$ operations, sketch-and-apply costs $\mathcal{O}(mn^2)$ operations as AP must be computed. Therefore, sketch-and-apply techniques have the same computational complexity as the classical QR-based LS solver. We note, however, that LSRN, another popular sketch-and-precondition algorithm, also uses $\mathcal{O}(mn^2)$ operations but is still competitive as the expensive computation is in matrix multiplication, which is highly parallelizable. The same applies to sketch-and-apply.

When $\kappa_2(A) \gtrsim u^{-1}$, our analysis does not guarantee that Algorithm 1.3 leads to an accurate solution. However, in practice, we often get accurate final LS residuals. To explain why this is the case, we model floating-point rounding errors using smoothed analysis. That is, we consider "smoothing" A to $A + \sigma G/\sqrt{m}$, where the entries of G are independent and identically distributed (i.i.d.) standard Gaussian random variables and σ is a scaling factor that we select as $\sigma = 10\|A\|_2 u$. The idea is that $A + \sigma G/\sqrt{m}$ is significantly better conditioned than A itself, assuming that A is extremely

Algorithm 1.3. A sketch-and-apply LS solver for (1.1). Here, HHQR refers to the Householder QR algorithm. One can include a sketch-and-solve initial guess by computing the QR decomposition $QR = SA$ and $z_0 = Q^T Sb$, and setting z_0 as an initial guess in step 5.

1: Draw a random sketching matrix $S \in \mathbb{R}^{s \times m}$, where $m \gg s > n$
2: Compute $B = SA$
3: Compute the triangular factor R of a QR factorization of B using HHQR
4: Compute $Y = AR^{-1}$ with forward substitution
5: Solve $Yz = b$ with LSQR and no preconditioner
6: Compute $x = R^{-1}z$ with back substitution

Algorithm 1.4. A smoothed sketch-and-apply LS solver for (1.1) when $\kappa_2(A) \gtrsim u^{-1}$.

1: Draw a random standard Gaussian matrix $G \in \mathbb{R}^{m \times n}$ with i.i.d. entries
2: Compute $\tilde{A} = A + \sigma G/\sqrt{m}$ for $\sigma = 10\|A\|_2 u$
3: Perform Algorithm 1.3 on $\tilde{A}x = b$

ill-conditioned. In fact, for sufficiently small σ, one can show that $\kappa_2(A + \sigma G/\sqrt{m}) \lesssim 1/\sigma$ with high probability [6] (see Corollary 2.2). The additive perturbation of $\sigma G/\sqrt{m}$ to A ensures that sketch-and-apply techniques can also compute solutions with good backward error, even when A is extremely ill-conditioned (see Algorithm 1.4). For extremely ill-conditioned LS problems, one could explicitly add an additive random perturbation or hope that floating-point rounding errors deliver the same effect, as it often does in practice.

1.4. Sketch-and-solve initialization versus sketch-and-apply. Figure 1.1(b), as well as various other figures throughout this work, shows the enormous practical significance of using the sketch-and-solve solution as an initial guess. In the version of this paper first submitted to the journal, we discussed the instability of sketch-and-precondition and presented sketch-and-apply as the fix. A careful reviewer suggested that we initialize sketch-and-precondition with the sketch-and-solve solution. The improvement in stability and speed of convergence of sketch-and-precondition when combined with sketch-and-solve initialization (Algorithm 1.2) should be one of the main takeaways of this work from a practical viewpoint. However, it should be noted that sketch-and-precondition with sketch-and-solve initialization is not completely freed from the numerical instabilities present in standard sketch-and-precondition (see Figure 1.1(a)). As a result, sketch-and-apply has significance when it is important to retrieve backward stable solutions.

The remainder of this work focuses on analyzing the sketch-and-apply algorithm. As far as the authors are aware, this is one of the first rigorous stability analyses for a randomized algorithm. As a result, it is also one of the first randomized algorithms proven to be backward stable.

1.5. The numerical instabilities of sketch-and-precondition. We now demonstrate the numerical instabilities that occur with sketch-and-precondition techniques.[2] We construct LS problems at random by setting $A = U\Sigma V^T$, where $U \in \mathbb{R}^{m \times n}$ and $V \in \mathbb{R}^{n \times n}$ are random orthogonal matrices from the Haar distribution

[2]Experiments are performed in MATLAB 2022b using 64-bit arithmetic on a single core of a MacBook Pro equipped with a 2.3 GHz Dual-Core Intel Core i5 processor and 16 GB 2133 MHz LPDDR3 of system memory.

and $\Sigma = \mathrm{diag}(\sigma_1, \ldots, \sigma_n)$, where $\sigma_1, \ldots, \sigma_n$ are logarithmically spaced between 1 and $\sigma_n = 10^{-10}$. We set $m = 10000$ and $n = 100$. The right-hand side of the LS problem is generated as $b = Ax^* + e$, where x^* is a random vector with i.i.d. standard Gaussian entries and e represents a noise vector. The noise is chosen to be orthogonal to the column space of A so that x^* is the exact solution to $Ax = b$. For this reason, the computed residual is bounded from below by $\|e\|_2 = \|Ax^* - b\|_2$. We consider two values of $\|e\|_2$: (a) $\|e\|_2 = 10^{-2}$ (see Figure 1.1(a)) and (b) $\|e\|_2 = 10^{-12}$ (see Figure 1.1(b)).

To solve the LS problems, we use (1) standard sketch-and-precondition Blendenpik (SAP; see Algorithm 1.1), (2) SAP Blendenpik with a sketch-and-solve initial guess (SAP-SAS; see Algorithm 1.2), (3) standard sketch-and-apply Blendenpik (SAA; see Algorithm 1.3), and (4) SAA Blendenpik with a sketch-and-solve initial guess (SAA-SAS; see the caption of Algorithm 1.3). The only difference between the SAP and SAA algorithms is how they employ the preconditioner; the actual preconditioner is identical. We use a sketching matrix $S \in \mathbb{R}^{4n \times m}$, known as a subsampled randomized cosine transform (see subsection 2.2.2). The tolerance is set to 10^{-14} and the maximum number of iterations to 50.

Figure 1.1 compares the performance of the algorithms in terms of the relative residual $\|Ax - b\|_2/\|b\|_2$ (left column), the normwise backward error $\eta_F(x)$ (middle column; see (1.2)), and the relative normal residual $\|A^T(Ax-b)\|_2/(\|A\|_F\|Ax-b\|_2)$.

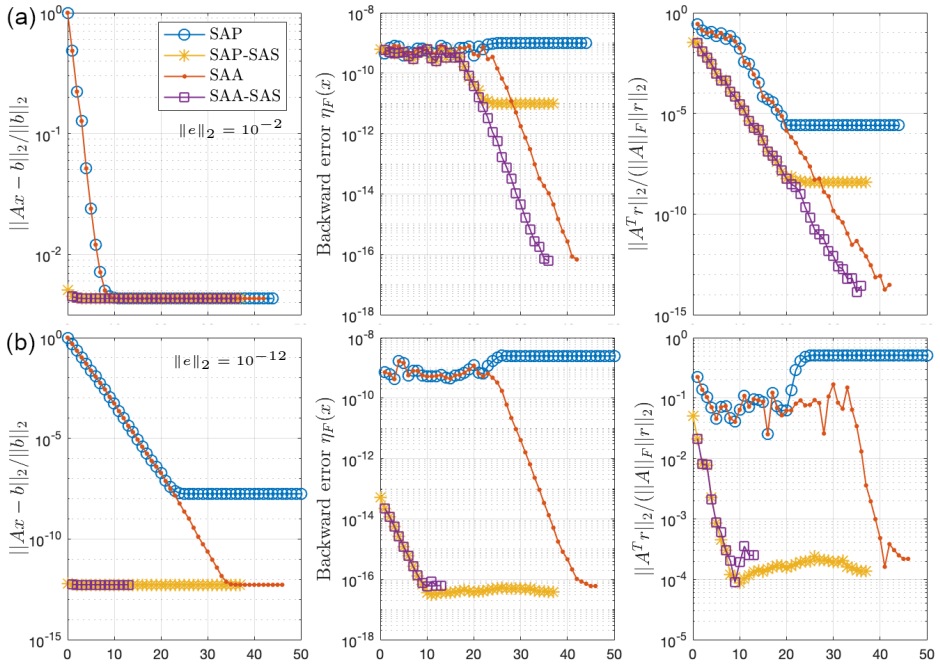

FIG. 1.1. *SAP Blendenpik and SAA Blendenpik, with and without SAS initialization, applied to two noisy and randomly generated LS problems. The noise level is* $\|Ax^* - b\|_2 = \|e\|_2 = 10^{-2}$ *in the top row* (a) *and* $\|Ax^* - b\|_2 = \|e\|_2 = 10^{-12}$ *in the bottom row* (b). *The matrix* A *is* 10000×100 *with condition number* $\kappa_2(A) = 10^{10}$. *We display the residual, the backward error, and the relative normal residual for each LSQR iteration. Note: color appears only in the online article.*

Consider first Figure 1.1(a): an inconsistent, moderately ill-conditioned problem. We see that all algorithms attain the optimal residual. However, considering the backward error, both SAP varieties fail to converge to a backward stable solution. The same behavior can be observed for the relative normal residual. In general we find that for inconsistent problems, i.e., problems with large optimal residual, SAP with or without initialization converges to a solution with a good residual. However, for moderately ill-conditioned problems, these solutions are often not backward stable.

Figure 1.1(b) shows the more obvious instabilities of SAP without initialization: the maximal attainable residual is not optimal. Consider the final residuals in Figure 1.1(b). The SAP solution stagnates around 10^{-8}, whereas all other algorithms attain 10^{-12}. This instability is also visible in the backward error and relative residual; we see SAP without initialization is not backward stable.

Perhaps most notable about Figure 1.1(b) is the success of SAS initialization. The initial guess has an accuracy of the same order as the optimal solution in terms of residual (reflecting standard theory [22]), and the following iterates usually quickly converge to the optimal residual. It appears that starting close to a good guess resolves much of the numerical instabilities of SAP Blendenpik. The initialization results in a backward stable solution. However, it must be noted that some floating-point errors persist, as the backward errors for the inconsistent problems are not optimal (see Figure 1.1(a)).

SAA Blendenpik, with or without initialization, attains accurate solutions with a backward error of order u in all cases. This supports our theoretical findings that SAA is a backward stable algorithm (see section 3). Again, LSQR converges faster when initialization is used, and we always recommend doing this.

In all of the experiments, the tolerance tol in LSQR is set to be very small to allow us to investigate the maximal attainable accuracy. This checks both the relative normal residual $\|A^T r\|_2/(\|A\|_F\|r\|_2) \leq \text{tol}$ and the residual $\|Ax-b\|_2/\|b\|_2 \leq \text{tol}$. For highly inconsistent problems, our small choice of tol can result in many iterations before LSQR is stopped. It should be noted however, that the initialized algorithms reach the levels they will stagnate on much sooner. The LSQR stopping criteria and tolerance are thus a delicate aspect, and significant research has been devoted to the subject [8, 17, 29].

The numerical instabilities in standard SAP Blendenpik could be due to multiple sources. To demonstrate that it is the way that the preconditioner is employed, and not how it is formed, we try one more randomly generated LS problem with $A \in \mathbb{R}^{20000\times 100}$, $\kappa_2(A) = 10^{12}$, and a noise level of $\|e\|_2 = 10^{-14}$. We solve the LS problem with the previously mentioned randomized algorithms, as well as unpreconditioned LSQR and preconditioned LSQR, where the preconditioner is obtained by computing the QR factorization of A. This final approach is not practical because it requires the QR factorization of A, but is done to demonstrate that the quality of the preconditioners is not at fault. We find in Figure 1.2 that the SAS initialization approach computes accurate backward-stable solutions when combined with either SAP or SAA. For the various algorithms employing $x_0 = 0$ as an initial guess, SAA Blendenpik is the only approach that computes a solution with a residual close to optimal and a backward error close to machine precision. We also note that the qualitative behavior shown in Figures 1.1 and 1.2 is unaffected by different types of sketch matrices or sketch dimensions. These choices influence $\kappa_2(AP)$ and $\kappa_2(P)$ slightly, which can somewhat affect the convergence behavior in terms of speed and maximal

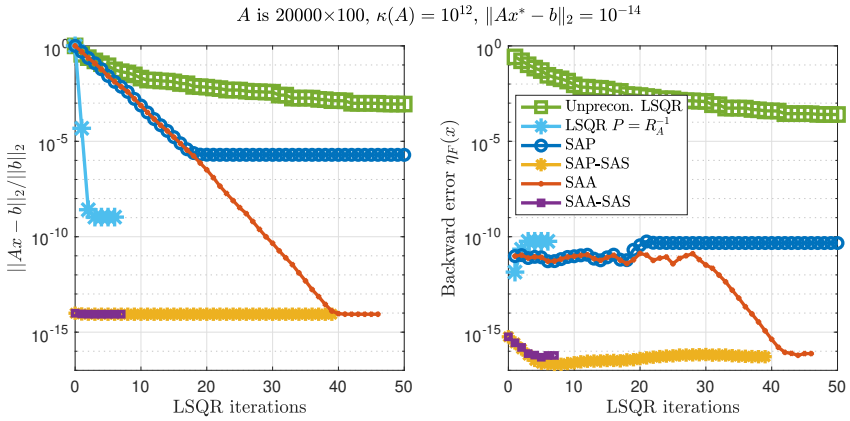

FIG. 1.2. *A comparison of the convergence of the relative residual and backward errors for unpreconditioned LSQR, preconditioned LSQR with $P = R_A^{-1}$ from the QR factorization of $A = Q_A R_A$, SAP Blendenpik with and without SAS initialization, SAA Blendenpik with and without SAS initialization. The tolerance is set to machine precision. Note: color appears only in the online article.*

attainable accuracy. However, we do not observe qualitative differences provided that $\kappa_2(AP) = \mathcal{O}(1)$.

While we have not spotted the precise source of instability in the standard preconditioned LSQR routines, we believe it lies in the fact that each application of the preconditioner P—which involves solving an ill-conditioned linear system $Rx = b$—incurs a relative error proportional to $u\kappa_2(R)$. Such errors are present every time we apply P or P^T. Moreover, each application behaves somewhat differently, in that in the ith iterate we have $(R + \Delta R_i)\hat{x}_i = b_i + \delta b_i$, where $\Delta R_i, \delta b_i$ are small but different for each i. In other words, one can view the preconditioner as having a $u\kappa_2(R)$ nonlinear effect. The fact that the convergence of iterative methods can get impaired by nonlinear preconditioners has been observed in [35].

1.6. Paper structure. In section 2, we introduce some background material on SAP LS solvers, sketching, numerical stability analysis, and smoothed analysis. In section 3, we consider SAA Blendenpik (see Algorithm 1.3) in finite precision. In section 4, we consider extremely ill-conditioned LS problems and look at the numerical stability of smoothed SAA Blendenpik (see Algorithm 1.4). We introduce a master algorithm and display numerical experiments in section 5. Finally, in section 6, we conclude by noting the practical significance of the SAS initialization, and discuss the wider implications of the observed instabilities of standard SAP.

While in this paper we focus on LS problems, we expect much of the stability results to carry over to solving underdetermined linear systems using sketching, as done in LSRN [24].

2. Background material. We now introduce some background material for SAP techniques (see subsection 2.1), random sketching matrices (see subsection 2.2), numerical stability analysis (see subsection 2.3), and smoothed analysis (see subsection 2.4).

2.1. SAP LS solvers. The idea behind the SAP technique is that a reasonable preconditioner for the LS problem in (1.1) can be constructed from a sketch of A.

In Blendenpik, the matrix A is sketched to a small tall-skinny matrix SA and the preconditioner is taken to be the inverse of the upper triangular factor from a QR factorization of SA, i.e., $P = R^{-1}$. In exact arithmetic, the condition number of AP is equal to SQ_A, where Q_A is an orthonormal basis for the column space of A.

LEMMA 2.1. *Let $A \in \mathbb{R}^{m\times n}$ have linearly independent columns, $S \in \mathbb{R}^{s\times m}$ with $s \geq n$ have linearly independent rows, and $B = SA$. If $A = Q_A R_A$ and $B = QR$ are economized QR factorizations of A and B, respectively, then*

$$\kappa_2(AP) = \kappa_2(SQ_A),$$

where $P = R^{-1}$.

Proof. The proof follows the same argument by Meng, Saunders, and Mahoney in [24, Lem. 4.2] and Rokhlin and Tygert in [28, Thm. 1] Note $AP = Q_A R_A R^{-1}$ so that $\kappa_2(AP) = \kappa_2(R_A R^{-1})$. Now, since $B = SA = QR = S(Q_A R_A)$, we have that $(SQ_A)R_A R^{-1} = Q$ is orthonormal. Hence, $\kappa_2(R_A R^{-1}) = \kappa_2(SQ_A)$. ∎

Lemma 2.1 shows that the main idea behind SAP techniques is excellent in exact arithmetic. The value of $\kappa_2(AP)$ is independent of $\kappa_2(A)$. In particular, one expects rapid convergence of iterative LS solvers using the preconditioner P. While this is correct in exact arithmetic, we have seen that rounding errors cause significant problems for ill-conditioned LS problems (see subsection 1.5). Regardless, in exact arithmetic, we are left with the task of designing a sketching matrix so that $\kappa_2(SQ_A)$ is close to 1 with high probability.

2.1.1. SAS initialization. Rokhlin and Tygert [28] included an SAS initial guess in the description of their original SAP algorithm. This initial guess is the solution to $\min \|S(Ax - b)\|_2$ computed with a direct solver. It can be computed using the QR decomposition of SA, which is necessary in any case. The accuracy of SAS solutions are well studied and bounds are generally of the form [22]

$$\|Ax_0 - b\|_2 \leq \frac{1+\epsilon}{1-\epsilon}\|Ax^* - b\|_2 \quad \text{for } s \sim n\log(n)/\epsilon^2,$$

where x_0 is the SAS solution (see (1.3)), x^* is the optimal solution to (1.1), and s is the sketch dimension ($S \in \mathbb{R}^{s\times m}$). Note that ϵ is the subspace embedding constant of $[A, b]$, the concatenation of A and b, and is usually modestly small, say $\epsilon = 0.5$. In the language used in Lemma 2.1, $\frac{1+\epsilon}{1-\epsilon} = \kappa_2(SQ_{[A,b]})$, whose value can be bounded using identical arguments to those for $\kappa_2(SQ_A)$, which will be discussed below. It follows that for a sufficiently large sketch dimension, the accuracy (in terms of residual) of the SAS solution is of the same order as that of the optimal solution.

2.2. Sketching matrices. Given Lemma 2.1, it is paramount to understand how to construct a sketching matrix S so that SQ_A is well-conditioned. Matrices that achieve this are called sketching matrices. We generally desire $m \gg s > n$ so that SA has full column rank and computing a preconditioner from SA is computationally efficient.

There are various ways to construct sketching matrices; almost all are randomly generated. We consider two important types here: (1) Gaussian matrices [24, 33, 36], and (2) subsampled randomized trigonometric transforms (SRTTs) [1, 22, 28, 31]. Other sketching techniques include random sampling [21], sparse embeddings [9], and hashing matrices [7].

2.2.1. Gaussian sketching matrices. A Gaussian sketching matrix of size $s \times m$ is a matrix with i.i.d. Gaussian entries of mean 0 and variance $1/s$. If one selects $s \geq \lceil cn \rceil$ with $c > 1$, then one can show that $\kappa_2(SQ_A)$ is a small constant depending on c with high probability [22]. The drawback of Gaussian matrices is the cost of computing SA. In particular, SA costs $\mathcal{O}(mn^2)$ operations as $s = \mathcal{O}(n)$. However, Gaussian sketching matrices are often used in theoretical analysis due to the availability of excellent probability theory and concentration of measure arguments [15, 22].

2.2.2. SRTTs. An SRTT has the form $S = \mathcal{S}FD$, where D is a square diagonal matrix with ± 1 entries at random, F is an orthogonal trigonometric transform (e.g., Fourier, cosine, or Hadamard), and $\mathcal{S}$ is an $s \times m$ scaled sampling matrix with one nonzero entry per row. For an SRTT, if $s \geq \lceil cn \log n \rceil$ with some constant $c > 1$, then SQ_A is well-conditioned with high probability [31]. In practice, the $\log n$ factor in s can often be dropped [22]. Moreover, the matrix-matrix product SA can be computed in $\mathcal{O}(mn \log s)$ operations, although the usual computational cost is $\mathcal{O}(mn \log m)$ associated with computing $\mathcal{S}(F(DA))$, as fast implementations for the SRTT are not readily available. We use subsampled randomized cosine transforms as the sketching matrices in our numerical experiments.

2.3. Numerical stability analysis. Basic arithmetic operations (e.g., $+$, $-$, $\cdot$, and $/$) on a computer are performed with rounding errors due to the finite precision of numbers in floating-point representation. The stability of a numerical algorithm refers to the property to compute solutions with a small backward error [18] in the presence of rounding errors. We are particularly interested in proving that SAA Blendenpik computes backward stable solutions. The backward error for LS problems is defined in (1.2). A backward stable algorithm computes solutions with a backward error of the same order as the precision.

We assume that computations are performed with a precision of $u \ll 1$ and that numbers and arithmetic operations are exact up to this precision. We now consider three example algorithms: (1) the standard algorithm for matrix-matrix multiplication, (2) Householder QR, and (3) triangular solve with substitution. We need these results to understand SAA LS solvers.

2.3.1. Matrix-matrix multiplication. Consider matrix-matrix multiplication of two matrices $A \in \mathbb{R}^{m \times n}$ and $B \in \mathbb{R}^{n \times p}$. It is shown that the standard algorithm for computing AB satisfies [18, Chap. 3]

$$|AB - \mathrm{fl}(AB)| \leq \frac{nu}{1-nu}|A||B| = \gamma_n |A||B|, \qquad \gamma_n = \frac{nu}{1-nu}, \tag{2.1}$$

where $\mathrm{fl}(AB)$ means that AB is computed with precision u and $|A|$ denotes the entrywise absolute value of A. The inequality in (2.1) is understood as holding for each entry. It can be shown that (2.1) leads to the bound

$$\|AB - \mathrm{fl}(AB)\|_2 \leq \gamma_n \min(\sqrt{m}, \sqrt{n}) \min(\sqrt{n}, \sqrt{p}) \|A\|_2 \|B\|_2, \tag{2.2}$$

where $\|\cdot\|_2$ is the spectral norm. The inequality in (2.2) is a forward error in the sense that it shows that $\mathrm{fl}(AB)$ is close to AB.

2.3.2. Householder QR factorization. Next, we consider the Householder QR of a matrix $A \in \mathbb{R}^{m \times n}$, where $m \geq n$. We denote the triangular factor computed by the Householder QR algorithm as $\hat{R} \in \mathbb{R}^{n \times n}$, and we are interested in the accuracy

of $\hat{R}$. By [18, Thm. 19.4], there exists a matrix $Q \in \mathbb{R}^{m\times n}$ with orthonormal columns such that

$$A + E = Q\hat{R}, \qquad \|e_j\|_2 \le \tilde{\gamma}_{mn}\|a_j\|_2, \quad 1 \le j \le n,$$

where e_j and a_j denote the jth columns of E and A, respectively, and $\tilde{\gamma}_{mn} = cmnu/(1-cmnu)$ for a small integer constant c. A direct consequence is that

$$\|E\|_F \le \tilde{\gamma}_{mn}\|A\|_F, \tag{2.3}$$

where $\|\cdot\|_F$ is the matrix Frobenius norm. The inequality in (2.3) has the interpretation that the QR factorization computed by Householder QR is exact for a slightly perturbed matrix A.

2.3.3. Triangular system solving with substitution. Finally, we consider the error arising from solving a triangular system with substitution. Let $R \in \mathbb{R}^{n\times n}$ be a nonsingular upper triangular matrix. It is known that the computed solution, $\hat{x}$ of the linear system $R^T x = b$ satisfies [18, sect. 8.2]

$$\|x - \hat{x}\|_2 \le \frac{\sqrt{n}\gamma_n\kappa_2(R)}{1-\sqrt{n}\gamma_n\kappa_2(R)}\|x\|_2, \tag{2.4}$$

where γ_n is defined in (2.1). The inequality in (2.4) tells us that we expect the relative forward error of the computed solution to be on the order of $\kappa_2(R)$.

When we do any SAA technique, we need to compute AP, where $P = R^{-1}$. We compute AP by solving $R^T x_i = a_i$ for $1 \le i \le m$, where a_i^T denotes the ith row of A and x_i^T is the ith row of AP. If we denote the computed solution to $R^T x_i = a_i$ by $\hat{x}_i$, then from (2.4) we find that

$$\sum_{i=1}^{m}\|x_i - \hat{x}_i\|_2^2 \le \left[\frac{\sqrt{n}\gamma_n\kappa_2(R)}{1-\sqrt{n}\gamma_n\kappa_2(R)}\right]^2 \sum_{i=1}^{m}\|x_i\|_2^2.$$

We conclude that for $P = R^{-1}$ we have

$$\|AP - \widehat{AP}\|_F \le \frac{\sqrt{n}\gamma_n\kappa_2(P)}{1-\sqrt{n}\gamma_n\kappa_2(P)}\|AP\|_F, \tag{2.5}$$

where $\widehat{AP}$ denotes the computed matrix-matrix product. Roughly speaking, $\widehat{AP}$ is close to AP when $\kappa_2(P)$ is modest. We additionally bound the backward error resulting from solving $Rx = b$ with back substitution. The computed solution $\hat{x}$ satisfies [18, Thm. 8.5]

$$(\hat{R} + \Delta\hat{R})\hat{x} = b, \quad \|\Delta\hat{R}\|_2 \le \gamma_n\sqrt{n}\|\hat{R}\|_2. \tag{2.6}$$

2.4. Smoothed analysis of condition numbers. Due to rounding errors, most matrices with $\kappa_2(A) > u^{-1}$ are perturbed to a matrix with condition number less than u^{-1} once represented on a computer. To explain this, we can model the rounding errors by an additive Gaussian perturbation to A. This type of technique fits into the field of smoothed analysis and has been studied in [19]. More precisely, we suppose that $A \in \mathbb{R}^{m\times n}$ and that we would like to bound the condition number of $A + \sigma G$, where σ is a small number and G is a standard Gaussian matrix with i.i.d. entries. We have the following statement, which is a corollary of [6, Thm. 1.1].

COROLLARY 2.2. *Let $A \in \mathbb{R}^{m\times n}$ with $m \geq 3n$ and $r \geq 10^4$. If A is perturbed to $\hat{A} = A + \sigma G/\sqrt{m}$ with $\sigma = 8.25\|A\|_2/r$, where G is standard Gaussian matrix with i.i.d. entries, then*

$$\mathbb{P}\Big[\kappa_2(\hat{A}) \geq r\Big] < 2^{n-m}.$$

Proof. The result immediately follows by setting $z = (r(1-\lambda))/e$ and $\sigma = 8.25/(r\sqrt{m})$ in Theorem 1.1 of [6]. This takes into account that Theorem 1.1 of [6] considers the case when $m < n$. The extra $\|A\|_2$ factor of σ in the statement of the corollary ensures that our final result continues to hold when $\|A\|_2 \neq 1$. ∎

Corollary 2.2 shows us that a small additive Gaussian perturbation to a matrix A ensures that the condition number is small with high probability. It partially explains why matrices represented in floating-point arithmetic rarely have a condition number $> u^{-1}$. In section 4, we use Corollary 2.2 to explain why SAA techniques continue to deliver accurate LS solutions, even when we have $\kappa_2(A) > u^{-1}$.

3. On the numerical stability of SAA Blendenpik. By the properties of sketching matrices and Lemma 2.1, we know that if one computes P and AP in exact arithmetic, then $\kappa_2(AP)$ is bounded by a constant independent of $\kappa_2(A)$ with high probability. In finite precision, one typically needs to be more careful. In the first part of this section, we show that when $\kappa_2(A) \ll u^{-1}$, the condition number of the computed matrix AP can also be bounded by a constant with high probability (see Theorem 3.4). Unfortunately, such a bound does not extend to the case when $\kappa_2(A) > u^{-1}$ (see section 4).

There are three main steps to compute AP. In finite precision, the computed matrices satisfy the following:

1. $\hat{B} = SA + E_1$,
2. $\tilde{Q}\hat{R} = \hat{B} + E_2$,
3. $\hat{Y} = A\hat{R}^{-1} + \Delta Y$,

where hats denote that we are accounting for rounding errors. The matrix-matrix multiplication error E_1 can be bounded using (2.2). We have a bound on the error arising from the Householder QR algorithm E_2 in (2.3). Note that $\tilde{Q}$ has orthonormal columns but is not equal to Q, where Q is the orthonormal factor of the exact QR factorization of $B = SA$. Finally, the error term ΔY in the step resulting from solving the triangular matrix system can be bounded using (2.5). After all these errors, we want to derive a bound on $\kappa_2(\hat{Y})$, as $\hat{Y}$ is the computed preconditioned matrix in finite precision.

3.1. Bounding $\kappa_2(\hat{\mathbf{Y}})$ in terms of numerical errors. First, let $E = E_1 + E_2$ be the sum of the additive errors from the matrix-matrix multiplication and the QR factorization, such that $\tilde{Q}\hat{R} = QR + E$. We show in Lemma 3.1 that the quantity $\varepsilon_1 := \|E\|_2\|R^{-1}\|_2$ controls the numerical error in the first two steps.

LEMMA 3.1. *Let $A = Q_A R_A$ and $SA = QR$ be economized QR factorizations of A and $B = SA$, respectively. Let $\hat{B}$ be the computed SA and $\hat{R}$ be the computed upper triangular factor from the Householder QR algorithm of $\hat{B}$, both computed in finite precision. Let $\tilde{Q}$ be a matrix with orthonormal columns such that $E = \tilde{Q}\hat{R} - QR$ is the numerical error associated with $\hat{R}$. Define $\varepsilon_1 := \|E\|_2\|R^{-1}\|_2$. Assuming that $\varepsilon_1 < 1$, we find that $\hat{R}$ is invertible and $A\hat{R}^{-1}$ is full rank. Furthermore,*

$$\kappa_2(\hat{R}) \leq \frac{\kappa_2(R) + \varepsilon_1}{1-\varepsilon_1} \leq \frac{\kappa_2(SQ_A)\kappa_2(A) + \varepsilon_1}{1-\varepsilon_1}$$

and

$$\kappa_2(A\hat{R}^{-1}) \leq \kappa_2(SQ_A)\frac{1+\varepsilon_1}{1-\varepsilon_1}.$$

Proof. We have

$$\|\hat{R}\|_2 = \|\tilde{Q}\hat{R}\|_2 = \|QR+E\|_2 \leq \|R\|_2 + \|E\|_2 \tag{3.1}$$

and

$$\sigma_{\min}(\hat{R}) = \sigma_{\min}(QR+E) \geq \sigma_{\min}(R) - \|E\|_2. \tag{3.2}$$

We see that $\sigma_{\min}(\hat{R}) > 0$ provided that $\varepsilon_1 < 1$. We can combine (3.1) and (3.2) together to obtain an upper bound on $\kappa_2(\hat{R})$. To bound $\kappa_2(A\hat{R}^{-1})$, we use the fact that[3] $A\hat{R}^{-1}\tilde{Q}^T = AR^{-1}(Q+ER^{-1})^\dagger$ to find that for any $v \in \mathbb{R}^m$, we have

$$\|v^T AR^{-1}\|_2(1+\varepsilon_1)^{-1} \leq \|v^T A\hat{R}^{-1}\|_2 \leq \|v^T AR^{-1}\|_2(1-\varepsilon_1)^{-1}. \tag{3.3}$$

The bound follows as $\kappa_2(AR^{-1}) = \kappa_2(SQ_A)$ (see Lemma 2.1). ∎

Lemma 3.1 shows us, assuming ε_1 is sufficiently small, that $\kappa_2(\hat{R})$ cannot be larger than about $\kappa_2(SQ_A)\kappa_2(A)$, and $\kappa_2(A\hat{R}^{-1})$ cannot be much larger than $\kappa_2(SQ_A)$.

As the following lemma shows, the error incurred by solving the triangular system depends on the quantities in Lemma 3.1.

LEMMA 3.2. *Let $\hat{R}^{-1}$ be the upper triangular matrix defined in Lemma* 3.1 *and let $\hat{Y}$ be the matrix obtained by computing $A\hat{R}^{-1}$ with forward substitution in finite precision. Define $\Delta Y = \hat{Y} - A\hat{R}^{-1}$ and $\varepsilon_2 = \|\Delta Y\|_2\|(A\hat{R}^{-1})^\dagger\|_2$. Assuming that $\varepsilon_2 < 1$, we find that $\hat{Y}$ is full rank and*

$$\kappa_2(\hat{Y}) \leq \frac{\kappa_2(A\hat{R}^{-1})+\varepsilon_2}{1-\varepsilon_2}.$$

Proof. This follows immediately from the relationship $\hat{Y} = A\hat{R}^{-1} + \Delta Y$ and similar reasoning to the proof of Lemma 3.1. ∎

In exact arithmetic, clearly $\kappa_2(Y) = \kappa_2(AR^{-1})$. The numerical error incurred due to the triangular solve is controlled by $\|\Delta Y\|$, which depends on $\kappa_2(\hat{R})$ and $\kappa_2(A\hat{R}^{-1})$. We show that the error is small under the assumption that $\kappa_2(A)u \ll 1$. As a result, the condition number of $\hat{Y}$ will be bounded by a small constant related to the conditioning of S and SQ_A. Thus, the computed AP has a reasonable condition number provided that the numerical errors ε_1 and ε_2 are small.

3.2. Bounding the numerical errors. We now bound the errors ε_1 and ε_2.

LEMMA 3.3. *Assume the same setup as in Lemmas* 3.1 *and* 3.2, *where $\varepsilon_1 = \|E\|_2\|R^{-1}\|_2$ and $\varepsilon_2 = \|\Delta Y\|_2\|(A\hat{R}^{-1})^\dagger\|_2$, and all quantities in finite precision are computed with unit roundoff u. Furthermore, define*

$$C_1 := \sqrt{sn}\gamma_m + \sqrt{n}\tilde{\gamma}_{sn}(1+\sqrt{sn}\gamma_m), \qquad k(S) := \|S\|_2\|(SQ_A)^\dagger\|_2. \tag{3.4}$$

If $\kappa_2(A) < 1/(C_1k(S))$, then we have

$$\varepsilon_1 \leq C_1k(S)\kappa_2(A),$$

[3]The "†" superscript on a matrix denotes the matrix pseudoinverse.

and

$$\varepsilon_2 \le \frac{n\gamma_n\kappa_2(SQ_A)(\kappa_2(A)\kappa_2(SQ_A)+\varepsilon_1)(1+\varepsilon_1)}{(1-\varepsilon_1-\sqrt{n}\gamma_n(\kappa_2(A)\kappa_2(SQ_A)+\varepsilon_1))\,(1-\varepsilon_1)}.$$

Proof. We first note that $E = E_1 + E_2$, where $E_1 = \hat{B} - SA$ is the error arising from matrix-matrix multiplication and $E_2 = \tilde{Q}\hat{R} - \hat{B}$ is the error arising from the Householder QR algorithm. We can bound these terms using (2.2) and (2.3) to find

$$\begin{aligned}\|E\|_2 &\le \|E_1\|_2 + \sqrt{n}\tilde{\gamma}_{sn}\|SA + E_1\|_2 \\ &\le \sqrt{n}\tilde{\gamma}_{sn}\|S\|_2\|A\|_2 + (1+\sqrt{n}\tilde{\gamma}_{sn})\sqrt{sn}\gamma_m\|S\|_2\|A\|_2 = C_1\|S\|_2\|A\|_2.\end{aligned}$$

This is then combined with

$$\sigma_{\min}(R) = \sigma_{\min}(QR) = \sigma_{\min}(SQ_AR_A) \ge \sigma_{\min}(SQ_A)\sigma_{\min}(A).$$

As for $\|\Delta Y\|_2$, by (2.5) we have

$$\|\Delta Y\|_2 \le \|\Delta Y\|_F \le \frac{\sqrt{n}\gamma_n\kappa_2(\hat{R})}{1-\sqrt{n}\gamma_n\kappa_2(\hat{R})}\|A\hat{R}^{-1}\|_F \le \phi(\hat{R})\|A\hat{R}^{-1}\|_2,$$

where

$$\phi(\hat{R}) := \frac{n\gamma_n\kappa_2(\hat{R})}{1-\sqrt{n}\gamma_n\kappa_2(\hat{R})} \le \frac{n\gamma_n(\kappa_2(A)\kappa_2(SQ_A)+\varepsilon_1)}{1-\varepsilon_1-\sqrt{n}\gamma_n(\kappa_2(A)\kappa_2(SQ_A)+\varepsilon_1)}. \tag{3.5}$$

We have $\varepsilon_2 \le \phi(\hat{R})\kappa_2(A\hat{R}^{-1})$, which can be bounded using the result on $\kappa_2(A\hat{R}^{-1})$ in Lemma 3.1. ∎

The condition that $\kappa_2(A) < 1/(C_1k(S))$ is closely related to the assumption $\kappa_2(A) \ll u^{-1}$ (see subsection 3.3 for further discussion), which ensures that $\varepsilon_1 < \kappa_2(A)u \ll 1$. The bound $\varepsilon_2 \le \phi(\hat{R})\kappa_2(A\hat{R}^{-1})$, where $\phi(\hat{R})$ is defined in (3.5), tells us that the dominant term in the bound for ε_2 is $n^2\kappa_2(SQ_A)\kappa_2(A)u$, when $\varepsilon_1 < 1$.

3.3. Bounding $\kappa_2(\hat{\mathbf{Y}})$. We now combine Lemmas 3.1 to 3.3 to obtain a bound on the condition number of $\hat{Y}$, which is the computed version of AR^{-1}. We find that $\kappa_2(\hat{Y}) \le 4\kappa_2(SQ_A) + 1$ provided that $\kappa_2(A)$ is sufficiently small.

THEOREM 3.4. *Let $A \in \mathbb{R}^{m\times n}$ have linearly independent columns, $S \in \mathbb{R}^{s\times n}$ have linearly independent rows, and $m > s > n$. Let $A = Q_AR_A$ and $SA = QR$ be the economized QR factorizations of A and $B = SA$, respectively. Assume that*

$$\kappa_2(SQ_A) < 49, \quad n^2u < 1/201, \quad \frac{m}{n} + \sqrt{n} > 200, \tag{3.6}$$

and that $\kappa_2(A)$ is sufficiently small so that

$$\kappa_2(A) < \frac{1}{3C_1k(S)}, \tag{3.7}$$

where $C_1 = \mathcal{O}((\sqrt{sn}m + sn^{3/2})u)$ and $k(S)$ are defined in (3.4)*. Compute the preconditioned matrix $\hat{Y}$ as detailed in Lemmas* 3.1 *and* 3.2.[4] *Then,*

$$\kappa_2(\hat{Y}) \le 4\kappa_2(SQ_A) + 1.$$

[4]That is, compute the matrix-matrix product SA, compute its R factor by the Householder QR algorithm, and finally compute the preconditioned matrix by solving AR^{-1} with forward substitution.

Proof. Throughout the proof, we assume the notation introduced in Lemmas 3.1 to 3.3. By assumption (3.7) we immediately see that $\varepsilon_1 < 1/3$, defined in Lemma 3.1, and hence $\hat{R}$ and $A\hat{R}^{-1}$ are full rank. It follows that $\kappa_2(A\hat{R}^{-1}) \le 2\kappa_2(SQ_A)$. Furthermore, also by Lemma 3.1, we find

$$\kappa_2(\hat{R}) \le \frac{\kappa_2(R)+\varepsilon_1}{1-\varepsilon_1} \le \frac{1}{2}(3\kappa_2(SQ_A)\kappa_2(A)+1) < \frac{1}{2}\left(\frac{1}{C_1}+1\right). \tag{3.8}$$

By using the fact that $m > s > n$, we can show that $C_1 > (m+n^{3/2})\gamma_n$, which can be substituted into (3.8). One can use the bound on $\kappa_2(\hat{R})$ to then bound $\phi(\hat{R})$, defined in (3.5), as

$$\phi(\hat{R}) = \frac{n\gamma_n\kappa_2(\hat{R})}{1-\sqrt{n}\gamma_n\kappa_2(\hat{R})} < \frac{\frac{1}{2}n\left(\frac{1}{m+n^{3/2}}+\gamma_n\right)}{1-\frac{1}{2}\sqrt{n}\left(\frac{1}{m+n^{3/2}}+\gamma_n\right)} = \frac{\sqrt{n}c(m,n)}{1-c(m,n)},$$

where

$$c(m,n) = \frac{1}{2}\sqrt{n}\left(\frac{1}{m+n^{3/2}}+\gamma_n\right).$$

The assumptions in (3.6) imply that $c(m,n) < (4\sqrt{n}\kappa_2(SQ_A)+1)^{-1}$, where we used that $n^2u < 1/201$, i.e., $n\gamma_n < 1/200$. Now we have that $\phi(\hat{R}) < (4\kappa_2(SQ_A))^{-1}$ and so

$$\varepsilon_2 \le \phi(\hat{R})\kappa_2(A\hat{R}^{-1}) \le \kappa_2(SQ_A)\phi(\hat{R})\frac{1+\varepsilon_1}{1-\varepsilon_1} < \frac{1}{2}.$$

By Lemma 3.2, $\hat{Y}$ is full rank and its condition number is bounded by

$$\kappa_2(\hat{Y}) \le \frac{\kappa_2(A\hat{R}^{-1})+\varepsilon_2}{1-\varepsilon_2} < 4\kappa_2(SQ_A)+1. \qquad \square$$

It is worth discussing the assumptions of Theorem 3.4. Consider the assumption in (3.7). The $k(S)$ term (see (3.4)) is a small constant depending on the particulars of the embedding matrix. The constant C_1 (see (3.4)) can be bounded by the product of a low-degree polynomial in m, n, and s and the unit roundoff u, say $C_1 \le p(m,n,s)u$. We specifically have $C_1 = \mathcal{O}(\sqrt{sn}(m+\sqrt{sn})u)$, as $\gamma_m = \mathcal{O}(mu)$ and $\tilde{\gamma}_{sn} = \mathcal{O}(snu)$. In classical stability analysis, this polynomial term $p(m,n,s)$ can be more or less ignored. As a result, a condition of the form $\kappa_2(A) < (p(m,n,s)u)^{-1}$ is often loosely restated as $\kappa_2(A) \ll u^{-1}$.

The rationale behind ignoring these m, n, and s terms is that classic stability analysis, as we performed in this section, provides us with worst-case bounds on numerical errors. These are generally pessimistic [18]; composing n operations would only result in an error of order nu if each rounding error is of the same sign and of maximum magnitude [19]. We could improve these bounds by using probabilistic backward error analysis, which would give us results proportional to $\sqrt{n}u$ instead of nu [19, 20]. However, it must be noted that even probabilistic stability analysis would theoretically result in a large factor multiplied by u.

The other assumptions of Theorem 3.4, in (3.6), are reasonable in practice. The condition number of SQ_A is generally observed to be bounded by a modest number, such as 5 or 10, with high probability. Furthermore, we assume that the problem dimensions are small compared to the unit roundoff. Most importantly, Theorem 3.4 informs us that, provided that $\kappa_2(A) \ll u^{-1}$, the condition number of the preconditioned matrix is independent of $\kappa_2(A)$ (with high probability).

3.4. The backward error of SAA Blendenpik. Until now, we have only considered the condition number of the preconditioned matrix AP. The analysis in this section related rounding errors incurred in the final steps—solving $(AR^{-1})z = b$ with LSQR and computing $x = R^{-1}z$ with back substitution—to the backward error of SAA. It is not immediate that the last step in the algorithm preserves the numerical stability of the algorithm. After all, solving linear systems with R in LSQR iterations is one potential reason for the numerical instability of SAP Blendenpik. Nonetheless, we show that SAA Blendenpik performs as well as LSQR on very well-conditioned matrices and that the numerical error in the last step is $\mathcal{O}(u)$ instead of $\mathcal{O}(u\kappa_2(A))$ (see Theorem 3.5).

Throughout this analysis, we will not consider an SAS initialization and consider stability properties from any starting point.

We show that the backward error in SAA Blendenpik is of the same order as the backward error from unpreconditioned LSQR with $\mathrm{fl}(AR^{-1})$, under the assumptions of Theorem 3.4.

THEOREM 3.5. *Assume the embedding matrix $S \in \mathbb{R}^{s\times m}$ and $A \in \mathbb{R}^{m\times n}$ satisfy assumptions* (3.6) *and* (3.7) *in Theorem* 3.4. *Apply Algorithm* 1.3 *and assume that LSQR terminates at an iterate $\hat{z}$ that satisfies the backward error*[5]

$$\hat{z} = (\hat{Y} + \Delta\hat{Y})^{\dagger}(b + \delta b). \tag{3.9}$$

Then, the computed solution $\hat{x}$ to $Ax = b$ has the backward error $(A + \Delta A)\hat{x} = b + \delta b$ satisfying

$$\|\Delta A\|_2 < \|S\|_2\|A\|_2 \left(6.04n\gamma_n\|(SQ_A)^{\dagger}\|_2 + 2.01\|\Delta\hat{Y}\|_2\right).$$

Proof. The final step of Algorithm 1.3 involves solving the triangular system $\hat{R}x = \hat{z}$. We know from (2.6) that the computed solution $\hat{x}$ satisfies

$$(\hat{R} + \Delta\hat{R})\hat{x} = \hat{z}, \quad \|\Delta\hat{R}\|_2 \le \gamma_n\sqrt{n}\|\hat{R}\|_2.$$

We now have

$$\hat{x} = (\hat{R} + \Delta\hat{R})^{-1}\hat{z} = (\hat{R} + \Delta\hat{R})^{-1}(\hat{Y} + \Delta\hat{Y})^{\dagger}(b + \delta b) = (A + \Delta A)^{\dagger}(b + \delta b),$$

where $\Delta A = (\hat{Y}\hat{R} - A) + \Delta\hat{Y}\hat{R} + \hat{Y}\Delta\hat{R} + \Delta\hat{Y}\Delta\hat{R}$. For the $\hat{Y}\hat{R} - A$ term we have that each row satisfies

$$\hat{y}_i^T\hat{R} - a_i^T = -\hat{y}_i^T\Delta_i\hat{R}, \qquad |\Delta_i\hat{R}| \le \gamma_n|\hat{R}|,$$

so that $\|\hat{Y}\hat{R} - A\|_2 \le n\gamma_n\|\hat{R}\|_2\|\hat{Y}\|_2$. It follows that

$$\|\Delta A\|_2 \le \|\hat{R}\|_2 \left((n + \sqrt{n})\gamma_n\|\hat{Y}\|_2 + (1 + \sqrt{n}\gamma_n)\|\Delta\hat{Y}\|_2\right),$$

where $\|\hat{R}\|_2 \le (1 + C_1)\|S\|_2\|A\|_2 < 2\|S\|_2\|A\|_2$, and

$$\|\hat{Y}\|_2 \le \|A\hat{R}^{-1}\|_2 + \|\Delta Y\|_2 \le \left(1 + \phi(\hat{R})\right)\frac{\|(SQ_A)^{\dagger}\|_2}{1 - \varepsilon_1} < 1.51\|(SQ_A)^{\dagger}\|_2,$$

[5]Note that the backward error term $\Delta\hat{Y}$ is not the same as the numerical error ΔY resulting from solving the triangular system $Y = A\hat{R}^{-1}$ in finite precision.

where the last inequality follows from the assumptions (3.6) and (3.7). We obtain

$$\|\Delta A\| \le 2\|S\|_2\|A\|_2\left(3.02n\gamma_n\|(SQ_A)^\dagger\|_2 + 1.005\|\Delta\hat{Y}\|_2\right),$$

where we used that $n+\sqrt{n}\le 2n$ and $\sqrt{n}\gamma_n \le n\gamma_n \le 1/200$ by (3.6). ∎

The moral of Theorem 3.5 is that SAA Blendenpik is backward stable as long as LSQR on $\hat{Y}$ is backward stable, that is, $\|\Delta\hat{Y}\|_2$ and $\|\delta b\|_2/\|b\|_2$ are both $\mathcal{O}(u)$ in (3.9). Under the conditions of Theorem 3.5, the matrix $\hat{Y}$ is well-conditioned with high probability. The backward stability of LSQR on a well-conditioned matrix is not an undisputedly clear fact. Practical evidence is abundant and several studies [4, 12, 13, 14, 25] have addressed the convergence of the conjugate gradient (CG) method (LSQR is a stable implementation of CG applied to the normal equation); we are unaware of a precise result that proves CG (or LSQR) applied to a well-conditioned system is backward stable. Such stability analysis would depend on the precise implementation of the algorithm (see section 6).

Remarkably, the final step of the algorithm does not influence the accuracy of the solution, i.e., solving $Rx = y$ with back substitution. This is despite the fact that R is ill-conditioned when A is. The mechanism with which stability is established is similar to the backward stability of an algorithm based on repeated CholeskyQR [37].

4. On the numerical stability of smoothed SAA Blendenpik. We now analyze smoothed SAA Blendenpik (see Algorithm 1.4). When $\kappa_2(A) \ll 1/u$, Theorem 3.4 states that SAA computes a preconditioned matrix with a condition number independent of $\kappa_2(A)$ with high probability. Moreover, we can reliably estimate $\kappa_2(A)$ by $\kappa_2(SA)$. We do not have a guarantee on the stability of SAA Blendenpik if $\kappa_2(A) \gtrsim 1/u$. It is in this context that we consider smoothing.

One could consider Algorithm 1.4 from two points of view. First, a practitioner may add an $\mathcal{O}(u\|A\|_2)$ Gaussian perturbation to a severely ill-conditioned problem. As a result, the perturbed problem can be accurately solved with SAA. Second, floating-point errors arising from the representation of any matrix in finite precision could be modeled as a Gaussian perturbation and then smoothing partially explains the success of SAA to highly ill-conditioned LS problems.

4.1. Bounding $\kappa_2(\hat{\mathbf{Y}})$. Similarly to the stability analysis in section 3, we have the following steps in finite precision after we have smoothed A,

1. $\tilde{A} = A + \sigma G/\sqrt{m}$,
2. $\hat{B} = S\tilde{A} + E_1$,
3. $\tilde{Q}\hat{R} = \hat{B} + E_2$,
4. $\hat{Y} = \tilde{A}\hat{R}^{-1} + \Delta Y$,

where the hats denote that we are accounting for rounding errors and the scaling parameter σ controls how much A is perturbed. Assuming σ is sufficiently large to ensure that $\tilde{A}$ satisfies (3.7), we then show that the stability analysis from section 3 carries over to $\tilde{A}$.

The following theorem tells us how large σ needs to be.

THEOREM 4.1. *Let $A \in \mathbb{R}^{m\times n}$ with $m \ge 3n$, $S \in \mathbb{R}^{s\times n}$ have linearly independent rows, and let $m > s > n$. Set*

$$\tilde{A} = A + \sigma G/\sqrt{m}, \qquad \sigma = 52\|A\|_2 k(S) sm\sqrt{n}u, \tag{4.1}$$

where $G \in \mathbb{R}^{m\times n}$ is a standard Gaussian matrix with i.i.d. entries, $k(S)$ is defined in (3.4)*, and u is the unit roundoff. Let $\tilde{A} = Q_{\tilde{A}}R_{\tilde{A}}$ and $S\tilde{A} = QR$ be economized QR factorizations of $\tilde{A}$ and $B = S\tilde{A}$, respectively. Last, assume that*[6]

$$\kappa_2(SQ_{\tilde{A}}) < 49, \qquad \frac{m}{n} + \sqrt{n} > 200, \qquad smu < 1/67. \tag{4.2}$$

Now compute first the product $S\tilde{A}$, then the upper triangular factor R by the Householder QR algorithm on $S\tilde{A}$, and finally compute $\hat{Y} = \tilde{A}R^{-1}$ with forward substitution. Assume all computations are performed in finite precision. Then,

$$\mathbb{P}\left(\kappa_2(\hat{Y}) > 4\kappa_2(SQ_{\tilde{A}}) + 1\right) < 2^{n-m}.$$

Proof. We start by showing that σ in (4.1) is sufficiently large to ensure that $\tilde{A}$ satisfies (3.7) in Theorem 3.4. To this end, note the assumptions in (4.2) imply that $mu < 1/64$ and $csnu < 1/64$ so that we have $\gamma_m < 1.02mu$ and $\tilde{\gamma}_{sn} < 1.02csnu < 1.02smu$. We can then bound C_1 defined in (3.4) as

$$C_1 < 1.02 \times 65sm(1+\sqrt{n})u/64 < 2.1sm\sqrt{n}u.$$

As a result

$$\mathbb{P}\left(\kappa_2(\tilde{A}) > \frac{1}{3C_1k(S)}\right) < \mathbb{P}\left(\kappa_2(\tilde{A}) > \frac{8.25}{52k(S)sm\sqrt{n}u}\right) < 2^{n-m}.$$

The final result on $\kappa_2(\hat{Y})$ follows by applying Theorem 3.4 to $\tilde{A}$, which is possible since the last assumption in (4.2) is stronger than $nu < 1/201$. ∎

Theorem 4.1 shows that the magnitude of the perturbation $\sigma G/\sqrt{m}$ needs to be approximately $\mathcal{O}(sm\sqrt{n}u)$ to ensure the condition number of $\tilde{A}$ is sufficiently small, that is, $\kappa_2(A) < 1/u$. The low-degree polynomial term $sm\sqrt{n}$ arises from (3.7), which includes the constant $C_1 = \mathcal{O}(\sqrt{sn}(m+\sqrt{sn})u)$. As discussed in subsection 3.4, these are often pessimistic bounds. Probabilistic error analysis provides a theoretical justification to select σ proportional to $\sqrt{smn}n^{1/4}u$. This may still be a relatively large perturbation if the dimensions of the problem are enormous, enlarging the backward error to an unacceptable level (see subsection 4.2).

4.2. The backward error of smoothed SAA Blendenpik. In subsection 4.1, we showed that a small additive random perturbation to a small-skinny matrix ensures that the condition number of the perturbed matrix is sufficiently small. This allows us to conclude that $\kappa_2(\hat{Y})$ is small with high probability. We now show that smoothed SAA Blendenpik (see Algorithm 1.4) computes a backward error similar to Theorem 3.5 with an additional term from the additive perturbation.

THEOREM 4.2. *Assume the same setup as in Theorem* 4.1*. Apply Algorithm* 1.3 *to $\tilde{A}x = b$ and assume LSQR terminates at an iterate $\hat{z}$ that satisfies the backward error*

$$\hat{z} = (\hat{Y} + \Delta\hat{Y})^{\dagger}(b + \delta b).$$

[6]We also assume that $cn < m$ for the same integer c defined in $\tilde{\gamma}_{sn}$ to simplify the notation.

Then, the computed solution $\hat{x}$ to $Ax = b$ has the backward error $(A + \Delta A)\hat{x} = b + \delta b$ satisfying

$$\|\Delta A\|_2 \le \|A\|_2 \left[\|S\|_2 \left(1 + 197k(S)sm\sqrt{n}u\right) \times \left(6.04n\gamma_n \|(SQ_{\tilde{A}})^\dagger\|_2 + 2.01\|\Delta\hat{Y}\|_2\right) + 197k(S)sm\sqrt{n}u \right] \tag{4.3}$$

with probability at least $1 - 2^{n-m} - 10^{-3}$.

Proof. We combine Theorems 3.5 and 4.1 to obtain the result. Additionally, we use the following classic result in the analysis of Gaussian matrices [10]:

$$\mathbb{P}\left(\|G\|_2/\sqrt{m} > 1 + \sqrt{\frac{n}{m}} + \frac{t}{\sqrt{m}} \right) \le e^{-t^2/2},$$

applied with $t = 3.8$. This shows $\|\tilde{A}\|_2 \le \|A\|_2 + \sigma\|G\|_2/\sqrt{m} \le \|A\|_2 + 3.78\sigma$ with probability at least 10^{-3}, where $\sigma = 52\|A\|_2 k(S)sm\sqrt{n}u$. ∎

Theorem 4.2 allows us to conclude that under mild assumptions, smoothed SAA Blendenpik results in a backward error of the same order as the backward error of unpreconditioned LSQR on $\hat{Y}z = b$. Under the conditions of the theorem, $\hat{Y}$ is indeed a well-conditioned matrix and LSQR is expected to obtain backward stable solutions. We removed the assumptions on $\kappa_2(A)$.

Theorem 4.2 suggests that the perturbation needs to be of size $\mathcal{O}(sm\sqrt{n}u)$. This would be impermissible for many problems, especially if the dimensions are large. We recommend a perturbation of magnitude approximately $10\|A\|_2 u$. Although we lack the theoretical evidence for this choice, it works in our experiments (see subsection 5.2).

Smoothing an LS problem should be done with much caution. There are many contexts where $\kappa_2(A) > u^{-1}$, yet perturbing the problem is unnecessary. For instance, if the ill-conditioning is caused by poor column scaling, SAA will still compute accurate solutions without smoothing (see section 6). In this sense, $\kappa_2(A)$ is not an effective predictor of whether smoothing will be beneficial. We suggest one should smooth only if SAA does not converge (see Algorithm 5.1).

5. Master SAA algorithm and numerical experiments. We now present the implementation of the SAA algorithms (see Algorithm 5.1) in a similar fashion to the overall solver in Blendenpik [1]. We perform the standard initial steps, i.e., drawing an embedding matrix S, computing the sketch SA, and computing the QR factorization $QR = SA$ using the Householder QR algorithm. Next, we compute the preconditioned matrix $Y = AR^{-1}$ explicitly with forward substitution and use (unpreconditioned) LSQR to solve $Yz = b$. We include SAS initialization in this algorithm to speed up convergence. This step consists of computing $z_0 = Q^T Sb$. If LSQR converges to the desired tolerance, we compute $x = R^{-1}z$ with back substitution. In this case, we assume our algorithm computed a backward stable solution.

In the case where LSQR does not converge, we perturb/smooth A. We compute $\tilde{A} = A + \sigma G/\sqrt{m}$ for a Gaussian matrix G with $\sigma = 10\|A\|_2 u$ and perform the SAA steps including an SAS initial guess on $\tilde{A}x = b$. We recommend doing this because there are examples, although rare, where SAA Blendenpik converges to an accurate solution only after smoothing (see subsection 5.2). The spectral norm of A can be estimated, for instance, with the sketch SA or using the power method.

Algorithm 5.1. Master algorithm for SAA to solve (1.1). Here, HHQR refers to the Householder QR algorithm.

Draw a random sketching matrix $S \in \mathbb{R}^{s\times m}$, where $m \gg s > n$
Compute $B = SA$ and $c = Sb$
Compute both Q and R of the QR factorization of B using HHQR
Compute $Y = AR^{-1}$ with forward substitution
Compute initial guess $z_0 = Q^T c$
Solve $Yz = b$ with LSQR and no preconditioner and initial guess z_0
if LSQR converges to the desired tolerance **then**
    Compute $x = R^{-1}z$ with back substitution
else
    Draw a random standard Gaussian matrix $G \in \mathbb{R}^{m\times n}$ with i.i.d. entries
    Compute $\tilde{A} = A + \sigma G/\sqrt{m}$ for $\sigma = 10\|A\|_2 u$
    Compute $B = S\tilde{A}$
    Compute both Q and R of the QR factorization of B using HHQR
    Compute $Y = \tilde{A}R^{-1}$ with forward substitution
    Compute initial guess $z_0 = Q^T c$
    Solve $Yz = b$ with LSQR and no preconditioner and initial guess z_0
    Compute $x = R^{-1}z$ with back substitution
end if

It should be noted that without smoothing, an LSQR iteration can converge to a solution with a suboptimal residual (see Figure 5.3 (right)). However, it is infeasible to compute the backward error when m is large as it requires computing the smallest singular value of an $m \times (m + n)$ matrix (although it is possible to reduce the cost [17]). If the residual obtained from SAA is less accurate than desired, one could estimate the condition number of A via the condition number of R. If the condition number estimate of R is of the same order of magnitude as u^{-1}, then the problem is severely ill-conditioned and one could try smoothing to potentially improve the accuracy of the computed solution. Smoothing can improve the convergence rate of LSQR, even if SAA obtains a backward stable solution without smoothing (see Figure 5.2). Nonetheless one should be careful with smoothing because examples exist where $\kappa_2(A)$ is large but smoothing remains unnecessary (see Figure 5.2). Exactly when and when not to smooth remains unclear.

5.1. Numerical experiments with SAA Blendenpik. Now we examine the performance of SAP Blendenpik with SAS initialization, SAA Blendenpik with and without SAS initialization, and the QR-based direct solver. The experiment considers ill-conditioned LS problems that are randomly generated (see subsection 1.5 for details on how A and b are formed) and can be stored in a local cache. The QR solver is implemented with the `qr` and backslash \ (with R) commands in MATLAB. We compare these algorithms for matrices of various dimensions ($n = 4000$, $2^{12} \le m \le 2^{20}$ and $m = 10^6$, $2^7 \le n \le 2^{12}$). The execution times are shown in Figure 5.1.

We find that on this scale, SAA Blendenpik, especially with SAS, is competitive relative to QR in terms of computational time. An SAS initialization improves the computing time as fewer iterations are needed. SAP with SAS is the computationally most efficient, and will in almost all cases converge to a solution with optimal residual

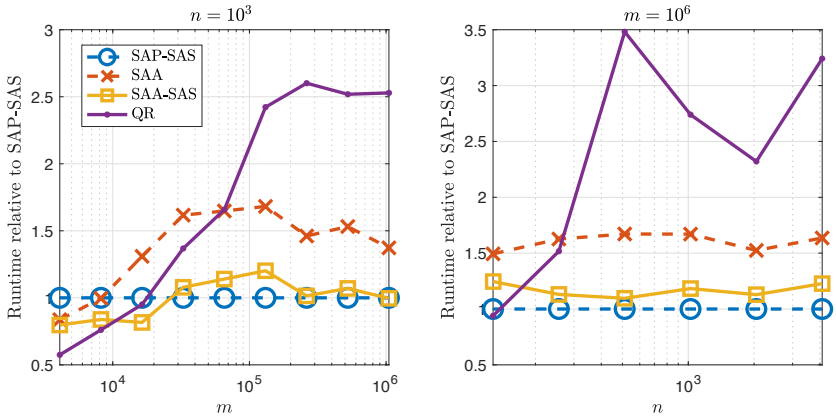

FIG. 5.1. *The relative execution time to solve large LS problems using SAP Blendenpik with SAS initialization (SAS), SAA Blendenpik, SAA Blendenpik with SAS, and Householder QR. For each problem* $\kappa(A) = 10^{10}$ *and the noise level* $\|e\|_2 = 10^{-10}$. *The tolerance is set to* 10^{-12} *and the maximum number of iterations to* 100 *(this is never reached). Left: Timings for* $n = 4000$ *and* $2^{12} \le m \le 2^{20}$. *Right: Timings for* $m = 10^6$ *and* $2^7 \le n \le 2^{12}$. *Note: color appears only in the online article.*

and backward error. For most practical applications, this should remain the preferred option.

We furthermore note that SAA (and SAP) algorithms could outperform the direct QR method by a larger factor if it is expensive to communicate with the matrix A. This context occurs, for example, when A is too large to be stored in a local cache and is instead stored on disk. Provided we obtain a good sketch of A, the number of iterations in LSQR before convergence will be modest, requiring limited streaming. The QR method, however, requires n views for an $m \times n$ matrix.

5.2. Numerical experiments with smoothed SAA Blendenpik. SAP Blendenpik with initialization, SAA Blendenpik, its smoothed version, and Householder QR can compute accurate solutions to LS problems, even when $\kappa_2(A) > 1/u$ (see Figure 5.2; the problems are generated as explained in subsection 1.5). Therefore, smoothing is only sometimes required for extremely ill-conditioned LS problems. However, even when smoothing is unnecessary, there is some potential benefit to smoothing because the perturbed LS problem can lead to rapid LSQR convergence. Of course, there is a tradeoff here as one converges rapidly to an accurate solution of the perturbed problem, and the computed solution may be less accurate for the original LS problem of interest.

However, there are also examples for which smoothing provides a more accurate solution to the original problem of $\min_x \|Ax - b\|$, not $\min_x \|\tilde{A}x - b\|$ (see Figure 5.3). For example, we take an LS problem involving a 1000×100 Kahan matrix with $\theta = 1.1$ from the MATLAB `gallery` collection and another one involving a column-scaled 1000×10 Vandermonde matrix involving equally spaced points between -1 and 1. The Kahan and Vandermonde matrices are designed so that their condition numbers are $> 1/u$. Without smoothing, even the QR-based algorithm fails to compute an accurate solution along with all the other methods. This is because the problem is so ill-conditioned that even a backward stable solution can behave wildly. However, after smoothing, SAA Blendenpik computes an accurate solution to the original LS problems. It should be noted that one could also smooth before solving the problems with QR to obtain more accurate solutions.

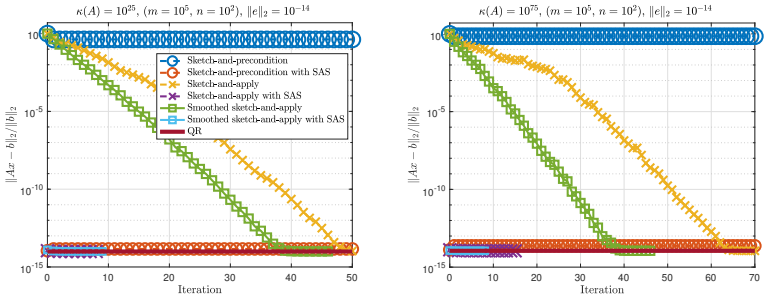

FIG. 5.2. *Smoothing is not always needed for LS problems with $\kappa_2(A) > 1/u$. These matrices are formed by drawing singular vectors from the Haar distribution and letting the singular values decay exponentially from 1 to $1/\kappa_2(A)$. For comparison, we show the residual error computed by Householder QR. These problems are so ill-conditioned that the backward error is not an informative measure, i.e., most computed solutions (e.g., with very few LSQR iterations) give a backward error $\mathcal{O}(u)$. Note: color appears only in the online article.*

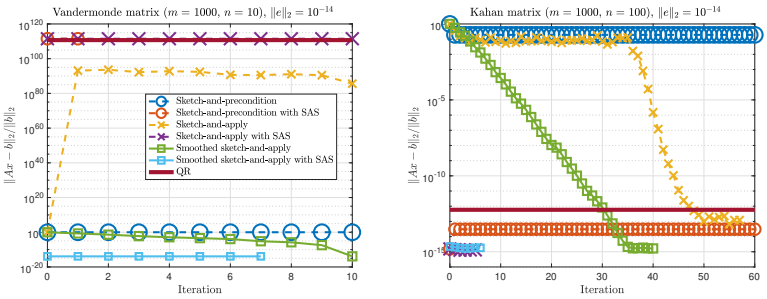

FIG. 5.3. *Smoothing can solve extremely ill-conditioned LS problems, even when SAA Blendenpik and Householder QR cannot. For the Vandermonde matrix, we can only obtain accurate solutions using a smoothed algorithm (note the y-axis). The problem with the Kahan matrix can only be solved optimally with certain SAA variations; the initialized SAP algorithm and standard SAA attain suboptimal residuals. Again, these problems are too ill-conditioned for the backward error to be informative. Note: color appears only in the online article.*

5.3. Sparse matrices. We briefly discuss the performance of the various algorithms in question on sparse matrices. Firstly, it should be noted that any SAA variant will not respect the sparsity: the matrix $Y = AR^{-1}$ is generally dense. As a result, convergence will be slower and computational cost will be unnecessarily high. Remarkably, Figure 5.4 shows that standard SAP can be numerically stable for sparse matrices with few nonzero entries. It appears that if $\mathrm{nnz}(A)$ is sufficiently small, the rounding errors compound less and the algorithm finds accurate solutions. A precise explanation is left for future work. Unsurprisingly, all algorithms with SAS initialization converge rapidly to accurate solutions.

6. Discussion. We have shown that SAP algorithms, such as Blendenpik [1], are numerically unstable in their standard form for solving LS problems. We have stabilized the algorithm by explicitly computing the preconditioned matrix AP and using an unpreconditioned iterative solver on AP. We coined this modification SAA. We furthermore displayed that using SAS initialization greatly improves convergence properties as well as the maximal attainable accuracy of SAP.

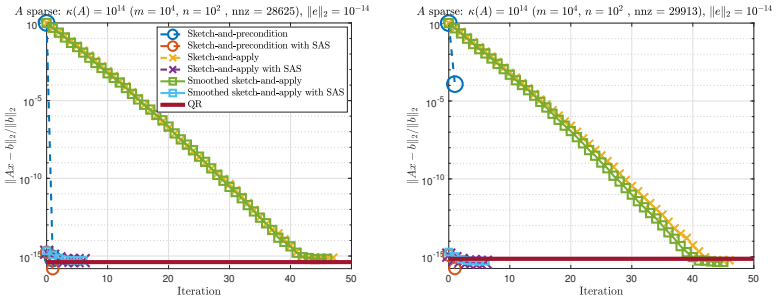

FIG. 5.4. *SAA algorithms cannot take advantage of sparsity of A. Note that SAP without SAS initialization can lead to accurate solutions for sparse matrices with few nonzero entries (left figure). This behavior depends nontrivially on the sparsity, etc.; in the right figure, we observe the numerical instabilities we have seen throughout this work. Note: color appears only in the online article.*

6.1. The effectiveness of SAS initialization. Although a large part of this work was dedicated to investigating the (provable) numerical stability of SAA and its smoothed version, one of the main messages—especially for practitioners—should be the remarkable effectiveness of the SAS initial guess. Apart from extremely ill-conditioned cases (see Figure 5.3), SAP with a SAS initial guess attains accurate solutions in terms of the residual, albeit it is not always backward stable. We urge practitioners to always choose Algorithm 1.2 over Algorithm 1.1, as the additional cost is minimal but it results in a better rate of convergence and better maximal attainable accuracy.

6.2. The numerical stability of data-driven preconditioners. The numerical instabilities observed in SAP Blendenpik raise larger questions on the stability of iterative methods with data-driven preconditioners for LS problems. Here, we refer to a data-driven preconditioner as a preconditioner constructed directly from A, without knowing where A came from (such as the discretization of a continuous problem). Figure 1.2 shows that even using R_A (where $A = Q_A R_A$) as a preconditioner—the perfect data-driven preconditioner—does not lead to a backward stable solution. We suspect this is due to the numerical errors incurred each time the ill-conditioned preconditioner is applied in an iterative solver. Is it possible for a data-driven preconditioner to avoid compounding these rounding errors?

6.3. The numerical stability of iterative LS solvers. As to the numerical stability of SAA, we were not able to state that the backward error $\|\Delta A\|_2$ is $\mathcal{O}(u\|A\|_2)$ (see Theorems 3.5 and 4.2). Instead, we have shown that $\|\Delta A\|_2/\|A\|_2$ is of the same order as the backward error $\|\Delta \hat{Y}\|$ incurred when $\hat{Y}z = b$ is solved with unpreconditioned LSQR, where $\hat{Y}$ is well-conditioned. It has proven challenging to understand the literature on the numerical stability of CG-like iterative solvers such as LSQR. Various works seem to strongly hint at backward stability under assumptions on the condition number, but use computational results to complement the claim; see [4, 13]. The numerical stability depends strongly on the specific implementation used in a way that we are yet to understand fully. We note that a recent result by Musco, Musco, and Sidford [26, Thm. 2.1], when specialized to well-conditioned positive definite linear systems, implies that ϵ forward error is achieved by using $\mathcal{O}(\log \frac{1}{\epsilon})$ bits, with Lanczos with modified Gram–Schmidt orthogonalization. For well-conditioned linear systems, taking $\epsilon = u$ this would imply the solution has an $\mathcal{O}(u)$ backward error, hence backward stable.

6.4. Variants of SAP. We have assumed specific choices for how the randomized preconditioner is constructed. In LSRN [24], for instance, the preconditioner is chosen to be $P = V\Sigma^{-1}$, where $SA = U\Sigma V^T$ is the SVD of the sketch. Numerical experiments show us that SVD-based SAP methods incur similar numerical instabilities as the QR-based variant. The SAA technique can also be used with LSRN ideas where the preconditioned matrix is computed as $Y = AV\Sigma^{-1}$. We suspect one can also prove similar results to our SAA Blendenpik analysis for SAA LSRN. Versions of Lemmas 3.1 and 3.2 hold almost identically for LSRN, with R replaced by ΣV. The key difference is the numerical errors ε_1 and ε_2, which we have not investigated. Since computing the SVD is typically more expensive than a QR factorization, we recommend SAA Blendenpik. One notable exception is when solving a sequence of Tikhonov regularized LS problems for various regularization parameters [23, 24].

6.5. Ill-conditioning caused by poor column scaling. If the ill-conditioning in an LS problem is caused by poor column scaling in A, we strongly recommend using QR-based SAA techniques. The reason is that computing AR^{-1} is invariant to column scaling, while the SVD-based computation is not. In fact, the assumption that $\kappa_2(A) \ll u^{-1}$ in section 3 can be replaced by

$$\min_D \{\kappa_2(AD) : D = \mathrm{diag}(b^{k_i}),\ k_i \in \mathbb{Z},\ i = 1, \ldots, n\} \ll u^{-1},$$

where b is the machine base (usually $b = 2$). Of course, one can also preprocess an LS problem by scaling the columns of A by powers of the machine base so that each column has a norm that is close to 1.

Acknowledgments. We thank Erin Carson and Zdeněk Strakoš for their valuable comments regarding the numerical stability of LSQR. We thank Françoise Tisseur for her input on Blendenpik in finite precision arithmetic. We are indebted to Ilse Ipsen and Michael Mahoney for their presentation and subsequent discussions on LS problems, which occurred during the "Complexity of Matrix Computations" seminar on 1st September 2021. We thank the referees and the editor for their valuable comments. We are especially indebted to the referee who suggested trying SAP with the SAS initial guess.

REFERENCES

[1] H. Avron, P. Maymounkov, and S. Toledo, *Blendenpik: Supercharging LAPACK's least-squares solver*, SIAM J. Sci. Comput., 32 (2010), pp. 1217–1236, https://doi.org/10.1137/090767911.
[2] Å. Björck, *Numerical Methods for Least Squares Problems*, SIAM, Philadelphia, 1996.
[3] Å. Björck, *The calculation of linear least squares problems*, Acta Numer., 13 (2004), pp. 1–53.
[4] Å. Björck, T. Elfving, and Z. Strakos, *Stability of conjugate gradient and Lanczos methods for linear least squares problems*, SIAM J. Matrix Anal. Appl., 19 (1998), pp. 720–736.
[5] C. Boutsidis and P. Drineas, *Random projections for the nonnegative least-squares problem*, Linear Algebra Appl., 431 (2009), pp. 760–771.
[6] P. Bürgisser and F. Cucker, *Smoothed analysis of Moore–Penrose inversion*, SIAM J. Matrix Anal. Appl., 31 (2010), pp. 2769–2783.
[7] C. Cartis, J. Fiala, and Z. Shao, *Hashing Embeddings of Optimal Dimension, with Applications to Linear Least Squares*, preprint, arXiv:2105.11815, 2021.
[8] X.-W. Chang, C. C. Paige, and D. Titley-Péloquin, *Stopping criteria for the iterative solution of linear least squares problems*, SIAM J. Matrix Anal. Appl., 31 (2009), pp. 831–852.
[9] K. L. Clarkson and D. P. Woodruff, *Low-rank approximation and regression in input sparsity time*, J. ACM, 63 (2017), pp. 1–45.
[10] K. R. Davidson and S. J. Szarek, *Local operator theory, random matrices and Banach spaces*, Handb. Geom. Banach Spaces, 1 (2001), 131.

[11] S. Gratton, P. Jiránek, and D. Titley-Peloquin, *On the accuracy of the Karlson–Waldén estimate of the backward error for linear least squares problems*, SIAM J. Matrix Anal. Appl., 33 (2012), pp. 822–836.
[12] A. Greenbaum, *Behavior of slightly perturbed Lanczos and conjugate-gradient recurrences*, Linear Algebra Appl., 113 (1989), pp. 7–63.
[13] A. Greenbaum, *Estimating the attainable accuracy of recursively computed residual methods*, SIAM J. Matrix Anal. Appl., 18 (1997), pp. 535–551.
[14] A. Greenbaum and Z. Strakos, *Predicting the behavior of finite precision Lanczos and conjugate gradient computations*, SIAM J. Matrix Anal. Appl., 13 (1992), pp. 121–137.
[15] N. Halko, P. G. Martinsson, and J. A. Tropp, *Finding structure with randomness: Probabilistic algorithms for constructing approximate matrix decompositions*, SIAM Rev., 53 (2011), pp. 217–288.
[16] E. Hallman, *Error Estimates for Least-Squares Problems*, Ph.D. thesis, UC Berkeley, Berkeley, CA, 2019.
[17] E. Hallman, *Estimating the backward error for the least-squares problem with multiple right-hand sides*, Linear Algebra Appl., 605 (2020), pp. 227–238.
[18] N. J. Higham, *Accuracy and Stability of Numerical Algorithms*, SIAM, Philadelphia, 2002.
[19] N. J. Higham and T. Mary, *A new approach to probabilistic rounding error analysis*, SIAM J. Sci. Comput., 41 (2019), pp. A2815–A2835.
[20] N. J. Higham and T. Mary, *Sharper probabilistic backward error analysis for basic linear algebra kernels with random data*, SIAM J. Sci. Comput., 42 (2020), pp. A3427–A3446.
[21] I. C. F. Ipsen and T. Wentworth, *The effect of coherence on sampling from matrices with orthonormal columns, and preconditioned least squares problems*, SIAM J. Matrix Anal. Appl., 35 (2014), pp. 1490–1520, https://doi.org/10.1137/120870748.
[22] P.-G. Martinsson and J. A. Tropp, *Randomized numerical linear algebra: Foundations and algorithms*, Acta Numer., 29 (2020), pp. 403–572.
[23] M. Meier and Y. Nakatsukasa, *Randomized Algorithms for Tikhonov Regularization in Linear Least Squares*, preprint, arXiv:2203.07329, 2022.
[24] X. Meng, M. A. Saunders, and M. W. Mahoney, *LSRN: A parallel iterative solver for strongly over- or underdetermined systems*, SIAM J. Sci. Comput., 36 (2014), pp. C95–118, https://doi.org/10.1137/120866580.
[25] G. Meurant and Z. Strakoš, *The Lanczos and conjugate gradient algorithms in finite precision arithmetic*, Acta Numer., 15 (2006), pp. 471–542.
[26] C. Musco, C. Musco, and A. Sidford, *Stability of the Lanczos method for matrix function approximation*, in Proceedings of the Twenty-Ninth Annual ACM-SIAM Symposium on Discrete Algorithms, SIAM, Philadelphia, 2018, pp. 1605–1624.
[27] C. C. Paige and M. A. Saunders, *LSQR: An algorithm for sparse linear equations and sparse least squares*, ACM Trans. Math. Software, 8 (1982), pp. 43–71.
[28] V. Rokhlin and M. Tygert, *A fast randomized algorithm for overdetermined linear least-squares regression*, Proc. Natl. Acad. Sci. USA, 105 (2008), pp. 13212–13217.
[29] G.-W. Stewart, *Stability of the Lanczos method for matrix function approximation*, in Research, Development, and LINPACK, Mathematical Software III, Academic Press, New York, 1977, pp. 1–14.
[30] D. Titley-Péloquin, *Backward Perturbation Analysis of Least Squares Problems*, Ph.D. thesis, McGill University, Montreal, Canada, 2010.
[31] J. A. Tropp, *Improved analysis of the subsampled randomized Hadamard transform*, Adv. Adapt. Data Anal., 3 (2011), pp. 115–126.
[32] J. A. Tropp, *Randomized block Krylov methods for approximating extreme eigenvalues*, Numer. Math., 150 (2022), pp. 217–255.
[33] R. Vershynin, *Introduction to the non-asymptotic analysis of random matrices*, in Compressed Sensing, Y. C. Eldar and G. Kutyniok, eds., Cambridge University Press, Cambridge, 2012, pp. 210–268.
[34] B. Waldén, R. Karlson, and J.-G. Sun, *Optimal backward perturbation bounds for the linear least squares problem*, Numer. Linear Algebra Appl., 2 (1995), pp. 271–286.
[35] A. J. Wathen and T. Rees, *Chebyshev semi-iteration in preconditioning for problems including the mass matrix*, Electron. Trans. Numer. Anal., 34 (2008), pp. 125–135.
[36] D. P. Woodruff, *Sketching as a tool for numerical linear algebra*, Found. Trends Theor. Comput. Sci., 10 (2014), pp. 1–157.
[37] Y. Yamamoto, Y. Nakatsukasa, Y. Yanagisawa, and T. Fukaya, *Roundoff error analysis of the Cholesky QR2 algorithm*, Electron. Trans. Numer. Anal., 44 (2015), pp. 306–326.