The Dual JL Transforms and Superfast Matrix Algorithms

Qi Luan[1],[a] and Victor Y. Pan[1,2],[b]
[1] Ph.D. Programs in Computer Science and Mathematics
The Graduate Center of the City University of New York
New York, NY 10036 USA
[2] Department of Computer Science
Lehman College of the City University of New York
Bronx, NY 10468 USA
[a] qi_luan@yahoo.com
[b] victor.pan@lehman.cuny.edu
http://comet.lehman.cuny.edu/vpan/

Abstract

We call a matrix algorithm superfast (aka running at sublinear cost) if it involves much fewer flops and memory cells than the matrix has entries. Using such algorithms is highly desired or even imperative in computations for Big Data, which involve immense matrices and are quite typically reduced to solving linear least squares problem and/or computation of low rank approximation of an input matrix. The known algorithms for these problems are not superfast, but we prove that their certain superfast modifications output reasonable or even nearly optimal solutions for large input classes. We also propose, analyze, and test a novel superfast algorithm for iterative refinement of any crude but sufficiently close low rank approximation of a matrix. The results of our numerical tests are in good accordance with our formal study.

Keywords: Superfast (sublinear cost) algorithms, Linear least squares problem, Low rank approximation, Johnson and Lindenstrauss lemma, Iterative refinement

2020 Math. Subject Classification: 65Y20, 65F05, 65F55, 68Q25, 68W20, 68W25

1 Introduction to dual superfast matrix algorithms

We study Least Squares Solution of a Highly Overdetermined Linear System of Equations, aka Linear Least Squares Problem *(LLSP)* for short,[1] and Low Rank Approximation of a matrix *(LRA)*.[2] Both problems are among the most fundamental in Numerical Linear Algebra and Data Mining and Analysis, with applications ranging from PDEs to machine learning theory, neural networks, term document data, and DNA SNP data. See the classical books [36] and [4] for LLSP and more recent coverage of both LLSP and LRA in the book [69] and survey [41]; also see [18], [61], [7], [3],

[1] Some authors call this problem differently, e.g., *Linear Least Squares Approximation Problem* [41] or *Linear Regression* [69].

[2] In the study of LRA it is customary to rely on an informal basic concept of low rank; we also use other informal concepts such as "large", "small", "ill-" and "well-conditioned", "near", "close", and "approximate", quantified in context, in our high level presentation but we complement this description with formal presentation and analysis.

[20], [40], and the bibliography therein for LLSP and [39], [32], [14], [67], [46], [68], [35], [42], [6], [70], and the bibliography therein for LRA.

An algorithm runs *superfast* (aka *at sublinear cost*) if for an $m \times n$ input matrix it involves $o(mn)$ flops,[3] thus accessing only a small fraction, $o(mn)$ of all its mn entries – at most two entries per flop. Can such an algorithm output a meaningful solution to LLSP and/or LRA? No, it cannot for worst case inputs, for which the output error of any superfast algorithm are arbitrarily large, but one cannot accept this "No" for an answer in the study of Big Data (e.g., unfolding matrices of multidimensional tensors): quite typically they are so immense that realistically one can access only a tiny fraction of their entries and must perform computations superfast.

Empirically one can solve LRA by means of applying Cross–Approximation *C-A* iterations[4] (aka Adaptive Cross–Approximation) iterations (see [65], [30], [31], [29], [66], [5], [26], [9], [8], [25], [27], [46], [45]), which can be viewed as specialization of ADI celebrated techniques to LRA, but the power of C-A iterations relies on their surprisingly fast and consistent global convergence, right from the start, whose formal support has long remained overdue; a very limited recent support in [38] should motivate further effort.

Dramatic progress in the LLSP-LRA area was based on the Johnson and Lindenstrauss *(JL)* transform of an input matrix into a matrix of a much smaller size, for which optimal solution is computed superfast and is still nearly optimal with a high probability *(whp)* for the original task of a much larger size (cf. [33], [17], [41]). Overall cost of the solution, however, remains superlinear, dominated by the cost of the JL transform.

In this paper we explore the following *dual* way to circumventing the conundrum: in addition to the algorithms that at superlinear cost compute solution for ANY input matrix whp we devise, analyze, and test superfast ones, which solve the same problem for MANY input matrices, namely, for their large and important classes.

Towards our goal we first introduce a natural probabilistic structure in the space of input matrices, then trivialize the known JL transforms to run them superfast, and finally prove that for a random input the resulting algorithms whp output nearly optimal solution to LLSP as well as meaningful LRA. Thus our superfast algorithms dramatically accelerate solution for a large class of inputs based on dual JL transform.

One would prefer to compute solution superfast for all inputs, but

(a) as we said, the output errors of any superfast algorithm can be arbitrarily large for both LLSP and LRA and

(b) the user should be satisfied even if the problem is solved just for the input class of her/his interest.

Furthermore we can strengthen the chances for success of our superfast algorithms by reapplying them under modification of their parameters and/or superfast pre-processing of an input matrix. In this way we narrow the need for application of the known solution algorithms that run at superlinear cost, and this gain is significant for LRA according to Corollary 10.3 and Remark C.1 and for LLSP according to our extensive tests with both synthetic and real world inputs.

We have almost instantly arrived at formal support of our superfast dual algorithm for LLSP, but technical challenge was significantly greater in the case of LRA, and our progress was restricted to input matrices admitting sufficiently close LRA. Our effort, however, has been rewarded when with more work we devised an algorithm for superfast iterative refinement of any crude but reasonably close LRAs, say, those output by our superfast algorithms, C-A iterations, or another known algorithm, which must sacrifice so called "relative error" property of output accuracy – within a

[3] "Flop" stands for "floating point arithmetic operation".

[4] This concept has been coined by E.E. Tyrtyshnikov in [66]).

factor of $1+\epsilon$ from the optimal – in order to ensure their numerical stability [23, Thm. 3.4]. We analyz, extensively test this refinement algorithm, and in Part III propose to enhance its power by combining it with C-A iterations.

We hope that our progress will motivate further effort towards the design, analysis, and implementation of superfast algorithms for LLSP, LRA, and possibly other challenges in matrix computation (cf. Part III of our paper).

Some related works (also see the end of Section 6). Our dual JL transforms extend pioneering dual matrix algorithms devised, analyzed, and tested in [54], [57], [56], [55], [59], [60], [49], [52], and [53] for LRA and other fundamental matrix computations such as Gaussian elimination with no pivoting. In Part II we compare our study with an alternative JL-like approaches to LRA in [52] and [53], which had preliminary versions in the reports [50], [51], and [47]. The report [49] has been devoted to a distinct and more primitive approach to superfast iterative refinement of LRA – based on recursive application of a superfast Subalgorithm for LRA given as a part of an input, whereas for our present algorithm for iterative refinement of LRA we do not assume that such a basic Subalgorithm is available.

Organization of the paper. In Parts I and II we recall some related known works, specify LLSP and LRA, and present and analyze our algorithms in some detail. The main result of Part I is very simple, but introduces our duality approach, and is well supported with our extensive tests. We devote short Part III to conclusions.

PART I. Superfast Approximate Least Squares Solution of a Highly Overdetermined Linear System of Equations (LLSP)

2 Introduction to Superfast LLSP

Our progress in LLSP. In the beginning of Section 4 we demonstrate that the output of any superfast algorithm fails miserably for the solution to LLSP problem on the worst case inputs, but next we circumvent the problem by applying a dual variation of the Sarlòs randomized algorithm of [63].

That algorithm implements the JL transform by means of multiplying an input matrix of LLSP by a random rectangular multiplier, which dramatically decreases the input size of LLSP. Sarlòs proves that the solution of the resulting LLSP is also a nearly optimal solution of the original LLSP of a much larger size whp in the case of proper choices of a multiplier, in particular in the case of a Gaussian random multiplier, that is, a matrix filled with independent identically distributed *i.i.d.* Gaussian (aka normal) random variables. Hereafter we call such a matrix just *Gaussian*.

In his algorithm the original input matrix can be assumed to be orthogonal[5] without loss of generality [63], and then its product with a Gaussian multiplier of a smaller size is Gaussian, by virtue of the well-known orthogonal invariance property. Now we observe that his result still holds for the solution of the *dual LLSP*, with a Gaussian input matrix and any orthogonal multiplier (see our Theorem 4.1).

Is this observation of any interest? Yes, because by choosing sparse orthogonal multipliers we arrive at *superfast algorithms* that compute nearly optimal solutions to LLSP for a large class of inputs since they do so for a random input whp.

Organization of the rest of PART I. In the next section we recall the LLSP and its randomized approximate solution of [63]. We present its superfast variation in Section 4. In

[5] A real $m \times n$ matrix M is *orthogonal* or *orthonormal* if $M^TM = I_n$ or $MM^T = I_m$ for M^T denoting the transpose of M and I_s denoting the $s \times s$ identity matrix.

Section 5 we cover our numerical tests.

3 Randomized Approximate Solution of an Overdetermined Linear System of Equations

Problem 3.1. [Least Squares Solution of an Overdetermined Linear System of Equations (LLSP).] *Given two integers m and d such that $1 \le d < m$, a matrix $A \in \mathbb{R}^{m\times d}$, and a vector $\mathbf{b} \in \mathbb{C}^m$, compute a vector $\mathbf{x} \in \mathbb{R}^d$ that minimizes the spectral norm $||A\mathbf{x} - \mathbf{b}||$ or equivalently computes the subvector $\mathbf{x} = (y_j)_{j=0}^{d-1}$ of the vector*

$$\mathbf{y} = (y_j)_{j=0}^{d} = \text{argmin}_{\mathbf{v}}\ ||M\mathbf{v}|| \text{ such that } M = (A \mid \mathbf{b}) \text{ and } \mathbf{v} = \begin{pmatrix} \mathbf{x} \\ -1 \end{pmatrix}. \tag{3.1}$$

The minimum norm solution to this problem is given by the vector $\mathbf{x} = A^+\mathbf{b}$ for A^+ denoting the Moore–Penrose pseudo inverse of A; $A^+\mathbf{b} = (A^*A)^{-1}A^*\mathbf{b}$ if a matrix A has full rank d.

Algorithm 3.1. *(*Randomized Approximate Solution of LLSP from [63].*)*

INPUT: *An $m \times (d+1)$ matrix M.*

OUTPUT: *A vector $\mathbf{x} \in \mathbb{R}^d$ approximating a solution of Problem 3.1.*

INITIALIZATION: *Fix an integer s such that $d \le s \ll m$.*

COMPUTATIONS: *1. Generate a matrix $F \in \mathbb{R}^{s\times m}$.*

2. Compute a solution $\mathbf{x}$ of Problem 3.1 for the $s \times (d+1)$ matrix FM.

Clearly, the transition to an input matrix FM simplifies Problem 3.1 because its size decreases, and the simplification is dramatic where $s \ll m$, while the following theorem shows that the algorithm still outputs nearly optimal approximate solution to Problem 3.1 for M whp if $\sqrt{s}\ F$ is in the linear space of $s \times m$ Gaussian matrices.[6]

Theorem 3.1. (Error Bound for Algorithm 3.1. See [63] or [69, Theorem 2.3].) *Let us be given two integers s and d such that $0 < d \le s$, a real $m \times (d+1)$ matrix $M \in \mathbb{R}^{m\times(d+1)}$, an $s \times m$ Gaussian matrix F, and two tolerance values γ and ϵ such that*

$$0 < \gamma < 1,\ 0 < \epsilon < 1,\ \text{and } s = \left(d + \log\left(\frac{1}{\gamma}\right)\right)\ \frac{\eta}{\epsilon^2} \tag{3.2}$$

for a constant η. Then

$$\text{Probability}\left\{1 - \epsilon \le \frac{1}{\sqrt{s}}\ \frac{||FM\mathbf{y}||}{||M\mathbf{y}||} \le 1 + \epsilon \text{ for all vectors } \mathbf{y} \ne \mathbf{0}\right\} \ge 1 - \gamma. \tag{3.3}$$

The computation of the matrix FM involves order of $dsm \ge d^2m$ flops; for $m \gg s$ this dominates the overall arithmetic computational cost of the solution of Problem 3.1.

The current record upper estimate for this cost is $O(d^2m)$, while the record lower bound has order $(m+d)s\epsilon^{-1}\log(md)$ provided that the relative output error norm is within a factor of $1+\epsilon$ from its minimal value (see [69, Section 2.1]).

[6] Such an approximate solution serves as a pre-processor for practical implementation of numerical linear algebra algorithms for Problem 3.1 of least squares computation [41, Section 4.5], [3].

4 Superfast Dual LLSP

Any superfast algorithm for LLSP misses an input entry $m_{i,j}$ for some pair i and j; therefore for its output the norm $||M\mathbf{y}||$ exceeds the norm $||M\mathbf{x}_{\text{optimal}}||$ by an arbitrarily large factor for the worst case input M. Indeed modification of $m_{i,j}$ does not change the output of such an algorithm but can dramatically change an optimal solution to the LLSP.[7] The argument can be immediately extended to randomized algorithms, and so any superfast deterministic or randomized algorithm fails miserably on some inputs. These observations should make the following simple extension of Theorem 3.1 quite interesting.

Theorem 4.1. [Error Bounds for Dual LLSP.] *Suppose that we are given three integers s, m, and d such that $0 < d \le s < m$, and two tolerance values γ and ϵ satisfying (3.2). Define an orthogonal matrix $Q_{s,m} \in \mathbb{R}^{s\times m}$ and an $m \times (d+1)$ Gaussian matrix $G_{m,d+1}$. Write*

$$F := a\ Q_{s,m} \text{ and } M := b\ G_{m,d+1} \tag{4.1}$$

for two scalars a and b such that $ab\sqrt{s} = 1$. Then

$$\text{Probability}\Big\{1-\epsilon\ \le \frac{||FM\mathbf{z}||}{||M\mathbf{z}||} \le 1+\epsilon \text{ for all vectors } \mathbf{z} \ne \mathbf{0}\Big\} \ge 1-\gamma.$$

Proof. The claim follows from Theorem 3.1 because the $s \times (d+1)$ matrix $\frac{1}{ab}FM$ is Gaussian by virtue of orthogonal invariance of Gaussian matrices. □

The theorem shows that for a Gaussian matrix M and any properly scaled orthogonal matrix F of (4.1), Algorithm 3.1 outputs a solution of Problem 3.1 that whp is optimal up to a factor lying in the range $[1-\epsilon, 1+\epsilon]$, and clearly, these computations become superfast for a proper choice of a sparse multiplier F.

5 Numerical Tests for LLSP

In this section we present the results of our tests of superfast dual version of Algorithm 3.1 for both synthetic inputs from [3] and real-world data. We worked with random orthogonal multipliers, let $\mathbf{x} := \arg\min_{\mathbf{u}} ||FA\mathbf{u} - F\mathbf{b}||$, and computed the relative residual norm

$$\frac{||A\mathbf{x}-\mathbf{b}||}{\min_{\mathbf{u}} ||A\mathbf{u}-\mathbf{b}||}.$$

In our tests these ratios quite closely approximated 1 from above.

We used the following random scaled orthogonal multipliers $F \in \mathbb{R}^{s\times m}$:

(i) full rank $s \times m$ submatrices of $m \times m$ random permutation matrices,

(ii) ASPH matrices of our Appendix A, recalled from [52] and [53]. They are output after performing just the first three recursive steps out of $\log_2 m$ steps involved into the generation of the matrices of subsampled randomized Hadamard thansform, and

(iii) block permutation matrices formed by filling $s\times m$ matrices with $c = m/s$ identity matrices, each of size $s \times s$, and by performing random column permutations; we have chosen $c = 8$ to match the computational cost of the application of ASPH multipliers.

We also included the test results with $s \times m$ Gaussian multipliers, for comparison.

We performed our tests on a machine with Intel Core i7 processor running Windows 7 64bit; we invoked the *lstsq* function from Numpy 1.14.3 for solving the LLSPs.

[7] Here we assume that $y_j \ne 0$. Otherwise we could delete the jth column of M, thus decreasing the input size.

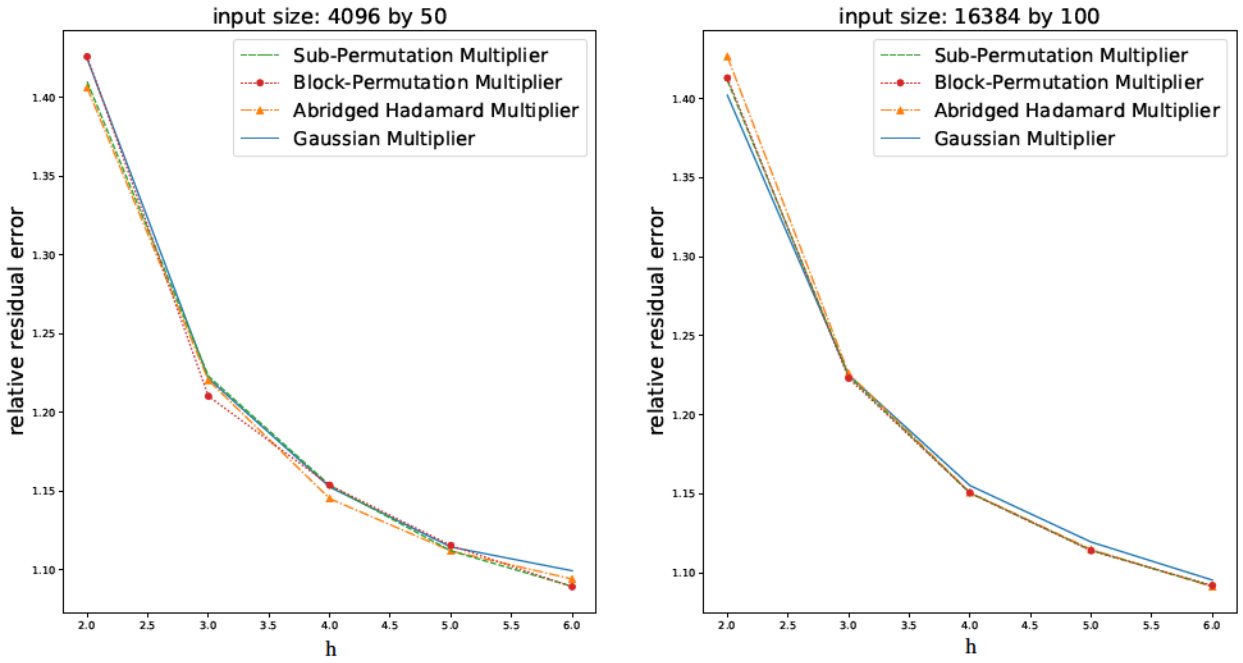

Figure 1: Relative residual norm in tests with Gaussian inputs

5.1 Synthetic Input Matrices

For synthetic inputs, we generated input matrices $A \in \mathbb{R}^{m \times d}$ by following (with a few modifications) the recipes of extensive tests in [3], which compared the running time of the regular LLSP problems and the reduced ones with WHT, DCT, and DHT pre-processing.

We used input matrices A of the sizes 4096×50 and 16834×100 being either Gaussian matrices or random ill-conditioned matrices. We generated the input vectors $\mathbf{b} = \frac{1}{||A\mathbf{w}||} A\mathbf{w} + \frac{0.001}{||\mathbf{v}||}\mathbf{v}$, where $\mathbf{w}$ and $\mathbf{v}$ were random Gaussian vectors of size d and m, respectively, and so $\mathbf{b}$ was in the range of A up to a smaller term $\frac{0.001}{||\mathbf{v}||}\mathbf{v}$.

Figure 1 displays the test results for Gaussian input matrices.

Figure 2 displays the test results for ill-conditioned random inputs defined by their SVD $A = U\Sigma V^*$ where the orthogonal matrices U and V of singular vectors were given by the Q factors in the thin QR-factorization of two independent Gaussian matrices and where $\Sigma = \mathrm{diag}(\sigma_j)_j$ with $\sigma_j = 10^{5-j}$ for $j = 1, 2 \ldots, 14$ and $\sigma_j = 10^{-10}$ for $j > 14$.

Our input matrices A were highly over-determined, having many more rows than columns. We have chosen $s = dh$, $h = 2, 3, 4, 5, 6$ for the multipliers F. By decreasing the integer s and the ratio $h = s/d$ we could have accelerated our algorithm, but we had to keep them large enough in order to yield accurate solution.

We performed 100 tests with 100 random multipliers for every triple of the input class, multiplier class, and test sizes (cf. (3.2)) and computed the mean of the 100 relative residual norms of the outputs.

We display the test results in Figures 1 and 2 with ratio h marked on the horizontal axis. The tests show that our multipliers were consistently effective for random matrices. The performance was not affected by the conditioning of input matrices.

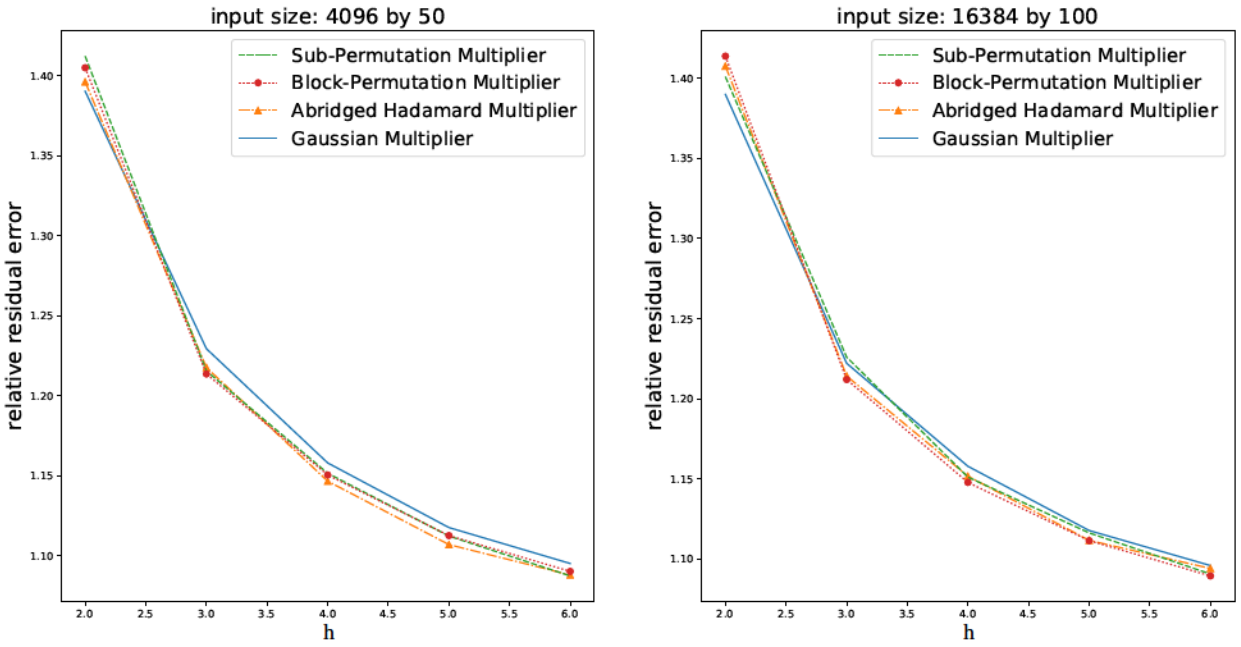

Figure 2: Relative residual norm in tests with ill-conditioned random inputs

5.2 Red Wine Quality Data and California Housing Prices Data

In this subsection we present the test results for some real world inputs, namely the Red Wine Quality Data and California Housing Prices Data. For each triple of the datasets, multiplier type, and multiplier size, we repeated the test for 100 random multipliers and computed the mean relative residual norm. The results for these two input classes are displayed in Figures 3 and 4.

11 physiochemical feature data of the Red Wine Quality Data such as *fixed acidity*, *residual sugar level*, and *pH level* were the input variables in our tests and one sensory data *wine quality* were the output data; the tests covered 1599 variants of the Portuguese wine "Vinho Verde". See further information in [12]. This data set is often applied in regression tests that use physiochemical data of a specific wine in order to predict its quality, and among various types of regression LLSP algorithms are considered a popular choice.

From this data set we constructed a 2048×12 input matrix A with each row representing one variant of red wine, and with columns consisting of a bias column and eleven physiochemical feature columns. The input vector $\mathbf{b}$ was a vector consisting of the wine quality level (between 0 and 10) for each variant. We kept the 1599 rows of the original data, padded the rest of the rows with zeros, and performed a full row permutation of A.

The California Housing Prices data appeared in [48] and were collected from the 1990 California Census, including 9 attributes for each of the 20,640 Block Groups observed. This data set is used for regression tests in order to predict the *median housing value* of a certain area given collected information of this area, such as *population*, *median income*, and *housing median age*.

We randomly selected 16,384 observations from the data set in order to construct an independent input matrix A_0 of size 16384×8 and a dependent input vector $\mathbf{b} \in \mathbb{R}^{16384}$. Furthermore, we augmented A_0 with a single bias column, i.e. $A = \begin{bmatrix} A_0 & \mathbf{1} \end{bmatrix}$.

We computed approximate solutions by applying the algorithm supporting Theorem 4.1 and by using our multipliers. Figure 3 and 4 show that the resulting solution was almost as accurate as

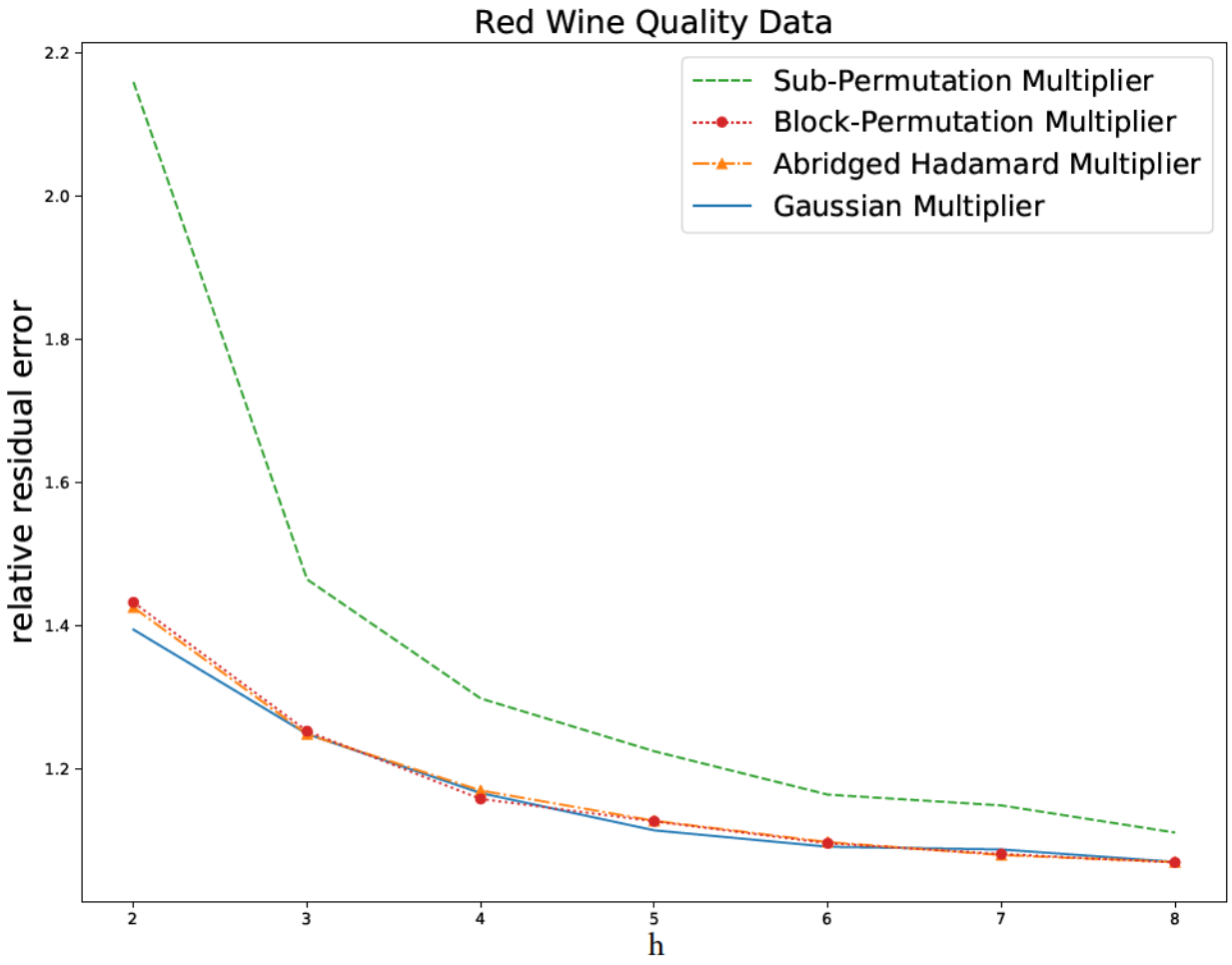

Figure 3: Relative residual norm in tests with Red Wine Quality Data

the optimal solution. Moreover, using Gaussian multipliers rather than our sparse multipliers only enabled a marginal decrease of relative residual norm.

PART II. Superfast Low Rank Approximation Directed by Leverage Scores

6 Introduction to superfast computation of LRA

Every superfast LRA algorithm fails miserably on the input matrices of the small families of Appendix B, but similarly to the case of LLSP we do not stop at that point. In [52] and [53] the authors extend our superfast solution of LLSP of Part I to superfast LRA whp.[8]

Namely, in the known randomized LRA of a matrix M by means of subspace projection (cf., e.g., [32] or [67]) a JL transform is performed by means of computing the matrix FM or MH for a random rectangular multiplier F or H (called a *test matrix*) and then LLSP is solved for an input

[8] In [52] extension relies on solving *generalized LLSP*, where a matrix B substitutes for a vector $\mathbf{b}$ on the right-hand side. The Sarlós solution and our variation of it can be readily extended.

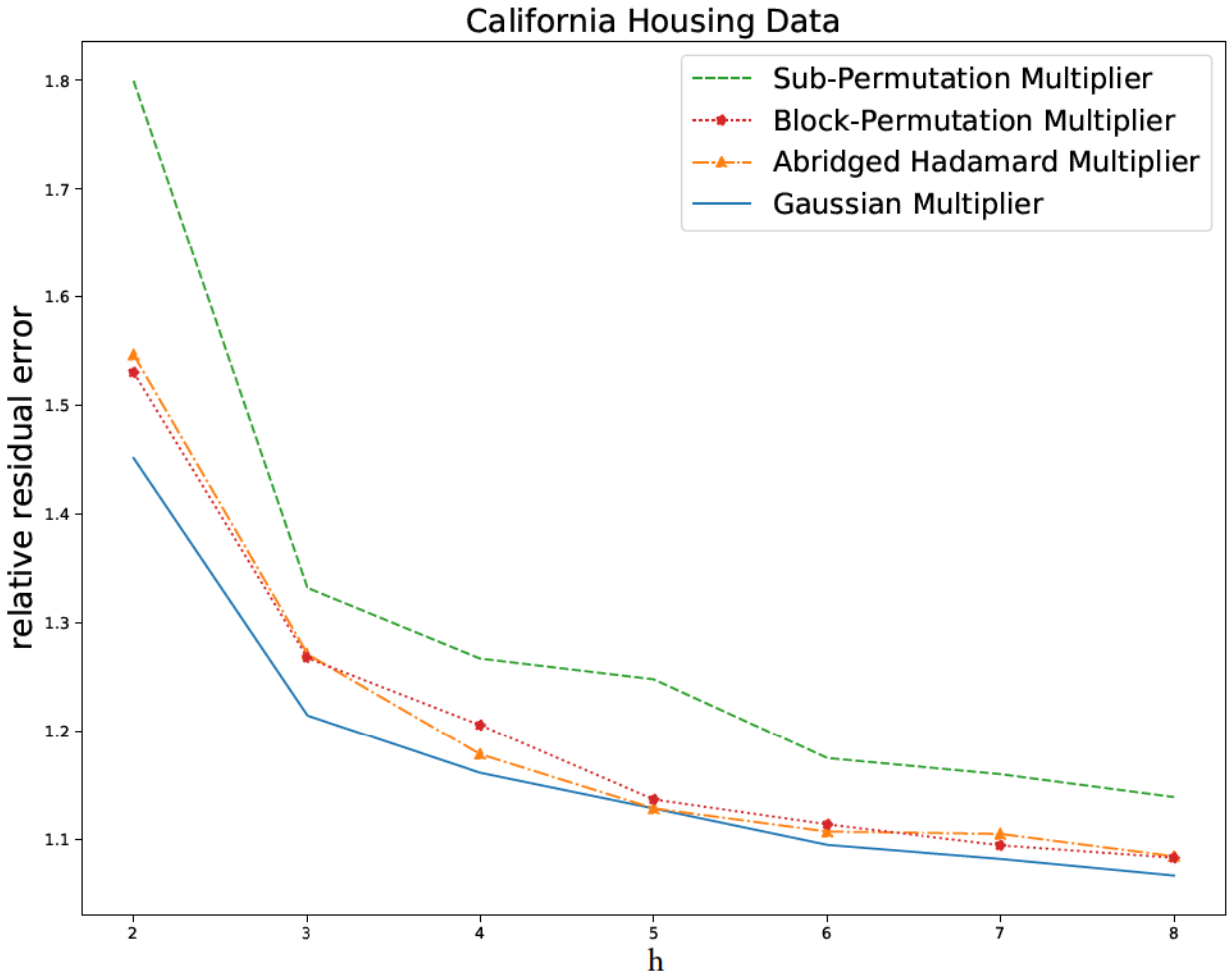

Figure 4: Relative residual norm in tests with California Housing Prices Data

matrix FM or MH of a smaller size. As in the case of LLSP, it has been proved that for Gaussian, SRHT, SRFT, and some other random test matrices the output LRA is close to optimal whp, and that the computation can be performed superfast except for the stage of computing the product FM or MH. By choosing a proper sparse test matrix F or H, [52] and [53] run the entire LRA algorithm superfast.

In both papers the authors define a probabilistic structure in the input space and then prove that their output LRAs are reasonably close whp, although are not optimal. Furthermore in [49] the authors propose, analyze, and test a superfast extension to LRA of the classical algorithm for iterative refinement of the solution of a matrix equation, which is based on recursive application of the superfast algorithm of [52] as a basic Subalgorithm given as a part of an input.

Next we follow this pattern but use a different framework. Instead of applying subspace projection, we rely on the random sampling algorithm of [19], which performs a JL transform by means of sampling relatively small numbers of rows and columns of an input matrix. This sampling is directed by sampling probabilities, called *leverage scores*; the algorithm outputs nearly optimal LRA whp for proper leverage scores and apart from the stage of their computation runs superfast.

We arrive at *superfast algorithms for dual LRA* by trivializing that stage and then prove that they still output reasonably close LRA of matrices of a large class. This follows from our stronger result that whp these algorithms output close LRA of a random input matrix provided that it admits sufficiently close LRA, that is, lies near a matrix of low rank within a sufficiently small distance specified in Theorem 8.3 and Remark 8.6 and estimated empirically in Section 11.2.

Actually, whp our superfast algorithms compute a close LRA of any matrix admitting sufficiently close LRA and pre-processed with Gaussian multipliers (see Theorem C.6). Of course, we cannot perform such pre-processing superfast, for otherwise we would have arrived at superfast LRA for a worst case input, but empirically superfast pre-processing with various sparse orthogonal multipliers works as efficiently (see Remark C.1).

[52] and [53] obtain similar results under the same randomization model, but rely on distinct and more primitive choices of test matrices and arrive at weaker LRAs. In particular in our present paper and both [52] and [53] the upper estimates for the output errors grow as an input matrix deviates from a low rank matrix, but this growth is the slowest in our case because leverage scores are not too much affected by the perturbation of an input matrix (see Section 11.2).

Our most surprising and *practically promising novelty* of Part II is a novel algorithm for superfast refinement of a crude but reasonably close LRA. We first observe that one can superfast compute leverage scores of a low rank matrix, and in particular of LRA of a matrix. That observation motivated our work on extension of the algorithms of [19] to superfast refinement of ANY crude but reasonably close LRA of a matrix. This natural challenge turned out to be technically demanding, but we succeeded based on estimating the angles between subspaces associated with singular vectors and computed recursively in our refinement process.

As in [19], our algorithms output CUR LRA, which is a particularly memory efficient form of LRA, traced back to [30], [29], and [31]. As in [19], our formal support of the proposed algorithm requires sampling a fairly large numbers of rows and columns of an input matrix. In numerical tests with real world data reported in [19], however, the algorithms of that paper succeeded with quite reasonable numbers of row and column samples, and similarly in our tests presented in Section 11.1, our superfast refinement algorithm improved an initial LRA by sampling only a small number of rows or columns at each iteration and output a close-to-optimal solution in a few iterations.

Related works: The papers [19] and [34] were the points of departure for our study of LRA and its refinement, respectively. The first formal support for dual superfast LRA is due to the papers [50], [51], and [47].

Organization of the rest of Part II. We devote the next section to background for LRA.

In Section 8 we recall subspace sampling algorithms of [19], directed by leverage scores. In Section 9 we cover randomized iterative refinement of a crude but sufficiently close LRA. In Section 10 we prove that our superfast variation of the algorithms of [19] is accurate whp for a random input. In Section 11.1 we present the results of our tests of randomized iterative refinement of Section 9. In Section 11.2, the contribution of the third author, we cover our tests of the perturbations of leverage scores caused by the perturbation of some real world inputs. In addition to the background material of the next section, we recall such material also in the Appendix, in particular some background on random matrices in Appendix C and the auxiliary algorithms of [19] for random sampling and re-scaling in Appendix D.

7 Background for LRA

7.1 Matrix norms, pseudo inverse, and SVD

For simplicity we assume dealing with real matrices in $\mathbb{R}^{p\times q}$ throughout. *r-top SVD* of a matrix M of rank at least r is the decomposition $M_r = U^{(r)}\Sigma^{(r)}V^{(r)T}$ for the diagonal matrix $\Sigma^{(r)} = \mathrm{diag}(\sigma_j)_{j=1}^{r}$ of the r largest singular values of M and two orthogonal matrices $U^{(r)}$ and $V^{(r)}$ of the associated top left and right singular spaces, respectively. M_r is said to be the *r-truncation* of M.

$M_r = M$ for a matrix M of rank r, and then its r-top SVD is just its *compact SVD*

$$M = U_M\Sigma_M V_M^T, \text{ for } U_M = U^{(r)},\ \Sigma_M = \Sigma^{(r)}, \text{ and } V_M = V^{(r)}.$$

$M^+ := V_M\Sigma_M^{-1}U_M^T$ is the Moore–Penrose pseudo inverse of M.

Hereafter $||\cdot||$ denotes the spectral norm, as in Part I, $||\cdot||_F$ denotes the Frobenius norm, and by following [32] we use the unified notation $|||\cdot|||$ for both of these matrix norms.

Lemma 7.1. [The norm of the pseudo inverse of a matrix product.] *Suppose that $A \in \mathbb{R}^{k\times r}$, $B \in \mathbb{R}^{r\times l}$, and the matrices A and B have full rank $r \le \min\{k,l\}$. Then $|||(AB)^+||| \le |||A^+|||\ |||B^+|||$.*

7.2 2-factor LRA

A matrix M has ϵ-rank at most r if it admits approximation within an error norm ϵ by a matrix M' of rank at most r or equivalently if there exist three matrices A, B and E such that

$$M = M' + E \text{ where } |||E||| \le \epsilon\ |||M|||,\ M' = AB,\ A \in \mathbb{R}^{m\times r}, \text{ and } B \in \mathbb{R}^{r\times n}. \tag{7.1}$$

ϵ-rank ρ of a matrix M is numerically unstable if ρth and $(\rho+1)$st or ρth and $(\rho-1)$st largest singular values of M are close to one another, but it is quite common to define *numerical rank*, $\mathrm{nrank}(M)$, of a matrix M as its ϵ-rank for a tolerance ϵ fixed in context, e.g., depending on computer precision, an input class and output requirement (cf. [24]).

A matrix admits its close approximation by a matrix of rank at most r if and only if it has numerical rank at most r.

Theorem 7.1. [24, Theorem 2.4.8].) *Write $\tau_{r+1}(M) := \min_{N:\ \mathrm{rank}(N)=r} |||M-N|||$. Then $\tau_{r+1}(M) = |||M - M_r|||$ under both spectral and Frobenius norms: $\tau_{r+1}(M) = \sigma_{r+1}(M)$ under the spectral norm and $\tau_{r+1}(M) = \sigma_{F,r+1}(M) := \sqrt{\sum_{j>r}\sigma_j^2(M)}$ under the Frobenius norm.*

7.3 Canonical CUR LRA and 3-factor LRA

For two sets $\mathcal{I} \subseteq \{1, \dots, m\}$ and $\mathcal{J} \subseteq \{1, \dots, n\}$ define the submatrices

$$M_{\mathcal{I},:} := (m_{i,j})_{i\in\mathcal{I};j=1,\dots,n}, M_{:,\mathcal{J}} := (m_{i,j})_{i=1,\dots,m;j\in\mathcal{J}}, \text{ and } M_{\mathcal{I},\mathcal{J}} := (m_{i,j})_{i\in\mathcal{I};j\in\mathcal{J}}.$$

Given an $m \times n$ matrix M of rank r and its nonsingular $r \times r$ submatrix $G = M_{\mathcal{I},\mathcal{J}}$ one can readily verify that $M = M'$ for

$$M' = CUR,\ C = M_{:,\mathcal{J}},\ U = G^{-1},\ \text{and } R = M_{\mathcal{I},:}. \tag{7.2}$$

We call the matrices G and U the *generator* and *nucleus* of *CUR decomposition* of M, respectively. [9]

In the case of a matrix M of numerical rank r (7.2) defines its *canonical CUR approximation* M' of rank r as long as the CUR generator G is nonsinguar, although this approximation M' can be arbitrarily poor in the case of ill-conditioned generator G.

Generalize canonical CUR LRA by allowing to use $k \times l$ CUR generators G of (7.3) for k and l satisfying

$$r \le k \le m,\ r \le l \le n \tag{7.3}$$

and for the nucleus defined by the r-truncation of G as follows:

$$U := G_r^+,\ ||U|| = 1/\sigma_r(G).$$

Hereafter we follow [19] [14], [46] by studying such a canonical CUR LRA, for which the computation of a nucleus involves kl memory cells and $O(kl \min\{k, l\})$ flops.

Remark 7.1. *In a more general definition of CUR LRA one fixes a pair of matrices C and R made up of two sets of columns and rows of M and chooses any $l \times k$ nucleus U for which the error matrix $E = CUR - M$ has a smaller norm. In particular the Frobenius error norm is minimized for the nucleus $U = C^+MR^+$, computed at superlinear cost (see [43, equation (6)]):*

$$||E||_F = ||M - CUR||_F \le ||M - CC^+M||_F + ||M - MR^+R||_F.$$

Unlike 2-factor LRA of (7.1), CUR LRA is a 3-factor LRA, which can generally be represented as follows:

$$M = M' + E,\ |E| \le \xi,\ M' = ATB,\ A \in \mathbb{R}^{m\times k},\ T \in \mathbb{R}^{k\times l},\ B \in \mathbb{R}^{l\times n}, \tag{7.4}$$

and one typically seeks LRA with $k \ll m$ and/or $l \ll n$. The pairs of maps $AT \to A$ and $B \to B$ as well as $A \to A$ and $TB \to B$ turn a 3-factor LRA ATB of (7.4) into a 2-factor LRA AB of (7.1).

The r-top SVD and a CUR LRA of M are two important examples of 3-factor LRAs.

7.4 Principal Angle Distance

Definition 7.1. *[34]. Let E_1 and E_2 be two subspaces of $\mathbb{R}^m$, and let G, $G_\perp$, H, and $H_\perp$ be matrices with orthonormal columns that generate subspace E_1, $(E_1)_\perp$, E_2, and $(E_2)_\perp$, respectively. Define the* **Principal Angle Distance** *between E_1 and E_2:*

$$\text{Dist}(E_1, E_2) = ||G_\perp^T H|| = ||H_\perp^T G||. \tag{7.5}$$

[9] The pioneering papers [30], [29], [31], [26],[27], [25], [44], and [46] define CGR approximations having nuclei G; "G" can stand, say, for "germ". We use the acronym CUR, which is more customary in the West. "U" can stand, say, for "unification factor", and we notice the alternatives of CNR, CCR, or CSR with N, C, and S standing for *"nucleus", "core", and "seed"*.

Remark 7.2. *Let E_1 and E_2 be two linear subspaces of $\mathbb{R}^m$. Then*
(i) $\mathrm{Dist}(E_1, E_2)$ *ranges from 0 to 1,*
(ii) $\mathrm{Dist}(E_1, E_2) = 0$ *if and only if Span(E_1) = Span(E_2), and*
(iii) $\mathrm{Dist}(E_1, E_2) = 1$ *if rank$(E_1) \neq$ rank(E_2).*

8 Linear least squares and LRA computation with leverage scores

In this section we recall statistical approach to the solution of generalized LLSP and the computation of CUR generators for LRA by means of *subspace sampling* directed by leverage scores. We refer the reader to Appendix C for background on random matrix computations.

8.1 Definition of rank-r leverage scores

Definition 8.1. *Given an $m \times n$ matrix M, with $\sigma_r(M) > \sigma_{r+1}(M)$, and its SVD*

$$M = \begin{bmatrix} U^{(r)} & U_\perp \end{bmatrix} \begin{bmatrix} \Sigma^{(r)} & \\ & \Sigma_\perp \end{bmatrix} \begin{bmatrix} (V^{(r)})^T \\ V_\perp^T \end{bmatrix} \tag{8.1}$$

where $U^{(r)}$ and $V^{(r)}$ are $m \times r$ and $n \times r$ orthogonal matrices, write

$$\gamma_i := \sum_{j=1}^{r} V^{(r)}(i,j)^2, \quad \text{for } i = 1, 2, \ldots, n, \text{ and} \tag{8.2}$$

$$\tilde{\gamma}_i := \sum_{j=1}^{r} U^{(r)}(i,j)^2, \quad \text{for } i = 1, 2, 3, \ldots, m, \tag{8.3}$$

and call γ_i and $\tilde{\gamma}_i$ the rank-r **Column** *and* **Row Leverage Scores** *of M, respectively.*

Remark 8.1. *Notice that $\sum_{i=1}^{m} \tilde{\gamma}_i = \sum_{i=1}^{n} \gamma_i = r$. Therefore these row/column leverage scores naturally define a probability distribution. In fact, we can fix β, $0 < \beta \leq 1$, and by applying one of Algorithms D.1 and D.2 of Appendix D, reproduced from [19], compute the sampling probability distribution $\{p_i | i = 1, ..., n\}$ such that*

$$p_j > 0, \ p_j \geq \beta \gamma_j / r \text{ for } j = 1, \ldots, n, \text{ and } \sum_{j=1}^{n} p_j = 1. \tag{8.4}$$

Given $\tilde{\gamma}_i$, we can fix β, $0 < \beta \leq 1$, and similarly compute distribution $\{\tilde{p}_i | i = 1, ..., m\}$ such that

$$\tilde{p}_j > 0, \ \tilde{p}_j \geq \beta \tilde{\gamma}_j / r \text{ for } j = 1, \ldots, m, \ \sum_{j=1}^{m} \tilde{p}_j = 1. \tag{8.5}$$

Remark 8.2. *Here we assume that $\sigma_k(M) > \sigma_{k+1}(M)$; then the k-top left and right singular spaces of M are uniquely defined.*

8.2 LRA based on solving generalized LLSP and directed by leverage scores

Theorem 8.1 (Adapted from Theorem 5 [19]). *Let $\tilde{\gamma}_i$ for $i = 1, ..., m$ be the rank-r row leverage scores of a rank r matrix $A \in \mathbb{R}^{m\times r}$ and let $M \in \mathbb{R}^{m\times n}$. Fix three positive numbers $\epsilon < 1$, $\xi < 1$, and $\beta \le 1$, and compute probability distribution $\{\tilde{p}_i | i = 1, ..., m\}$ satisfying relationships (8.5). Write $l := 1296\beta^{-1}r^2\epsilon^{-2}\xi^{-4}$ and let S and D be the sampling and scaling matrices output by Algorithm D.1. Then*

$$rank(D^TS^TA) = r \quad \text{and} \quad ||A\tilde{X} - M||_F \le (1+\epsilon)||AA^+M - M||_F \tag{8.6}$$

with a probability no less than $1 - \xi$ where

$$\tilde{X} := (D^TS^TA)^+D^TS^TM. \tag{8.7}$$

Sampling directed by leverage scores has two advantages:

(1) Even with sampling a small number of rows of the matrices A and M we can obtain a very accurate solution, whose error matrix $E = A\tilde{X} - M$ satisfies

$$||E||_F \le (1+\epsilon)\min_X ||AX - M||_F \tag{8.8}$$

whp for any fixed positive ϵ.

(2) Instead of solving generalized LLSP with matrices A and M at superlinear cost we solve it superfast for matrices D^TS^TA and D^TS^TM of much smaller sizes, and whp this yields a very accurate solution to the original generalized LLSP, defined by matrices A and M.

8.3 Matrix CUR LRA directed by leverage scores

The CUR LRA algorithms of [19], implementing this approach, outputs CUR LRA of a matrix M such that whp

$$||M - CUR||_F \le (1+\epsilon)\sigma_{F,r+1} \tag{8.9}$$

for $\sigma_{F,r+1}$ of Theorem 7.1 and any fixed positive ϵ. The algorithm is superfast even for the worst case input, except for the stage of computing leverage scores.

Let us supply some details. Let $M_r = U^{(r)}\Sigma^{(r)}V^{(r)T}$ be r-top SVD where $U^{(r)} \in \mathbb{R}^{m\times r}$, $\Sigma^{(r)} \in \mathbb{R}^{r\times r}$, ,$V^{(r)T} = (\mathbf{t}_j^{(r)})_{j=1}^n \in \mathbb{R}^{r\times n}$ and $\sigma_r(M) > \sigma_{r+1}(M)$.

Let scalars $\gamma_1, \ldots, \gamma_n$ be the *rank-r column leverage scores* for the matrix M (cf. (D.1)). They stay invariant if we pre-multiply the matrix $V^{(r)T}$ by an orthogonal matrix. Furthermore, for a fixed positive $\beta \le 1$, we can compute a sampling probability distribution $p, \ldots, p_n$ at a dominated computational cost, where

$$\tilde{p}_j > 0 \text{ and } \tilde{p}_j \ge \gamma_j/r \text{ for } j = 1, \ldots, n. \tag{8.10}$$

For any $m\times n$ matrix M [32, Algorithm 5.1] computes the matrix $V^{(r)}$ and distribution $p_1, \ldots, p_n$ by using mn memory cells and $O(mnr)$ flops.

Given an integer parameter l, $1 \le l \le n$, and distribution $p_1, \ldots, p_n$, Algorithm D.1 or D.2 computes auxiliary sampling and re-scaling matrices $S = S_{M,l}$ and $D = D_{M,l}$, respectively. (In particular Algorithm D.1 samples and re-scales exactly l columns of an input matrix M – the ith column with probability p_i, while Algorithm D.2 samples and re-scales at most its l columns in expectation – the ith column with probability $\min\{1, lp_i\}$.) Then [19, Algorithms 1 and 2] compute a CUR LRA of a matrix M as follows.

Algorithm 8.1. [CUR LRA by using leverage scores.]

INPUT: *A matrix $M \in \mathbb{R}^{m\times n}$ and a target rank r.*

INITIALIZATION: *Choose two integers $k \geq r$ and $l \geq r$ and real β and $\bar{\beta}$ in the range $(0, 1]$.*

COMPUTATIONS: *1. Compute the probability distribution $p_1, \dots, p_n$ of (8.4).*

2. *Compute sampling and re-scaling matrices S and D by applying Algorithm D.1 or D.2. Compute and output a CUR factor $C := MS$.*
3. *Compute distribution $\tilde{p}_1, \dots, \tilde{p}_m$ satisfying relationships (8.4) under the following replacement: $M \leftarrow (CD)^T$ and $\beta \leftarrow \bar{\beta}$.*
4. *By applying Algorithm D.1 or D.2 to these leverage scores compute $k \times l$ sampling matrix $\bar{S}$ and $k \times k$ re-scaling matrix $\bar{D}$.*
5. *Compute and output a CUR factor $R := \bar{S}^T M$.*
6. *Compute and output a CUR factor $U := DW^+\bar{D}$ for $W := \bar{D}\bar{S}^T MSD$.*

Complexity estimates: Overall Algorithm 8.1 involves $kn+ml+kl$ memory cells and $O((m+k)l^2 + kn)$ flops in addition to mn cells and $O(mnr)$ flops used for computing SVD-based leverage scores at stage 1. Except for that stage the algorithm is superfast if $k + l^2 \ll \min\{m, n\}$.

Bound (8.9) is expected to hold for the output of the algorithm if we choose integers k and l by combining [19, Theorems 4 and 5] as follows.

Theorem 8.2. *Suppose that*
(i) $M \in \mathbb{R}^{m\times n}$, $0 < r \leq \min\{m, n\}$, $\epsilon, \beta, \bar{\beta} \in (0, 1]$, and $\bar{c}$ is a sufficiently large constant,
(ii) four integers k, k_-, l, and l_- satisfy the bounds

$$0 < l_- = 3200r^2/(\epsilon^2\beta) \leq l \leq n \text{ and } 0 < k_- = 3200l^2/(\epsilon^2\bar{\beta}) \leq k \leq m \tag{8.11}$$

or

$$l_- = \bar{c}\, r\log(r)/(\epsilon^2\beta) \leq l \leq n \text{ and } k_- = \bar{c}\, l\log(l)/(\epsilon^2\bar{\beta}) \leq k \leq m, \tag{8.12}$$

(iii) we apply Algorithm 8.1 invoking at stages 2 and 4 either Algorithm D.1 under (8.11) or Algorithm D.2 under (8.12).

Then bound (8.9) holds with a probability at least 0.7.

Remark 8.3. *The bounds $k_- \leq m$ and $l_- \leq n$ imply that either $\epsilon^6 \geq 3200^3 r^4/(m\beta^2\bar{\beta})$ and $\epsilon^2 \geq 3200r/(n\beta)$ if Algorithm D.1 is applied or $\epsilon^4 \geq \bar{c}^2 r\log(r)\log(\bar{c}r\log(r)/(\epsilon^2\beta))/(m\beta^2\bar{\beta})$ and $\epsilon^2 \geq \bar{c}r\log(r)/(n\beta)$ if Algorithm D.2 is applied for a sufficiently large constant $\bar{c}$.*

Remark 8.4. *The estimates k_- and l_- of (8.11) and (8.12) are minimized for $\beta = \bar{\beta} = 1$ and a fixed ϵ. These estimates are proportional to $1/\beta$ and $1/(\beta^2\bar{\beta})$, respectively, and for any fixed numbers k and l of sampled rows/columns in the ranges (8.11) and (8.12) we can ensure randomized error bound (8.9).*

The following result implies that the r-top singular space and hence the leverages scores are stable if the perturbation of a matrix M is relatively small.

Theorem 8.3. (Adapted from [28, Theorem 1].) *Suppose that*

$$g =: \sigma_r(M) - \sigma_{r+1}(M) - 2\,||E|| > 0 \quad \textit{and} \quad ||E||_F \leq \frac{g}{2}.$$

Let $U^{(r)}$, $V^{(r)}$, $U_\perp$, and $V_\perp$ be matrices of the singular vectors of M, defined in equation (8.1). Then there exist matrices $P \in \mathbb{R}^{(m-r)\times r}$ and $Q \in \mathbb{R}^{(n-r)\times r}$ satisfying

$$|||[P^T, Q^T]|||_F < 2\frac{||E||_F}{g} < 1$$

and such that the columns of the matrices $U^{(r)} + U_\perp P$ and $V^{(r)} + V_\perp Q$ span the r-top left and right singular spaces of $M + E$.

Remark 8.5. *The matrices $U^{(r)} + U_\perp P$ and $V^{(r)} + V_\perp Q$ may have non-orthonormal columns, but it can be readily shown that the matrices*

$$\widetilde{U^{(r)}} = (U^{(r)} + U_\perp P)(I_r + P^T P)^{-\frac{1}{2}} \quad \text{and} \quad \widetilde{V^{(r)}} = (V^{(r)} + V_\perp Q)(I_r + Q^T Q)^{-\frac{1}{2}}$$

have orthonormal columns. Let γ_i and $\tilde{\gamma}_i$ denote the rank-r leverage scores of the i-th row of M and $M + E$, respectively, and recall that they correspond to the squared row norms of $U^{(r)}$ and $\widetilde{U^{(r)}}$, respectively. Therefore

$$\frac{1}{1+||P||^2}\left(1 - \frac{||P||}{\sqrt{\gamma_i}}\right)^2 \le \tilde{\gamma}_i/\gamma_i \le \left(1 + \frac{||P||}{\sqrt{\gamma_i}}\right)^2 \text{ for } \gamma_i \neq 0, \quad i = 1, 2, 3, \ldots, m,$$

and $||P|| = O(||E||_F)$ if the perturbation E satisfies the assumption of Theorem 8.3. The bound for the ratio of column leverage scores can be obtained similarly.

Leverage scores are expressed through the singular vectors, and in Section 11.2 we display the results of our tests that show the impact of input perturbation on the leverage scores.

Remark 8.6. *By choosing parameter $\beta < 1$ in (8.4) we can expand the range of perturbations of an input of LRA that can be covered by our study of LRA directed by the leverage scores.*

Remark 8.7. *At stage 6 of Algorithm 8.1 we can alternatively apply the simpler expressions $U := (\bar{S}^T MS)^+ = (S^T C)^+ = (RS)^+$, although this would a little weaken numerical stability of the computation of a nucleus of a perturbed input matrix M.*

9 Superfast randomized iterative refinement of LRA by means of refinement of leverage scores

Given a low rank matrix (e.g., a crude LRA of an input matrix output by the algorithms of [52], [53], or our Section 10, or by a fixed iteration of C-A algorithm applied to that input matrix), let us try to refine it. We observe that we can readily compute top SVD of LRA at a dominated cost; then we can compute leverage scores, again at a dominated cost. By using these scores we can compute new LRA of an input matrix with the hope to obtain a desired refinement, and if we do obtain it, we can reapply these computations recursively. Of course, this is only valuable if we compute a new LRA that refines the original one, and this is our next goal.

We first observe that it is sufficient to refine just one of the two factors A and B that form an LRA AB (hereafter let it be A) because we can compute the second factor superfast by solving a generalized LLSP. Now, given a matrix $A_0 \in \mathbb{R}^{m\times r}$ we first compute a matrix $B_0 \in \mathbb{R}^{r\times n}$ such that $A_0 B_0$ is a crude but reasonably close approximation of an input matrix $M \in \mathbb{R}^{m\times n}$ (we assume that there exists such a matrix B_0); then we successively compute the matrices A_1, B_1, A_2, B_2, ... such that the values $\mathrm{Dist}(A_t, U^{(r)})$ and $\mathrm{Dist}(B_t, V^{(r)})$ converge with a controllable error bound

as $t \to \infty$, where $U^{(r)}$ and $V^{(r)}$ denote two orthogonal matrices whose range (the column span) defines the r-top left and right singular spaces of M, respectively.

There seems to be some similarity of this approach to the algorithm of [34], which recursively decreases the principal angle distance by means of alternating computation of the factors A and B, but that algorithm is restricted to the case of a coherent[10] input matrix with exact rank r and relies on the strategy with uniform element-wise sampling. This is very much different from our approach, which we specify next.

Algorithm 9.1. [Alternating Refinement Using Leverage Scores.]

INPUT: *A matrix $M \in \mathbb{R}^{m \times n}$, an integer τ, a target rank r, positive real numbers ϵ and $\xi < 1$, and a matrix $A_0 \in \mathbb{R}^{m \times r}$.*

COMPUTATIONS:

FOR $t = 0, 1, ..., T$ **DO:**

1. *Compute the row leverage scores $\tilde{\gamma}_j$ of A_t, find an appropriate $0 < \beta \leq 1$, and compute distributions $\tilde{p}_j$ satisfying (8.5) for $j = 1, ..., m$.*
2. *Compute sampling and re-scaling matrices S and D by applying Algorithm D.1 with $l = 1296\beta^{-1}r^2\epsilon^{-2}\xi^{-4}$.*
3. *Compute $B_t = (D^T S^T A_t)^+ D^T S^T M$.*
4. *Compute the column leverage scores γ_j of B_t, find an appropriate $0 < \beta \leq 1$, and compute distributions p_j satisfying (8.4) for $j = 1, ..., n$.*
5. *Compute sampling and re-scaling matrices S and D by applying Algorithm D.1 with $l = 1296\beta^{-1}r^2\epsilon^{-2}\xi^{-4}$.*
6. *Compute $A_{t+1} = MSD(B_t SD)^+$.*

END FOR

OUTPUT: A_{t+1}.

Theorem 9.1. *Let M be an $m \times n$ matrix of (8.1) such that $\sigma_r(M) > \sigma_{r+1}(M)$. Let A be an $m \times r$ orthogonal matrix with $r \leq \min\{m, n\}$ such that*

$$\text{Dist}(A, U^{(r)}) = \delta < 1. \tag{9.1}$$

Fix positive numbers $\epsilon < 1$, $\xi < 1$, and $\beta \leq 1$ and compute the rank-r row leverage scores $\{\gamma_i | i = 1, ..., m\}$ of A and a sampling distribution $\{p_i | i = 1, ..., m\}$ satisfying (8.5). Suppose that Algorithm D.1, applied for $l = 1296\beta^{-1}r^2\epsilon^{-2}\xi^{-4}$, outputs two matrices S and D. Write $B := (D^T S^T A)^+ D^T S^T M$. Then

$$\text{Dist}(B, V^{(r)}) \leq \frac{\delta}{\sqrt{1-\delta^2}} \cdot \frac{\sigma_{r+1}(M)}{\sigma_r(M)} + \frac{2\epsilon}{\sqrt{1-\delta^2}} \cdot \frac{||M - M_r||_F}{\sigma_r(M)} \tag{9.2}$$

with a probability no less than $1 - \xi$.

[10] A matrix is coherent if its maximum row and column leverages scores are small in context.

Proof. For simplicity, let $S' = D^T S^T$ and hence $B = (S'A)^+ S'M$. Assume that B has full rank. Then there exists a QR factorization of B such that

$$B = RQ^T \text{ and } Q^T = R^{-1}B \in \mathbb{R}^{k \times n}.$$

Therefore

$$\begin{aligned} \text{Dist}(B, V^{(r)}) &= ||Q^T V_\perp|| \\ &= ||R^{-1}(S'A)^+ S'MV_\perp|| \\ &= ||R^{-1}(S'A)^+ S'U_\perp \Sigma_\perp|| \\ &\leq ||R^{-1}|| \ \ ||(C_1 A^T + C_2 A_\perp^T)U_\perp \Sigma_\perp|| \\ &\leq \tfrac{1}{\sigma_r(B)} \big(||C_1 A^T U_\perp \Sigma_\perp|| + ||C_2 A_\perp^T U_\perp \Sigma_\perp|| \big). \end{aligned}$$

The former inequality above holds because $\begin{bmatrix} A & A_\perp \end{bmatrix}$ is an orthogonal matrix and because there exists a unique pair of matrices C_1 and C_2 such that the rows of $(S'A)^+ S'$ are expressed as linear combinations of the rows of A^T and $A_\perp^T$ as follows:

$$(S'A)^+ S' = \begin{bmatrix} C_1 & C_2 \end{bmatrix} \cdot \begin{bmatrix} A^T \\ A_\perp^T \end{bmatrix}. \tag{9.3}$$

Given that

(1) $C_1 = I_r$,
(2) $||C_2 A_\perp^T U_\perp \Sigma_\perp|| \leq 2\epsilon ||\Sigma_\perp||_F$, and
(3) $\sigma_r(B) \geq \sqrt{1-\delta^2}\sigma_r(M)$, obtain

$$\text{Dist}(B, V^{(r)}) \leq \frac{\delta}{\sqrt{1-\delta^2}} \cdot \frac{\sigma_{r+1}(M)}{\sigma_r(M)} + \frac{2\epsilon}{\sqrt{1-\delta^2}} \cdot \frac{||\Sigma_\perp||_F}{\sigma_r(M)}.$$

Next we prove that assumptions (1) – (3) above hold provided that the matrix $S' = D^T S^T$ for the matrices D and S from Algorithm D.1 satisfies Equation (8.6) with a probability no less than $1 - \xi$.

Claim (1): Equation (8.6) implies that the matrix $S'A$ has full rank k, and hence

$$C_1 = (S'A)^+ S'A = C_1 A^T A = I_r.$$

Claim (2): Consider the following generalized LLSP,

$$\min_X ||Y - AX||_F$$

where $Y = A_\perp A_\perp^T U_\perp \Sigma_\perp$ denotes an $m \times (n-r)$ matrix. Clearly, $\min_X ||Y - AX||_F = ||Y||_F$ because the column space of Y is orthogonal to the column space of AX.

Furthermore recall that the column spaces of the matrices Y and A are orthogonal to one another. Combine this observation with Equation (9.3) and deduce that

$$\begin{aligned} & ||Y - A(S'A)^+ S'Y||_F^2 \\ = \ & ||Y - A(C_1 A^T + C_2 A_\perp^T)Y||_F^2 \\ = \ & ||Y - AA^T Y - AC_2 A_\perp^T Y||_F^2 \\ = \ & ||Y||_F^2 + ||AC_2 A_\perp^T Y||_F^2. \end{aligned}$$

Recall from Equation (8.6) that

$$||Y - A(S'A)^+S'Y||_F^2 \leq (1+\epsilon)^2||Y||_F^2$$

and conclude that

$$||C_2A_\perp^TY||_F < 2\epsilon||Y||_F = 2\epsilon||\Sigma_\perp||_F.$$

Claim (3): Recall that $B = (S'A)^+S'M$, and therefore

$$\begin{aligned}\sigma_r(B) &= \sigma_r\big((A^T + C_2A_\perp^T)M\big)\\ &\geq \sigma_r(A^TM)\\ &\geq \sigma_r(A^TU^{(r)}\Sigma^{(r)})\\ &\geq \sigma_r(A^TU^{(r)})\cdot\sigma_r(M).\end{aligned}$$

Notice that

$$\begin{aligned}\big(\sigma_r(A^TU^{(r)})\big)^2 &= \sigma_r(A^TU^{(r)}U^{(r)T}A)\\ &= \sigma_r\big(A^T(I_m - U_\perp U_\perp^T)A\big)\\ &= \sigma_r\big(I_r - (A^TU_\perp)(A^TU_\perp)^T\big)\\ &\leq 1-\delta^2 \qquad ,\end{aligned}$$

where the last inequality holds because the matrix $(A^TU_\perp)(A^TU_\perp)^T$ is Symmetric Positive Semi-Definite and has spectral norm $\mathrm{Dist}(A,U^{(r)})^2$. Conclude that $\sigma_r(B) \geq \sqrt{1-\delta^2}\sigma_r(M)$, and this also implies that $\mathrm{rank}(B) = r$. □

Simplify notation by writing $\sigma_j := \sigma_j(M)$ for $j = r$ and $\bar{\sigma}_{r+1} := ||M - M_r||_F$.

Lemma 9.1. *Let $m, n, r, \epsilon, \delta$, M, $U^{(r)}$, $V^{(r)}$, A and B be defined as in Theorem 9.1 such that A and B satisfy Equations (9.1) and (9.2). Then*

$$\mathrm{Dist}(B, V^{(r)}) \leq c\cdot\mathrm{Dist}(A, U^{(r)}),$$

where

$$c = \frac{\sigma_{r+1}}{\sigma_r}\cdot\frac{1}{\sqrt{1-\delta^2}}\cdot(1 + 2\epsilon\cdot\frac{\bar{\sigma}_{r+1}}{\delta\sigma_{r+1}}).$$

Furthermore, if $\frac{\sigma_{r+1}}{\sigma_r}\cdot\frac{1}{\sqrt{1-\delta^2}} < 1$ and $\epsilon\cdot\frac{\bar{\sigma}_{r+1}}{\sigma_{r+1}} < \frac{\delta}{2}\big(\sqrt{1-\delta^2}\frac{\sigma_r}{\sigma_{r+1}} - 1\big)$, then $c < 1$.

If $\mathrm{Dist}(B_t, V^{(r)}) < c\cdot\mathrm{Dist}(A_t, U^{(r)})$ and $\mathrm{Dist}(A_{t+1}, U^{(r)}) < c\cdot\mathrm{Dist}(B_t, V^{(r)})$ for $t \leq T$ and if $0 < c < 1$, then the principal angle distance is reduced by a constant factor $1/c > 1$ each time when for a given A_0 we recursively compute B_0, A_1, B_1, $A_2, \ldots$. In order to ensure that $1/c > 1$, we must have a gap between σ_r and σ_{r+1}; furthermore the initial factor A should be relatively close to $U^{(r)}$ in terms of the principal angle distance. Moreover the second term of the bound (9.2) comes from the error contributed by the perturbation $M - M_r$ and does not converge to zero even if we perform our recursive refinement indefinitely. We, however, are going to decrease the principal angle distance to a value of the order of $\epsilon\cdot\frac{\bar{\sigma}_{r+1}}{\sigma_{r+1}}$, and we can control this by controlling ϵ provided that $\frac{\bar{\sigma}_{r+1}}{\sigma_{r+1}}$ is a reasonably small constant.

In the following, we also make some other reasonable assumptions about an input matrix M and a starting factor A_0 and then show that after a small number of iterations of Algorithm 9.1, the principal angle distance of the output and $U^{(r)}$ converges to a small value whp.

Theorem 9.2. *Suppose that m, n, r, M, $U^{(r)}$, $V^{(r)}$ are defined as in Theorem 9.1,*

$$\frac{\sigma_{r+1}(M)}{\sigma_r(M)} \le \frac{1}{2},\ \frac{\bar{\sigma}_{r+1}}{\sigma_{r+1}} = \theta,\ A_0 \in \mathbb{R}^{m\times r},\ \text{and}\ \operatorname{Dist}(A_0, U^{(r)}) \le \frac{1}{2}.$$

Fix two sufficiently small positive numbers ξ and ϵ such that

$$\xi < 1 \text{ and } \epsilon \le (8\theta)^{-1} \le 1/2,$$

and let A denote the matrix output by Algorithm 9.1 applied for $\tau = \lceil \frac{1}{2}\log_{0.87}(8\theta\cdot\epsilon)\rceil$. Then

$$\operatorname{Dist}(A, U^{(r)}) \le 4\theta\cdot\epsilon \tag{9.4}$$

with a probability no less than $1 - 2\tau\cdot\xi$.

Proof. If $\delta = \delta_t := \operatorname{Dist}(A_t, U^{(r)}) \le 1/2$, then

$$\frac{1}{\sqrt{1-\delta^2}}\frac{\sigma_{r+1}}{\sigma_r} \le \frac{1}{\sqrt{3}}.$$

Furthermore (9.2) implies that

$$\begin{aligned}\operatorname{Dist}(B_t, V^{(r)}) &\le \tfrac{\delta}{\sqrt{1-\delta^2}}\cdot\tfrac{\sigma_{r+1}}{\sigma_r} + \tfrac{2\epsilon}{\sqrt{1-\delta^2}}\cdot\tfrac{\sigma_{r+1}}{\sigma_r}\cdot\tfrac{\bar{\sigma}_{r+1}}{\sigma_{r+1}}\\ &\le \tfrac{1}{\sqrt{3}}\cdot\delta + \tfrac{2\theta}{\sqrt{3}}\cdot\epsilon.\end{aligned}$$

Thus it can be easily verified that

$$\operatorname{Dist}(B_t, V^{(r)}) \le 3\delta/2\sqrt{3} < 0.87\cdot\operatorname{Dist}(A_t, U^{(r)}) \quad \text{if } \delta \ge 4\theta\cdot\epsilon,$$

and that

$$\operatorname{Dist}(B_t, V^{(r)}) \le 6\theta\cdot\epsilon/\sqrt{3} < 4\theta\cdot\epsilon \quad \text{if } \delta < 4\theta\cdot\epsilon.$$

Therefore, starting with A_0 such that by assumption $\operatorname{Dist}(A_0, U^{(r)}) \le 1/2$, every time when we compute B_t from A_t, the distance $\operatorname{Dist}(B_t, V^{(r)})$ stays small or at least does not exceed $0.87\cdot\operatorname{Dist}(A_t, U^{(r)})$ whp. Likewise when we compute A_{t+1} from B_t, the distance $\operatorname{Dist}(A_{t+1}, U^{(r)})$ stays small or decreases by a fixed constant factor compared to $\operatorname{Dist}(B_t, V^{(r)})$ whp, and in both cases we maintain the bound $\operatorname{Dist}(A_t, U^{(r)}) \le 1/2$. We prove this claim by applying Theorem 9.1 for B_t^T and M^T.

By combining the latter results, we obtain for all t such that $\operatorname{Dist}(A_t, U^{(r)}) \le 1/2$ that

$$\operatorname{Dist}(A_{t+1}, U^{(r)}) \le \max\left\{(0.87)^2\operatorname{Dist}(A_t, U^{(r)}), 4\theta\cdot\epsilon\right\} \tag{9.5}$$

with a probability no less than $1 - 2\xi$. Complete the proof of the theorem by combining this bound for $t = 0, ..., \tau - 1$. □

10 LRA with leverage scores for random inputs

The computation of leverage scores is the bottleneck stage of the algorithms of [19], and in this section we bypass that stage simply by assigning the uniform sampling distribution. Then we prove that whp the resulting algorithms still compute accurate CUR LRA of a perturbed factor-Gaussian matrix (see its definition in Appendix C.1), to which we can quite readily transform any matrix that admits LRA (see Theorem C.6 and Remark C.1).

We recall that Theorem 8.3 reduces our task to the case of a factor-Gaussian matrix M. The following theorem further reduces it to the case of a Gaussian matrix.

Theorem 10.1. *Let $M = GH$ for $G \in \mathbb{R}^{m\times r}$ and $H \in \mathbb{R}^{r\times n}$ and let $r = \text{rank}(G) = \text{rank}(H)$. Then the matrices M^T and M share their rank-r leverage scores with the matrices G^T and H, respectively.*

Proof. Let $G = S_G\Sigma_G T_G^* \in \mathbb{C}^{m\times r}$ and $H = S_H\Sigma_H T_H^*$ be SVDs.

Write $W := \Sigma_G T_G^* S_H \Sigma_H$ and let $W = S_W\Sigma_W T_W^*$ be SVD.

Notice that Σ_G, T_G^*, S_H, and Σ_H are $r \times r$ matrices.

Consequently so are the matrices W, S_W, Σ_W, and T_W^*.

Hence $M = \bar{S}_G\Sigma_W\bar{T}_H^*$ where $\bar{S}_G = S_G S_W$ and $\bar{T}_H^* = T_W^* T_H^*$ are orthogonal matrices.

Therefore $M = \bar{S}_G\Sigma_W\bar{T}_H^*$ is SVD.

It follows that the columns of the orthogonal matrices $\bar{S}_G$ and $\bar{T}_H^{*T}$ span the r top right singular spaces of the matrices M^T and M, respectively, and so do the columns of the matrices S_G and T_H^{*T} as well because $\bar{S}_G = S_G S_W$ and $\bar{T}_H^* = T_W^* T_H^*$ where S_W and T_W^* are $r \times r$ orthogonal matrices. This proves the theorem. □

If $M = GH$ (resp. $M^T = H^T G^T$) is a right or diagonally scaled factor-Gaussian matrix, then with probability 1 the matrices M and H (resp. M^T and G^T) share their leverage scores by virtue of Theorem 10.1. If we only know that the matrix M is either a left or a right factor-Gaussian matrix, apply Algorithm 8.1 to both matrices M and M^T and in at least one case reduce the computation of the leverage scores to the case of Gaussian matrix.

Now let $r \ll n$ and outline our further steps of the estimation of the leverage scores.

Outline 10.1. *Recall from [22, Theorem 7.3] or [62] that $\kappa(G) \to 1$ as $r/n \to 0$ for an $r \times n$ Gaussian matrix G. It follows that for $r \ll n$ the matrix G is close to a scaled orthogonal matrix whp; hence within a factor $\frac{1}{\sqrt{n}}$ it is close to the orthogonal matrix T_G^T of its right singular space whp. Therefore the leverage scores p_j of a Gaussian matrix $G = (\mathbf{g}_j)_{j=1}^n$ are close to the values $\frac{1}{rn}||\mathbf{g}_j||^2$, $j = 1, \dots, n$. They, however, are independent in j and close to $1/n$ for all j whp. This choice trivializes the approximation of the leverage scores of a Gaussian matrix and hence of a factor-Gaussian matrix. Since this bottleneck stage of Algorithm 8.1 has been made trivial, the entire algorithm becomes superfast, while it still outputs accurate CUR LRA whp in the case of a factor-Gaussian input. Theorem 8.3 implies extension to a perturbed factor-Gaussian input.*

Next we elaborate upon this outline.

Lemma 10.1. *Suppose that G is an $n \times r$ Gaussian matrix, $\mathbf{u} \in \mathbb{R}^r$, $\mathbf{v} = \frac{1}{\sqrt{n}}G\mathbf{u}$, and $r \le n$. Fix $\bar{\epsilon} > 0$. Then*

$$\text{Probability}\{(1-\bar{\epsilon})||\mathbf{u}||^2 \le ||\mathbf{v}||^2 \le (1+\bar{\epsilon})||\mathbf{u}||^2\} \ge 1 - 2e^{-(\bar{\epsilon}^2-\bar{\epsilon}^3)\frac{n}{4}}.$$

Proof. See [2, Lemma 2]. □

Lemma 10.2. *Fix the spectral or Frobenius norm $|||\cdot|||$ and let $M = S_M\Sigma_M T_M^T$ be SVD. Then $S_M T_M^T$ is an orthogonal matrix and*

$$|||M - S_M T_M^T|||^2 \le |||MM^T - I|||.$$

Proof. $S_M T_M^T$ is an orthogonal matrix because both matrices S_M and T_M^T are orthogonal and at least one of them is a square matrix.

Next observe that $M - S_M T_M^T = S_M\Sigma_M T_M^T - S_M T_M^T = S_M(\Sigma_M - I)T_M^T$, and so

$$|||M - S_M T_M^T||| = |||\Sigma_M - I|||.$$

Likewise $MM^T - I = S_M \Sigma_M^2 S_M^T - I = S_M(\Sigma_M^2 - I)S_M^T$, and so

$$|||MM^T - I||| = |||\Sigma_M^2 - I|||.$$

Complement these equations for the norms with the inequality

$$|||\Sigma_M^2 - I||| = |||\Sigma_M - I||| \; |||\Sigma_M + I||| \geq |||\Sigma_M - I|||,$$

which holds because Σ_M is a diagonal matrix having only nonnegative entries. □

Lemma 10.3. *Suppose that n and $r < n$ are two integers and that $0 < \epsilon < \frac{3r^2}{4}$ such that $n > 1296r^8\epsilon^{-4}$ is sufficiently large. Furthermore let $G = (\mathbf{g}_j)_{j=1}^n$ be an $r \times n$ Gaussian matrix. Then*

$$\Big|\Big|\frac{1}{n}GG^T - I_r\Big|\Big|_F^2 < \epsilon$$

with a probability no less than $1 - 2e^{-(\frac{\epsilon^2}{2} - \frac{2\epsilon^3}{3r^2})\frac{n}{9r^4}}$.

Proof. Let $\mathbf{e}_j$ denote the jth column of the identity matrix I_r. Apply Lemma 10.1 for $\mathbf{u}$ equal to the vectors $\mathbf{e}_j$ and $\mathbf{e}_i - \mathbf{e}_j$, for $\mathbf{v} = \frac{1}{\sqrt{n}}\mathbf{g}_j$, and for $i, j = 1, \ldots, r$ where $i \neq j$. For all i and j in this range substitute $||\mathbf{e}_j|| = 1$ and $||\mathbf{e}_i - \mathbf{e}_j||^2 = 2$ and deduce that

$$1 - \bar{\epsilon} < \frac{1}{n}||\mathbf{g}_j||^2 < 1 + \bar{\epsilon} \text{ and } 2 - \bar{\epsilon} < \frac{1}{n}||\mathbf{g}_i - \mathbf{g}_j||^2 < 2 + \bar{\epsilon} \tag{10.1}$$

with a probability no less than $1 - 2n^2 e^{-(\bar{\epsilon}^2 - \bar{\epsilon}^3)\frac{n}{4}} = 1 - 2e^{-(\bar{\epsilon}^2 - \bar{\epsilon}^3 - \frac{8\ln n}{n})\frac{n}{4}}$. If $\bar{\epsilon} < 1/2$ and $n > 256\bar{\epsilon}^{-4}$, then bounds (10.1) hold with a positive probability no less than $1 - 2e^{-(\frac{\bar{\epsilon}^2}{2} - \bar{\epsilon}^3)\frac{n}{4}}$.

Now, write $\epsilon = \frac{3r^2}{2}\bar{\epsilon}$, and since the (i,j)th entry of the matrix GG^T is given by $\mathbf{g}_i^T\mathbf{g}_j$, deduce that

$$\Big|\Big|\frac{1}{n}GG^T - I_r\Big|\Big|_F^2 \leq \Big(\frac{3}{2}r^2 - \frac{r}{2}\Big)\bar{\epsilon} < \frac{3}{2}r^2\bar{\epsilon} = \epsilon.$$

□

In Lemma 10.3, we proved that an $r \times n$ Gaussian matrix is close to a scaled orthogonal matrix whp if $n > 1296r^8\epsilon^{-4}$. Furthermore, it is clear that if a matrix $G \in \mathbb{R}^{r\times n}$ is "close" to a scaled orthogonal matrix, then the ratios of the squares of column norms and the corresponding leverage scores are "close" to r. In the following lemma, we formalize this observation and provide further details.

Lemma 10.4. *Let $G = (\mathbf{g}_j)_{j=1}^n$ be a $r \times n$ matrix such that $r \leq n$ and $\text{rank}(G) = r$. Let γ_i be the i-th column leverage score and $||\mathbf{g}_i||$ be the i-th column norm. Then*

$$\sigma_r^2(G) \leq \frac{||\mathbf{g}_i||^2}{\gamma_i} \leq \sigma_1^2(G) \text{ for } i = 1, 2, ..., n. \tag{10.2}$$

Proof. Let $G = U\Sigma V^T$ be SVD such that $U \in \mathbb{R}^{r\times r}$, $\Sigma \in \mathbb{R}^{r\times r}$, and $V \in \mathbb{R}^{n\times r}$. Then

$$\gamma_i := \sum_{j=1}^{r} \big(V(i,r)\big)^2,$$

$$||\mathbf{g}_i||^2 = ||U\Sigma V_{i,:}^T||^2 = ||\Sigma V_{i,:}^T||^2$$
$$= \sum_{j=1}^{r} \sigma_j^2(G)\big(V(i,r)\big)^2$$

for $V_{i,;}$ denoting the i-th row vector of matrix V and the i-th column vector of V^T and for $i = 1, 2, 3, ..., n$. □

Corollary 10.1. *Let r, n, G, ϵ be defined as in Lemma 10.3. Then*

$$(1 - \sqrt{\epsilon}) \le \frac{||\mathbf{g}_i||^2/n}{\gamma_i} \le (1 + \sqrt{\epsilon}) \text{ for } i = 1, 2, 3, ..., n$$

with a probability no less than $1 - 2e^{-(\frac{\epsilon^2}{2} - \frac{2\epsilon^3}{3r^2})\frac{n}{9r^4}}$.

Proof. Combine Lemmas 10.3 and 10.4. □

Remark 10.1. *In Corollary 10.1 we proved that whp the leverage scores of a tall skinny Gaussian matrix or its transpose are nearly proportional to the corresponding row or column norms. Next we are going to extend this result to the much more general class of $r \times n$ Gaussian matrices where we allow a moderate increase of the number of row or column samples and then only require that $r < n/2$.*

Observe that the squared norms $||\mathbf{g}_j||^2$ are i.i.d. chi-square random variables $\chi^2(r)$ and therefore are quite strongly concentrated in a reasonable range about their expected values.

Now suppose that $r < n/2$ for a $r \times n$ Gaussian matrix G and simply choose the uniform sampling probability distribution, $p_j = \frac{1}{n}$ for all j. Then we satisfy bounds (8.4) and consequently (8.9) by choosing a reasonably small positive value β.

Let us supply further details.

Lemma 10.5. *Let $Z = \sum_{i=1}^{r} X_i^2$ for i.i.d. standard Gaussian variables $X_1, \ldots, X_r$. Then*

$$\text{Probability}\{Z - r \ge 2\sqrt{rx} + 2rx\} \le e^{-x} \text{ for any } x > 0.$$

Proof. See [37, Lemma 1]. □

Corollary 10.2. *Given two integers n and r such that $r < n/2$, an $r \times n$ Gaussian matrix $G = (\mathbf{g}_j)_{j=1}^n$, denote its rank-$r$ column leverage scores by γ_j for $j = 1, ..., n$ and fix $x > \ln n$ and $0 < \beta < \frac{1}{16e^2(1+4x)}$. Then*

$$\frac{1}{n} > \beta\gamma_j/r \text{ for } j = 1, ..., n \tag{10.3}$$

with a probability no less than $1 - e^{-n\ln 2/2} - e^{-(x - \ln n)}$.

Proof. Apply Theorem C.5 (ii) with $t = 2$ and obtain that

$$\text{Probability}\left\{\sigma_r(G) \le \frac{1}{2e}(\sqrt{n} - \frac{r}{\sqrt{n}})\right\} \le 2^{-(\sqrt{n} - r)}.$$

Substitute $r < n/2$ and obtain

$$\text{Probability}\left\{\sigma_r(G) \le \frac{1}{4e}\sqrt{n}\right\} \le 2^{-n/2} = e^{-n\ln 2/2}.$$

Recall that $||\mathbf{g}_i||^2$ are i.i.d. chi-square random variables with r degrees of freedom and deduce from Lemma 10.5 that

$$\text{Probability}\left\{||\mathbf{g}_i||^2 \geq (1+4x)r\right\} \leq e^{-x}$$

for $x > 1$ and fixed i. Therefore, by using union bound, we obtain that

$$\text{Probability}\left\{||\mathbf{g}_i||^2 \leq (1+4x)r \ \text{ for all } \ i = 1, 2, 3, ..., n\right\} \geq 1 - e^{-(x-\ln n)}$$

Let γ_i denote the i-th column leverage scores of G, assume that $||\mathbf{g}_i||^2 \leq (1+4x)r$ for all $1 \leq i \leq n$ and $\sigma_r^2(G) \geq \frac{n}{16e^2}$, and deduce from the first bound of (10.2) that

$$\gamma_i \leq \frac{16e^2(1+4x)r}{n} \ \text{ for all } \ i = 1, 2, 3, ..., n$$

and consequently

$$\frac{1}{n} \geq \frac{1}{16e^2(1+4x)} \cdot \frac{\gamma_i}{r} \geq \beta\gamma_i/r \ \text{ for all } \ i = 1, 2, 3, ..., n.$$

□

Remark 10.2. *The number of samples $l = 1296r^2\beta^{-1}\epsilon^{-2}\xi^{-4}$ in Theorem 8.1 and $l = 3200r^2\beta^{-1}\epsilon^{-2}$ in Theorem 8.2 contains a factor of β^{-1}. When the sampling probability distribution $\{p_i\}$ are computed approximately or pre-defined, we try to choose $\beta \leq 1$ large enough such that the number of samples does not grow too much, and at the same time is small enough such that relationships (8.4) hold. In Corollary 10.2, we use a parameter x rather than β in order to control the number of required samples and to ensure a specified probability bound that inequality (10.3) holds. For example, let $x = 3 + \ln n$; then $\beta^{-1} \leq 16e^2(4 + \ln n)$ and (10.3) holds with a probability at least $0.95 - e^{-n\ln 2/2}$. Both of these bounds are rather desired; moreover $l = O(r^2\epsilon^{-2}\ln n)$, and so l is dramatically less than n and even than $\sqrt{n}$ for large n.*

We have completed our formal support for Outline 10.1 and arrived at the following result, where one can specify error bounds by using Theorem 8.3 and Corollary 10.2.

Corollary 10.3. *Suppose that the algorithms of [19] have been applied to the computation of CUR LRA of a perturbed factor-Gaussian matrix by using the uniform sampling distribution. Then this computation is superfast and whp outputs reasonably close CUR LRA.*

11 Numerical Tests

11.1 Iterative refinement directed by leverage scores

In this subsection, we present the test results for Algorithm 9.1 on iterative refinement directed by leverage scores. The algorithms are implemented in Python with Numpy and Scipy packages, and for solving generalized LLSP we call **lstsq**, which relies on **Lapack** function **gelsd**. All tests in this subsection are performed on a machine running Mac OS 10.15.7 with 2.6 GHz Intel Core i7 CPU and 16GB of Memory.

For our input, we used three classes of matrices having low numerical rank; we call them *single-layer potential*, *Cauchy*, and *shaw*.

We generated a 3000×3000 *single-layer potential* matrix by discretizing a single-layer potential operator of [32, Section 7.1].

We generated a 2000×2000 *Cauchy* matrix $(\frac{1}{X_i - Y_j})_{i,j=1}^{2000}$ for independent random variables X_i and Y_j uniformly distributed on the intervals $(0, 100)$ and $(100, 200)$ respectively. This matrix has fast decaying singular values (cf. [10]).

We defined a 1000×1000 *shaw* matrix by means of discretization of a one-dimensional image restoration model; see further comments on this class of matrices in the next subsection.

For a factor A_0 initializing iterative refinement we computed a matrix from SVD of the initial approximation $\tilde{M}$, defining its top-r left singular space. For this computation we applied Algorithm 8.1 with $k = l = r$, $\beta = \bar{\beta} = 1$, and the uniform probability distribution at step 1 (in which case the algorithm remains superfast); here r denotes the target rank of the input matrix; $r = 11$ for the *single-layer potential* input, and $r = 10$ for the two other input classes.

At the refinement stage of our tests, we sampled d columns and d rows, for $d = 15r$, which turned out to be sufficiently large in our tests, even though support for our proofs required much larger samples of order $r^2\epsilon^{-2}$.

For comparison, we also included the results of testing two other approaches to solving the generalized LLSP.

One approach is the well-known approximation through subspace embedding with a Gaussian multiplier, where in each iteration we compute $B_i = \arg\min_X ||GA_iX - GM||_F$ and $A_{i+1} = \arg\min_X ||XB_iH - MH||_F$ and where G and H are Gaussian matrices of size $d \times m$ and $n \times d$, respectively, generated independently in each iteration.

The other approach is the standard Linear Least Squares Solving, where one computes

$$B_i = \arg\min_X ||A_iX - M||_F \text{ and } A_{i+1} = \arg\min_X ||XB_i - M||_F;$$

this quite costly algorithm outputs an optimal solution.

We repeat each test 10 times; each time we compute the average principal angle distance, the residual error ratio $\frac{||M - A_iB_i||_F}{||M - M_r||_F}$, and the run time in each iteration.

We plot our tests results in Figure 5. They show that the low rank approximation improves dramatically in the first few iterations in terms of the Principal Angle Distance and Residual Error Ratio and is stabilized in the subsequent iterations.

While the approach with the standard algorithms for generalized LLSP has lead us to the best results, as one should have expected, the other approach still enabled us to compute approximations that are close to optimal.

For the *Single Layer Potentials* input, the principal Angle Distance of the refinement with Gaussian Embedding and the refinement with Leverage Scores are slightly worse than the approximations of other inputs. This could be caused by the "heavier" tail singular values. However, the approximation remains rather close to the optimal in terms of Residual Error Ratio regardless.

In terms of run time, the two approaches provide significant improvement over the standard LLSP-based Approach. Moreover the leverage scores approach uses considerably less time at each iteration than the Gaussian Embedding approach, and the run time decreases more substantially as the size of the input matrix grows. Even more importantly, in our tests this improvement comes at almost no cost in terms of the precision growth.

11.2 Testing perturbation of leverage scores

Table 11.1 shows the mean and standard deviation of the norms of the relative errors of approximation of the input matrix M and of its LRA AB and similar data for the maximum difference between the leverage scores of the pairs of these matrices. We have computed a close approximation to the leverage scores of an input matrix M superfast by using its LRA AB. The table also

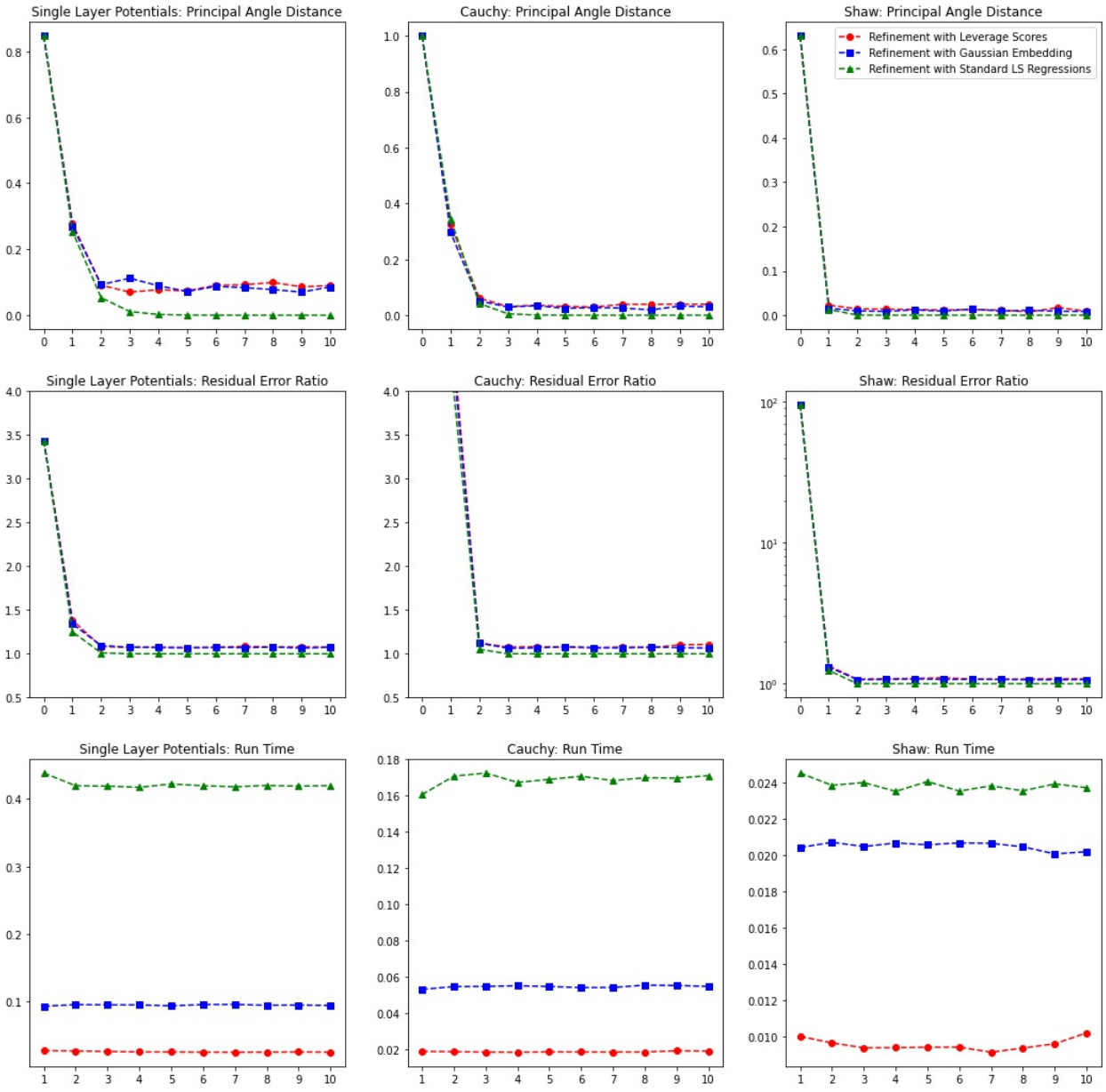

Figure 5: Iterative Refinement directed by Leverage Scores

displays numerical ranks of input matrices M defined up to tolerance 10^{-6}. Our statistics were gathered from 100 runs for each input matrix under 100 runs of sampling and re-scaling algorithm of Appendix D, reproduced from [19].

Input matrices. The dense matrices with smaller ratios of "numerical rank/n" from the built-in test problems in Regularization Tools, which came from discretization (based on Galerkin or quadrature methods) of the Fredholm Integral Equations of the first kind,[11] namely to the following six input classes from the Database:

baart: Fredholm Integral Equation of the first kind,
shaw: one-dimensional image restoration model,
gravity: 1-D gravity surveying model problem,
wing: problem with a discontinuous solution,
foxgood: severely ill-posed problem,
laplace: inverse Laplace transformation.

We computed the LRA approximations AB by using [58, Algorithm 1.1] with multipliers of Class 5 of [58, Section 5.3].

Our goal was to compare the approximate leverage scores with their true values. The columns "mean(Leverage Score Error)" and "std(Leverage Score Error)" of the table show that these approximations become more accurate as r increases.

In addition, the last three lines of Table 11.1 show similar results for perturbed two-sided factor-Gaussian matrices GH of rank r approximating an input matrix M up to perturbations.

PART III. Conclusions

We studied dual superfast algorithms for LLSP and LRA and proved that they output meaningful or even nearly optimal solutions whp under some natural randomization models in the spaces of input matrices. Next we list some natural further challenges.

1. Recall the known algorithms for LLSP, LRA, and possibly other challenging tasks of matrix computations based on the JL transforms, consider their modifications that are superfast or attractive in another sense, e.g., consider sampling for LLSP and LRA based on leverage scores that use fewer rows and columns, that is, decrease the parameter l of Theorem 8.1, and then prove that the algorithms still output valuable solutions for a large class of input matrices.

2. Define superfast (sublinear cost) algorithms relative to the number of NONZERO entries of an input matrix rather than to all its entries and then devise such algorithms for LLSP, LRA, and possibly other challenging tasks of matrix computations (cf. [1]).

3. In our study we have ad hoc combined randomization of the JL transforms and of the space of input matrices. Can we yield superfast solution for a wider input class based on more judicious combinations of this kind? Can this goal be achieved by means of recursive application of various superfast JL-like transforms and/or various pre-processing algorithms for input matrices?

4. Among the three known algorithms for iterative refinement of LRA – of [49], our Section 9, and by means of C-A iterations, the latter algorithm is apparently most efficient, but besides comparing the algorithms it may be interesting to combine their power, possibly just by properly intertwining the iterations of two or three kinds.

Appendix

[11] See http://www.math.sjsu.edu/singular/matrices and http://www2.imm.dtu.dk/∼pch/Regutools
For more details see Chapter 4 of the Regularization Tools Manual at
http://www.imm.dtu.dk/∼pcha/Regutools/RTv4manual.pdf

			LRA Rel Error		Leverage Score Error	
Input Matrix	r	rank	mean	std	mean	std
baart	4	6	6.57e-04	1.17e-03	1.57e-05	5.81e-05
baart	6	6	7.25e-07	9.32e-07	5.10e-06	3.32e-05
baart	8	6	7.74e-10	2.05e-09	1.15e-06	3.70e-06
foxgood	8	10	5.48e-05	5.70e-05	7.89e-03	7.04e-03
foxgood	10	10	9.09e-06	8.45e-06	1.06e-02	6.71e-03
foxgood	12	10	1.85e-06	1.68e-06	5.60e-03	3.42e-03
gravity	23	25	3.27e-06	1.82e-06	4.02e-04	3.30e-04
gravity	25	25	8.69e-07	7.03e-07	4.49e-04	3.24e-04
gravity	27	25	2.59e-07	2.88e-07	4.64e-04	3.61e-04
laplace	23	25	2.45e-05	9.40e-05	4.85e-04	3.03e-04
laplace	25	25	3.73e-06	1.30e-05	4.47e-04	2.78e-04
laplace	27	25	1.30e-06	4.67e-06	3.57e-04	2.24e-04
shaw	10	12	6.40e-05	1.16e-04	2.80e-04	5.17e-04
shaw	12	12	1.61e-06	1.60e-06	2.10e-04	2.70e-04
shaw	14	12	4.11e-08	1.00e-07	9.24e-05	2.01e-04
wing	2	4	1.99e-02	3.25e-02	5.17e-05	2.07e-04
wing	4	4	7.75e-06	1.59e-05	7.17e-06	2.30e-05
wing	6	4	2.57e-09	1.15e-08	9.84e-06	5.52e-05
factor-Gaussian	25	25	1.61e-05	3.19e-05	4.05e-08	8.34e-08
factor-Gaussian	50	50	2.29e-05	7.56e-05	2.88e-08	6.82e-08
factor-Gaussian	75	75	4.55e-05	1.90e-04	1.97e-08	2.67e-08

Table 11.1: Test results for the perturbation of leverage scores

A ASPH matrices

A $k \times n$ matrix of *Subsampled Randomized Hadamard Transform (SRHT),* for $n = 2^s$ being the sth power of 2, is defined as the product PHD where P is a $k \times n$ random column sampling matrix; H is the $n \times n$ Hadamard matrix, and D is an $n \times n$ diagonal matrix with ± 1 on the diagonal and where one can assign signs $+$ and $-$ independently for every entry with a probability 0.5.

The Hadamard matrix $H = H_n$ is defined recursively:

$$H_0 = 1, H_{i+1} = \begin{pmatrix} H_i & H_i \\ H_i & -H_i, \end{pmatrix} \quad i = 0, 1, \dots, s-1.$$

For any integer d, not exceeding s, define the d–Abridged $n \times n$ Hadamard matrix $H_{d,d}$ as follows:

$$H_{d,0} = I_{n/2^d}, H_{d,i+1} = \begin{pmatrix} H_{d,i} & H_{d,i} \\ H_{d,i} & -H_{d,i}, \end{pmatrix} \quad i = 0, 1, \dots, d-1.$$

The matrix $H_{d,d}$ is orthogonal up to scaling and is filled with 0s, except that it has exactly 2^d nonzero entries 1 or -1 in every row and every column. Thus it is very sparse for small integers $d \ll s$ and can be multiplied by a vector by using exactly $(2^d - 1)n$ additions and subtractions. For $d = s$ we arrive at the Hadamard matrix $H_{d,d} = H = H_n$.

Finally define the *d–Abridged $k \times n$ Scaled and Permuted Hadamard (ASPH)* matrices as $PH_{d,d}D$ for the matrices P, $H_{d,d}$, and D defined above.

B Small families of hard inputs for superfast LRA

Any superfast LRA algorithm fails on the following small input families.

Example B.1. *Define a family of $m \times n$ matrices of rank 1 (we call them δ-*matrices*):*

$$\{\Delta_{i,j},\ i = 1, \dots, m;\ j = 1, \dots, n\}.$$

Also include the $m \times n$ null matrix $O_{m,n}$ into this family. Now fix any superfast algorithm; it does not access the (i, j)th entry of its input matrices for some pair of i and j. Therefore it outputs the same approximation of the matrices $\Delta_{i,j}$ and $O_{m,n}$, with an undetected error at least 1/2. Apply the same argument to the set of $mn + 1$ small-norm perturbations of the matrices of the above family and to the $mn + 1$ sums of the latter matrices with any fixed $m \times n$ matrix of low rank. Finally, the same argument shows that a posteriori estimation of the output errors of an LRA algorithm applied to the same input families cannot run superfast.

This example actually covers randomized LRA algorithms as well. Indeed suppose that an LRA algorithm does not access a constant fraction of the entries of an input matrix with a positive constant probability. Then for some pair i, j with a positive constant probability the algorithm misses an (i, j)th entry of an input matrix $\Delta_{i,j}$ and outputs the same approximation to it and the matrix $O_{m,n}$. Therefore whp the algorithm fails to approximate that entry closely for at least one of these two matrices of the first family of input matrices of the above example, and similarly for its other input families. This, however, is a special case of input degeneration; this paper, [52], [53], and [47] show that apart from such cases various superfast algorithms tend to output reasonably close LRA of a matrix that admits LRA.

C Background on random matrix computations

C.1 Gaussian and factor-Gaussian matrices of low rank and low numerical rank

Theorem C.1. [Nondegeneration of a Gaussian Matrix.] *Suppose that $F \in \mathbb{R}^{r\times m}$, $H \in \mathbb{R}^{n\times r}$, $M \in \mathbb{R}^{m\times n}$, F and H are Gaussian matrices, and $r \le \text{rank}(M)$. Then the matrices F, H, FM, and MH have full rank r with probability 1.*

Proof. Fix any of the matrices F, H, FM, and MH and its $r \times r$ submatrix B. Then the equation $\det(B) = 0$ defines an algebraic variety of a lower dimension in the linear space of the entries of the matrix because in this case $\det(B)$ is a polynomial of degree r in the entries of the matrix F or H (cf. [11, Proposition 1]). Clearly, such a variety has Lebesgue and Gaussian measures 0, both being absolutely continuous with respect to one another. This implies the theorem. □

Assumption C.1. [Nondegeneration of a Gaussian matrix.] *Hereafter we simplify the statements of our results by assuming that a Gaussian matrix has full rank and ignoring the probability 0 of its degeneration.*

Definition C.1. [Factor-Gaussian matrices.] *Let $\rho \le \min\{m, n\}$ and define the classes of* left, right, *and* two-sided factor-Gaussian matrices of rank ρ, $G_{m,\rho}B$, $AG_{\rho,n}$, *and* $G_{m,\rho}\Sigma G_{\rho,n}$, *respectively, where $G_{p,q}$ denotes a $p \times q$ Gaussian matrix, $A \in \mathbb{R}^{m\times\rho}$, $B \in \mathbb{R}^{\rho\times n}$, $\Sigma \in \mathbb{R}^{\rho\times\rho}$, A, B, and Σ are well-conditioned matrices of full rank ρ, and $\Sigma = (\sigma_j)_{j=1}^{\rho}$ such that $\sigma_1 \ge \sigma_2 \ge \cdots \ge \sigma_\rho > 0$.*

Theorem C.2. *The class of two-sided $m \times n$ factor-Gaussian matrices $G_{m,\rho}\Sigma G_{\rho,n}$ of rank ρ does not change in the transition to the matrices $G_{m,r}CG_{r,n}$ for a well-conditioned nonsingular $\rho \times \rho$ matrix C.*

Proof. Let $C = U_C \Sigma_C V_C^*$ be SVD. Then the $m \times r$ matrix $A = G_{m,r} U_C$ and the $r \times n$ matrix $B = V_C^* G_{r,n}$ are Gaussian by virtue of orthogonal invariance of Gaussian matrices, and so $G_{m,r} C G_{r,n} = A \Sigma_C B$ for two Gaussian matrices A of size $m \times r$ and B of size $r \times n$. □

Definition C.2. The relative norm of a perturbation of a Gaussian matrix *is the ratio of the perturbation norm and the expected value of the norm of the matrix (estimated in Theorem C.4).*

We refer to all three matrix classes above as *factor-Gaussian matrices of rank* r, to their perturbations within a relative norm bound ϵ as *factor-Gaussian matrices of* ϵ*-rank* r, and to their perturbations within a small relative norm as *factor-Gaussian matrices of numerical rank* r, to which we also refer as *perturbations of factor-Gaussian matrices.*

Clearly $||(A\Sigma)^+|| \le ||\Sigma^{-1}||\, ||A^+||$ and $||(\Sigma B)^+|| \le ||\Sigma^{-1}||\, ||B^+||$ for a two-sided factor-Gaussian matrix $M = A\Sigma B$ of rank r of Definition C.1, and so whp such a matrix is both left and right factor-Gaussian of rank r.

We readily verify the following result.

Theorem C.3. *(i) A submatrix of a two-sided (resp. scaled) factor-Gaussian matrix of rank* ρ *is a two-sided (resp. scaled) factor-Gaussian matrix of rank* ρ*, (ii) a* $k \times n$ *(resp.* $m \times l$*) submatrix of an* $m \times n$ *left (resp. right) factor-Gaussian matrix of rank* ρ *is a left (resp. right) factor-Gaussian matrix of rank* ρ*.*

C.2 Norms of a Gaussian matrix and its pseudo inverse

Hereafter $\Gamma(x) = \int_0^\infty \exp(-t) t^{x-1} dt$ denotes the Gamma function, $\mathbb{E}(v)$ denotes the *expected value* of a random variable v, and $e := 2.71828\ldots$.

Definition C.3. [Norms of a Gaussian matrix and its pseudo inverse.] *Write* $\nu_{m,n} = ||G||$ *and* $\nu^+_{m,n} = ||G^+||$*, for a Gaussian* $m \times n$ *matrix* G*. (*$\nu_{m,n} = \nu_{n,m}$ *and* $\nu^+_{m,n} = \nu^+_{n,m}$*, for all pairs of* m *and* n*.)*

Theorem C.4. [Norm of a Gaussian matrix, see [21, Theorem II.7].]
Probability$\{\nu_{m,n} > t + \sqrt{m} + \sqrt{n}\} \le \exp(-t^2/2)$ *for* $t \ge 0$, $\mathbb{E}(\nu_{m,n}) \le \sqrt{m} + \sqrt{n}$.

Theorem C.5. [Norms of the pseudo inverse of a Gaussian matrix.]
(i) Probability $\{\nu^+_{m,n} \ge m/x^2\} < \frac{x^{m-n+1}}{\Gamma(m-n+2)}$ *for* $m \ge n \ge 2$ *and all positive* x*,*
(ii) Probability $\{\nu^+_{m,n} \ge t \frac{e\sqrt{m}}{m-n+1}\} \le t^{n-m}$ *for all* $t \ge 1$ *provided that* $m \ge 4$*,*
(iii) $\mathbb{E}(\nu^+_{m,n}) \le \frac{e\sqrt{m}}{m-n}$ *provided that* $m \ge n + 2 \ge 4$*,*
(iv) Probability $\{\nu^+_{n,n} \ge x\} \le \frac{2.35\sqrt{n}}{x}$ *for* $n \ge 2$ *and all positive* x*, and furthermore* $||M_{n,n} + G_{n,n}||^+ \le \nu_{n,n}$ *for any* $n \times n$ *matrix* $M_{n,n}$ *and an* $n \times n$ *Gaussian matrix* $G_{n,n}$*.*

Proof. See [13, Proof of Lemma 4.1] for claim (i), [32, Proposition 10.4 and equations (10.3) and (10.4)] for claims (ii) and (iii), and [64, Theorem 3.3] for claim (iv). □

Theorem C.5 implies reasonable probabilistic upper bounds on the norm $\nu^+_{m,n}$ even where the integer $|m-n|$ is close to 0; whp the upper bounds of Theorem C.5 on the norm $\nu^+_{m,n}$ decrease very fast as the difference $|m-n|$ grows from 1.

C.3 Randomized pre-processing of lower rank matrices

The following simple results, (cf. [53, Section 8.2]), where $A \preceq B$ means that A is statistically less than B, show that pre-processing with Gaussian multipliers X and Y transforms any matrix that admits LRA into a perturbation of a factor-Gaussian matrix.

Theorem C.6. *Consider five integers k, l, m, n, and ρ satisfying (7.3), an $m \times n$ well-conditioned matrix M of rank ρ, $k \times m$ and $n \times l$ Gaussian matrices G and H, respectively, and the norms $\nu_{p,q}$ and $\nu^+_{p,q}$ of Definition C.3. Then*

(i) GM is a left factor-Gaussian matrix of rank ρ such that

$$||GM|| \preceq ||M||\ \nu_{k,\rho} \text{ and } ||(GM)^+|| \preceq ||M^+||\ \nu^+_{k,\rho},$$

(ii) MH is a right factor-Gaussian matrix of rank ρ such that

$$||MH|| \preceq ||M||\ \nu_{\rho,l} \text{ and } ||(MH)^+|| \preceq ||M^+||\ \nu^+_{\rho,l},$$

(iii) GMH is a two-sided factor-Gaussian matrix of rank ρ such that

$$||GMH|| \preceq ||M||\ \nu_{k,\rho}\nu_{\rho,l} \text{ and } ||(GMH)^+|| \preceq ||M^+||\ \nu^+_{k,\rho}\nu^+_{\rho,l}.$$

Remark C.1. *Based on this theorem we can readily extend our results on LRA of perturbed factor-Gaussian matrices to all matrices that admit LRA and are pre-processed with Gaussian multipliers. We cannot perform such pre-processing superfast, but empirically superfast pre-processing with various sparse orthogonal multipliers works as efficiently [50], [51], [52].*

D Computation of Sampling and Re-scaling Matrices

We begin with the following simple computations. Given an n vectors $\mathbf{v}_1, \dots, \mathbf{v}_n$ of dimension l, such that $V = (\mathbf{v}_i)_{i=1}^n$ is orthogonal, and compute n leverage scores

$$\gamma_i = \mathbf{v}_i^T \mathbf{v}_i / ||V||_F^2, i = 1, \dots, n. \tag{D.1}$$

Notice that $\gamma_i \geq 0$ for all i and $\sum_{i=1}^n \gamma_i = 1$.

Now assume that some sampling distribution $p_1, \dots, p_n$ satisfying Equation (8.4) are given to us and next recall [19, Algorithms 4 and 5]. For a fixed positive integer l they sample either exactly l columns of an input matrix W (the ith column with a probability p_i) or at most l its columns in expectation (the ith column with a probability $\min\{1, lp_i\}$), respectively.

Algorithm D.1. (The Exactly(l) Sampling and Re-scaling. [19, Algorithm 4]).

INPUT: *Two integers l and n such that $1 \leq l \leq n$ and n positive scalars $p_1, \dots, p_n$ such that $\sum_{i=1}^n p_i = 1$.*

INITIALIZATION: *Write $S := O_{n,l}$ and $D := O_{l,l}$.*

COMPUTATIONS: *(1) For $t = 1, \dots, l$ do*

Pick $i_t \in \{1, \dots, n\}$ such that Probability$(i_t = i) = p_i$;

$s_{i_t,t} := 1$;

$d_{t,t} = 1/\sqrt{lp_{i_t}}$;

end

(2) Write $s_{i,t} = 0$ for all pairs of i and t unless $i = i_t$.

OUTPUT: *$n \times l$ sampling matrix $S = (s_i, t)_{i,t=1}^{n,l}$ and $l \times l$ re-scaling matrix $D = \mathrm{diag}(d_{t,t})_{t=1}^{l}$.*

The algorithm performs l searches in the set $\{1, \dots, n\}$, l multiplications, l divisions, and the computation of l square roots.

Algorithm D.2. (The Expected(l) Sampling and Re-scaling. [19, Algorithm 5]).

INPUT, OUTPUT AND INITIALIZATION *are as in Algorithm D.1.*

COMPUTATIONS: *Write* $t := 1$*;*

for $t = 1, \dots, l-1$ *do*

for $j = 1, \dots, n$ *do*

Pick j *with the probability* $\min\{1, lp_j\}$*;*

if j *is picked, then*

$s_{j,t} := 1$*;*

$d_{t,t} := 1/\min\{1, \sqrt{lp_j}\}$*;*

$t := t + 1$*;*

end

end

Algorithm D.2 involves nl memory cells. $O((l+1)n)$ flops, and the computation of l square roots.

Acknowledgements: Our work has been supported by NSF Grants CCF–1116736, CCF–1563942 and CCF–1733834 and PSC CUNY Award 69813 00 48. We are also grateful to E. E. Tyrtyshnikov for the challenge of formally supporting empirical power of C–A iterations, to N. L. Zamarashkin for his comments on his work with A. Osinsky on LRA via volume maximization and on the first draft of [51], and to S. A. Goreinov, I. V. Oseledets, A. Osinsky, E. E. Tyrtyshnikov, and N. L. Zamarashkin for reprints and pointers to relevant bibliography.

References

[1] N. Ailon, B. Chazelle, Approximate nearest neighbors and the fast Johnson – Lindenstrauss transform, *ACM STOC'06*, 557–563 (2006). doi:10.1145/1132516.1132597.

[2] R.I. Arriaga, S. Vempala, An Algorithmic Theory of Learning: Robust Concepts and Random Projection, *Machine Learning*, **63, 2**, 161–182, May 2006.

[3] H. Avron, P. Maymounkov, S. Toledo, Blendenpik: Supercharging LAPACK's Least-squares Solver, *SIAM Journal on Scientific Computing*, **32**, 1217–1236, 2010.

[4] A. Björk, *Numerical Methods for Least Squares Problems*, SIAM 1996. ISBN-10 : 0898713609 ISBN-13 : 978-0898713602

[5] M. Bebendorf, Approximation of Boundary Element Matrices, *Numer. Math.*, **86, 4**, 565–589, 2000.

[6] E. K. Bjarkason, Pass-Efficient Randomized Algorithms for Low-Rank Matrix Approximation Using Any Number of Views, *SIAM Journal on Scientific Computing*, **41(4)**, A2355 - A2383 (2019), https://doi.org/10.1137/18M118966X, preprint in arXiv:1804.07531 (2018).

[7] C. Boutsidis, P. Drineas, Random projections for the nonnegative least-squares problem, *Linear Algebra and its Applications*, **431**, 760–771 (2009).

[8] M. Bebendorf, R. Grzhibovskis, Accelerating Galerkin BEM for linear elasticity using adaptive cross approximation, *Math. Methods Appl. Sci.*, **29**, 1721—1747, 2006.

[9] M. Bebendorf, S. Rjasanow, Adaptive Low-Rank Approximation of Collocation Matrices, *Computing*, **70, 1**, 1–24, 2003.

[10] B. Beckerman, A. Townsend, On the singular values of matrices with displacement structure, *SIAM Journal on Matrix Analysis and Applications*, **30, 4**, 1227-1248, 2017.

[11] W. Bruns, U. Vetter, *Determinantal Rings, Lecture Notes in Math.,* **1327**, Springer, Heidelberg, 1988.

[12] P. Cortez, A. Cerdeira, F. Almeida, T. Matos and J. Reis, Modeling wine preferences by data mining from physicochemical properties, In *Decision Support Systems*, Elsevier, **47 4**, 547–553, 2009.

[13] Z. Chen, J. J. Dongarra, Condition Numbers of Gaussian Random Matrices, *SIAM J. on Matrix Analysis and Applications*, **27**, 603–620, 2005.

[14] C. Cichocki, N. Lee, I. Oseledets, A.-H. Phan, Q. Zhao, D. P. Mandic, Tensor Networks for Dimensionality Reduction and Large-scale Optimization: Part 1, Low-Rank Tensor Decompositions, in *Foundations and Trends® in Machine Learning*: **9, 4-5**, 249–429, 2016. http://dx.doi.org/10.1561/2200000059

[15] K. L. Clarkson, D. P. Woodruff, Numerical linear algebra in the streaming model, in *Proc. 41st ACM Symposium on Theory of Computing (STOC 2009)*, pages 205–214, ACM Press, New York, 2009.

[16] K. L. Clarkson, D. P. Woodruff, Low Rank Approximation and Regression in Input Sparsity Time, *Journal of the ACM*, **63, 6**, Article 54, 2017; doi: 10.1145/3019134.

[17] Kannan, R., Drineas, P., Mahoney, M. W., Fast Monte Carlo Algorithms for matrices I, II, III, *SIAM J. on Computing*, **36 (1)**, 132 – 206, 2006.

[18] Drineas, P., Mahoney, M. W., Muthukrishnan, S., Sampling algorithms for l_2 regression and applications. In *Proc. of the 17th Annual ACM-SIAM Symposium on Discrete Algorithms (SODA'06)*, pp. 1127-1136, 2006.

[19] P. Drineas, M.W. Mahoney, S. Muthukrishnan, Relative-error CUR Matrix Decompositions, *SIAM Journal on Matrix Analysis and Applications*, **30, 2**, 844–881, 2008.

[20] P. Drineas, M. W. Mahoney, S. Muthukrishnan, T. Sarlós, Faster Least Squares Approximation, *Numerische Mathematik*, **117 (2)**, 219–249 (2011)
DOI: 10.1007/s00211-010-0331-6 arXiv:0710.1435 26 Sep 2010.

[21] K. R. Davidson, S. J. Szarek, Local Operator Theory, Random Matrices, and Banach Spaces, in *Handbook on the Geometry of Banach Spaces* (W. B. Johnson and J. Lindenstrauss editors), pages 317–368, North Holland, Amsterdam, 2001.

[22] A. Edelman, Eigenvalues and Condition Numbers of Random Matrices, Ph.D. thesis, Massachusetts Institute of Technology, 1989.

[23] M. Gu, Subspace iteration randomization and singular value problems, *SIAM J. on Scientific Computing*, **37, 3**, A1139–1173 (2015) https://doi.org/10.1137/130938700

[24] G. H. Golub, C. F. Van Loan, *Matrix Computations*, The Johns Hopkins University Press, Baltimore, Maryland, 2013 (fourth edition).

[25] S. Goreinov, I. Oseledets, D. Savostyanov, E. Tyrtyshnikov, N. Zamarashkin, How to Find a Good Submatrix, in *Matrix Methods: Theory, Algorithms, Applications* (dedicated to the Memory of Gene Golub, edited by V. Olshevsky and E. Tyrtyshnikov), pages 247–256, World Scientific Publishing, New Jersey, ISBN-13 978-981-283-601-4, ISBN-10-981-283-601-2, 2010.

[26] S. A. Goreinov, E. E. Tyrtyshnikov, The Maximal-Volume Concept in Approximation by Low Rank Matrices, *Contemporary Mathematics*, **208**, 47–51, 2001.

[27] S. A. Goreinov, E. E. Tyrtyshnikov, Quasioptimality of Skeleton Approximation of a Matrix on the Chebyshev Norm, *Russian Academy of Sciences: Doklady, Mathematics (DOKLADY AKADEMII NAUK)*, **83, 3**, 1–2, 2011.

[28] Gratton, Serge, and J. Tshimanga-Ilunga. On a second-order expansion of the truncated singular subspace decomposition, *Numerical Linear Algebra with Applications*, **23**, **3**, 519-534, 2016.

[29] S. A. Goreinov, E. E. Tyrtyshnikov, N. L. Zamarashkin, A Theory of Pseudo-skeleton Approximations, *Linear Algebra and Its Applications*, **261**, 1–21, 1997.

[30] S. A. Goreinov, N. L. Zamarashkin, E. E. Tyrtyshnikov, Pseudo-skeleton approximations, *Russian Academy of Sciences: Doklady, Mathematics (DOKLADY AKADEMII NAUK)*, **343, 2**, 151–152, 1995.

[31] S. A. Goreinov, N. L. Zamarashkin, E. E. Tyrtyshnikov, Pseudo-skeleton Approximations by Matrices of Maximal Volume, *Mathematical Notes*, **62, 4**, 515–519, 1997.

[32] N. Halko, P. G. Martinsson, J. A. Tropp, Finding Structure with Randomness: Probabilistic Algorithms for Constructing Approximate Matrix Decompositions, *SIAM Review*, **53, 2**, 217–288, 2011.

[33] Johnson, W. B.; Lindenstrauss, J., Extension of Lipschitz mappings into a Hilbert space. In Beals, Richard; Beck, Anatole; Bellow, Alexandra; et al. (eds.). *Conference in modern analysis and probability* (New Haven, Conn., 1982), *Contemporary Mathematics*, **26**, Providence, RI: American Mathematical Society. pp. 189–206. doi:10.1090/conm/026/737400. ISBN 0-8218-5030-X. MR 0737400.

[34] P. Jain, P. Netrapalli, S. Sanghavi, Low-rank matrix completion using alternating minimization, *In Proceedings of the Forty-fifth Annual ACM Symposium on Theory of Computing*, pp. 665-674, 2013.

[35] N. Kishore Kumar, J. Schneider, Literature Survey on Low Rank Approximation of Matrices, *Linear and Multilinear Algebra*, **65 (11)**, 2212-2244, 2017, and arXiv:1606.06511v1 [math.NA] 21 June 2016.

[36] C. L. Lawson, R. J. Hanson, *Solving Least Squares Problems*, SIAM 1995. ISBN: 978-0-89871-356-5 eISBN: 978-1-61197-121-7

[37] B. Laurent, P. Massart, Adaptive Estimation of a Quadratic Functional by Model Selection, *The Annals of Statistics* **28, 5**, 1302–1338, 2000. Also see http://www.jstor.org/stable/2674095

[38] Q. Luan, V. Y. Pan, CUR LRA at Sublinear Cost Based on Volume Maximization, LNCS 11989, In Book: *Mathematical Aspects of Computer and Information Sciences (MACIS 2019)*, D. Salmanig et al (Eds.), Springer Nature Switzerland AG 2020, Chapter No: 10, pages 1– 17, Chapter DOI:10.1007/978-3-030-43120-4_10

[39] E. Liberty, F. Woolfe, P.-G. Martinsson, V. Rokhlin, M. Tygert, Randomized algorithms for the low-rank approximation of matrices, *Proc. Natl. Acad. Sci. USA (PNAS)*, **104 (51)** 20167–20172 (2007) https://doi.org/10.1073/pnas.0709640104

[40] M. E. Lopes, S. Wang, M. W. Mahoney, Error Estimation for Randomized Least-Squares Algorithms via the Bootstrap, *Proceedings of the 35th International Conference on Machine Learning*, PMLR 80:3217-3226, 2018 and arXiv:1803.08021, 2018.

[41] M. W. Mahoney, Randomized Algorithms for Matrices and Data, *Foundations and Trends in Machine Learning*, NOW Publishers, **3, 2**, 2011. Preprint: arXiv:1104.5557 (2011) (Abridged version in: *Advances in Machine Learning and Data Mining for Astronomy*, edited by M. J. Way et al., pp. 647–672, 2012.)

[42] P.-G. Martinsson, Randomized methods for matrix computations. In M.W. Mahoney, J.C. Duchi, and A.C. Gilbert, editors, *The Mathematics of Data*, **25, 4**, 187 – 231. American Mathematical Society, 2018. IAS/Park City Mathematics Series, preprint in arXiv:1607.01649 (2019)

[43] M. W. Mahoney, P. Drineas, CUR matrix decompositions for improved data analysis, *Proceedings of the National Academy of Sciences*, **106 3**, 697–702, 2009.

[44] A.I. Osinsky, N. L. Zamarashkin, New Accuracy Estimates for Pseudo-skeleton Approximations of Matrices, *Russian Academy of Sciences: Doklady, Mathematics (DOKLADY AKADEMII NAUK)*, **94, 3**, 643–645, 2016.

[45] A.I. Osinsky, Rectangular Matrix Volume and Projective Volume Search Algorithms, arXiv:1809.02334, September 17, 2018.

[46] A.I. Osinsky, N. L. Zamarashkin, Pseudo-skeleton Approximations with Better Accuracy Estimates, *Linear Algebra and Its Applications*, **537**, 221–249, 2018.

[47] V. Y. Pan, Low Rank Approximation of a Matrix at Sublinear Cost, preprint in arXiv:1907.10481, 21 July 2019.

[48] R.K. Pace, R. Barry, Sparse Spatial Autoregressions, *Statistics and Probability Letters*, **33**, 291-297, 1997.

[49] V. Y. Pan, Q. Luan, Refinement of Low Rank Approximation of a Matrix at Sublinear Cost, preprint in arXiv:1906.04223 (Submitted on 10 Jun 2019).

[50] V. Y. Pan, Q. Luan, J. Svadlenka, L.Zhao, Primitive and Cynical Low Rank Approximation, Preprocessing and Extensions, preprint in arXiv 1611.01391 (Submitted on 3 November, 2016).

[51] V. Y. Pan, Q. Luan, J. Svadlenka, L. Zhao, Superfast Accurate Low Rank Approximation, preprint in arXiv:1710.07946 (Submitted on 22 October, 2017).

[52] V. Y. Pan, Q. Luan, J. Svadlenka, L. Zhao, Low Rank Approximation by Means of Subspace Sampling at Sublinear Cost, in Book:*Mathematical Aspects of Computer and Information Sciences (MACIS 2019),* D. Salmanig et al (Eds.), Chapter No: 9, pages 1–16, Springer Nature Switzerland AG 2020, //Chapter DOI:org/10.1007/978-3-030-43120-4_9 and preprint in arXiv:1906.04327 (Submitted on 10 Jun 2019).

[53] V. Y. Pan, Q. Luan, J. Svadlenka, L. Zhao, CUR Low Rank Approximation at Sublinear Cost, preprint in arXiv:1906.04112 (Submitted on 10 Jun 2019).

[54] V. Y. Pan, G. Qian Solving Homogeneous Linear Systems with Randomized Preprocessing, *Linear Algebra and Its Applications*, **432**, 3272–3318 (2010).

[55] V. Y. Pan, G. Qian, X. Yan, Random Multipliers Numerically Stabilize Gaussian and Block Gaussian Elimination: Proofs and an Extension to Low-rank Approximation, *Linear Algebra and Its Applications*, **481**, 202–234, 2015.

[56] V. Y. Pan, G. Qian, A.–L. Zheng, Randomized Preconditioning versus Pivoting, *Linear Algebra and Its Applications*, **438, 4**, 1883–1889 (2013).

[57] V. Y. Pan, G. Qian, A.–L. Zheng, Z. Chen, Matrix Computations and Polynomial Root–finding with Preprocessing, *Linear Algebra and Its Applications*, **434**, 854–879 (2011).

[58] V. Y. Pan, L. Zhao, Low-rank Approximation of a Matrix: Novel Insights, New Progress, and Extensions, Proc. of the *Eleventh International Computer Science Symposium in Russia (CSR'2016)*, (Alexander Kulikov and Gerhard Woeginger, editors), St. Petersburg, Russia, June 2016, *Lecture Notes in Computer Science (LNCS)*, **9691**, 352–366, Springer International Publishing, Switzerland (2016).

[59] V. Y. Pan, L. Zhao, Numerically Safe Gaussian Elimination with No Pivoting, *LAA*, **527**, 349–383, 2017. http://dx.doi.org/10.1016/j.laa.2017.04.007. Also preprint in arxiv 1501.05385.

[60] V. Y. Pan, L. Zhao, New Studies of Randomized Augmentation and Additive Preprocessing, *Linear Algebra and Its Applications*, **527**, 256–305, 2017.
http://dx.doi.org/10.1016/j.laa.2016.09.035. Also preprint in arxiv 1412.5864.

[61] V. Rokhlin, M. Tygert, A Fast Randomized Algorithm for Overdetermined Linear Least-squares Regression, *Proc. Natl. Acad. Sci. USA*, **105, 36**, 13212–13217, 2008.

[62] M. Rudelson, R. Vershynin, Smallest Singular Value of a Random Rectangular Matrix, *Comm. Pure Appl. Math.*, **62, 12**, 1707–1739, 2009.
https:// doi.org/10.1002/cpa.20294

[63] T. Sarlós, Improved Approximation Algorithms for Large Matrices via Random Projections, *FOCS'06*, 143–152, 2006.

[64] A. Sankar, D. Spielman, S.-H. Teng, Smoothed Analysis of the Condition Numbers and Growth Factors of Matrices, *SIAM J. Matrix Anal. Appl*, **28**, **2**, 446–476, 2006.

[65] E.E. Tyrtyshnikov, Mosaic-Skeleton Approximations, *Calcolo*, **33, 1**, 47–57, 1996.

[66] E. Tyrtyshnikov, Incomplete Cross-Approximation in the Mosaic-Skeleton Method, *Computing*, **64**, 367–380, 2000.

[67] J. A. Tropp, A. Yurtsever, M. Udell, V. Cevher, Practical Sketching Algorithms for Low-rank Matrix Approximation, *SIAM J. Matrix Anal. Appl.*, **38, 4**, 1454–1485, 2017. preprint in arXiv:1609.00048.

[68] J. A. Tropp, A. Yurtsever, M. Udell, and V. Cevher, *SIAM J. Sci. Comp,* **41, 4**, pp. A2430-A2463, 2019. doi:10.1137/18M1201068, preprint in arXiv 1902.08651.

[69] D.P. Woodruff, Sketching As a Tool for Numerical Linear Algebra, *Foundations and Trends in Theoretical Computer Science*, **10, 1–2**, 1–157, 2014.

[70] X. Ye, J. Xia, L. Ying, Analytical low-rank compression via proxy point selection, *SIAM J. Matrix Anal. Appl.*, **41** (2020), pp. 1059-1085.