An Overview of Sketching Techniques Comparison for Randomized Numerical Linear Algebra

Yuqi Liu, Leon Mikulinsky, Konstantin Zörner

October 2024

1 Introduction

Due to the ubiquity of linear algebra in applied mathematics, dimension reduction and memory saving have been perpetual topics as there is an ever-increasing demand for solving larger problems faster. However, numerical linear algebra's potential in this field has been well understood and exploited. On the other hand, randomized linear algebra (RLA) is a promising field that still has considerable space for utilization. In 1984, it was discovered that projecting onto a random basis approximately preserves pairwise distances with high probability [**beals1984conference**], thereby opening the doors to using randomized techniques to solve previously difficult or expensive problems.

Sketching – the process of reducing a standard linear algebra problem to a smaller, more tractable one by projecting the matrix of interest to a lower-dimensional subspace – forms the backbone of aplty named sketch-and-solve RLA algorithms and is thus essential to building a deeper understanding of the subject. As such, in this paper, we focus on this important building block rather than one individual example of a particular RLA routine.

A large variety of different techniques has been proposed to construct a random sketching operators. Further, different linear algebra problems lead themselves to different matrix structures, and as such, to different sketching matrix structures. Therefore, this paper focuses on investigating which sketch matrix structures are most suitable for a number of sample problems, both from the theoretical and computational perspective.

2 Literature Review

Randomized sketch-and-solve algorithms consist of two steps: first, reduce the problem to one of a smaller dimension, and secondly, apply the deterministic algorithm to the reduced problem. This survey focuses on analyzing the first step.

In the following section, we present a number of sketching matrix S architectures. In this paper, we discuss left-sketching, that is, projecting the target matrix A to SA. Unless otherwise noted, the original matrix A is $m \times n$, and the sketching operator S is $k \times m$, where $k \leq m$.

2.1 Preliminaries

Sketching matrices are linear maps defined as $(1 \pm \varepsilon)\ell_2$ embeddings

$$(1-\varepsilon)\|Ax\|_2^2 \leq \|SAx\|_2^2 \leq (1+\varepsilon)\|Ax\|_2^2 \tag{1}$$

with a certain probability depending on the structure of S, which represents how much SA deviates from being an isometry for A. The distortion ε can be explicitly calculated as follows [**magdonismail2019fastfixeddimensionl2subspa**

$$\varepsilon = \|I - (A^TA)^{-1/2}(SA)^T(SA)(A^TA)^{-1/2}\|_2 \tag{2}$$

2.2 Random Orthogonal Matrices

Arguably, one of the simplest sketch matrix types is the random orthogonal matrix, which is sometimes referred to as Haar operators. It is defined in the Johnson–Lindenstrauss lemma [**https://doi.org/10.1002/rsa.10073**; **Johnson1984ExtensionsOL**], which states the following:

Lemma 1 *For $0 < \varepsilon < 1$ and any integer n, for $k \geq 8\varepsilon^{-2}\log n$, then for any set X of n vectors in $\mathbb{R}^m$, there is a random orthogonal matrix scaled by $\sqrt{m/k}$, S, such that for all $x_i \in X$*

$$(1-\varepsilon)||x_i - x_j||_2^2 \leq ||S(x_i - x_j)||_2^2 \leq (1+\varepsilon)||x_i - x_j||_2^2$$

for all $1 \leq i, j \leq n, i \neq j$ with probability greater than or equal to $1/n$.

Naively, this allows the construction of a sketching matrix through the following algorithm:

1. Generate a random $k \times m$ matrix M whose entries are, say, i.i.d. gaussian, i.e. $(M)_{ij} \sim \mathcal{N}(0,1)$.
2. Decompose M to its QR factorization, where Q is orthogonal and R is upper triangular, and store Q.
3. Set $S = \sqrt{\frac{m}{k}}\, Q$.[1].

Unfortunately, in order to achieve low distortions, the random orthogonal matrix defined by the Johnson-Lindenstrauss lemma is impractical due to the fact that the embedding dimension is proportional to ε^{-2}, making it difficult to achieve low distortions [**martinsson2021randomizednumericallinearalgebra**]. Moreover, the sketching process itself may be rather prohbitive because it nessesitates running the QR decomposition algorithm (which takes $\mathcal{O}(mk^2)$ on a wide left-sketching matrix) as well as multiplying S and A (which takes $\mathcal{O}(mnk)$ operations).

2.3 Dense sketching operators

In dense sketching operators, we sample sketching operators with i.i.d. entries drawn from particular distributions. We introduce three kinds of operators for this survey, noting that depending on the dimensions of the problem, they may need to be rescaled to preserve isometry:

- **Rademacher sketching operators**: entries are ± 1 with equal probability.
- **Uniform sketching operators**: entries are sampled from a uniform distribution over a symmetric interval.
- **Gaussian sketching operators**: entries are sampled from a normal distribution with mean 0.

Universality principles in high-dimensional probability [**oymak2018universality**] guarantee that these sketching operators are practically equivalent [2], which is also highlighted in our results.

The intended use case for dense sketching operators is mainly sketching in the sampling regime, where the sketching operator is far smaller than the data to be sketched. When the sketching matrix is larger than the data to be sketched, these distributions are much less useful because they are more expensive to apply to dense matrices[**murray2023randomizednumericallinearalgebra**] – indeed, although for S a $k \times m$ sketching operator, the distortion for a sketched matrix $\varepsilon \in \Theta(\sqrt{n/k})$, where A is $m \times n$ [**magdonismail2019fastfixeddimensionl2subsp**

2.4 Sparse sketching operators

Sparse sketching operators are often constructed by independently generating the rows or columns of the sketching operator. Nevertheless, it should be noted that there exists another type of sparse sketching operator – the i.i.d. sparse sketching operator – constructed by randomly setting many of a dense sketching operator's

[1]This is justified by the fact that sketching matrices S are defined to preserve expectations($\mathbb{E}\|Sx\|_2^2 = \|x\|_2^2$) due to its embedding property [**Nakatsukasa2024accuraterandomizedalgorithms**]

[2]This applies to any such matrix when each of the entries are independent random variables, have mean 0 and variance 1, are drawn from a symmetric distribution, and have uniformly bounded moments [**oymak2018universality**].

entries to 0. However, because of their random structure, their theoretical guarantees are not as robust as the aforementioned row-by-row or column-by-column sketching operator [**murray2023randomizednumericallinearalgebra**].

A prototypical example of a sparse sketching operator is the Clarkson-Woodruff transform (CWT), also known as the CountSketch matrix [**10.1145/3019134**]. This matrix is generated by choosing one element in each of its columns to be equal to ± 1 with equal probability, and setting the rest to 0. Using this as a sketching matrix, we have that the distortion $\varepsilon \in \Theta(\sqrt{n^2/k})$ [**magdonismail2019fastfixeddimensionl2subspace**].

This can be generalized to a Sparse Sign Embedding (SSE) [**9414030**], which can contain more than one non-zero entry per column. Namely, for a $k \times m$ SSE matrix S and parameter $p \in (0, 1)$, it satisfies

$$S = \alpha s_{ij}, \qquad s_{ij} \sim \begin{cases} +1 \text{ with probability } p/2 \\ -1 \text{ with probability } p/2 \\ 0 \quad \text{otherwise} \end{cases} \tag{3}$$

where $\alpha = \frac{1}{kp}$ is a normalizing factor included to ensure S has an expectation of I_m [**tropp˙2023˙0na16-j0x38**].

For S $k \times m$, $p \in (0, 1)$, C_1, C_2 positive constants and the following condition:

$$k \gtrapprox C_1(d + \log m)\log d, \qquad p \gtrapprox C_2 \frac{\log d}{k} \tag{4}$$

then with high probability, for any d-dimensional subspace, S is an embedding with constant distortion $\frac{1}{2}$ [**tropp˙2023˙0na16-j0x38**].

These can also be generated by specifying a sparsity parameter ζ to represent the number of non-zero elements per column. Analogously, it has been shown that a sparse sign matrix serves as a subspace embedding with high probability with constant distortion for an arbitrary l-dimensional subspace of $\mathbb{R}^m$ when the embedding dimension grows $\mathcal{O}(d \log d)$ and the sparsity parameter ζ as $\mathcal{O}(\log d)$ [**martinsson2021randomizednumericallinearalg**

2.5 Subsampled Random Trigonometric Transformations

Fast trigonometric transforms are orthogonal or unitary operators that take m-vectors to m-vectors in $\mathcal{O}(m \log m)$ time or better. In randomized algorithms, we treasure their ability to map inputs that lack periodic structure to dense outputs, and they are referred to as *subsampled randomized fast trig transforms.*

These sketching operators are defined as follows [**halko2010findingstructurerandomnessprobabilistic**]:

$$S = \sqrt{\frac{m}{k}} RTD$$

Where D is a $m \times m$ diagonal matrix whose diagonal entries are uniformly distributed around the unit circle in the complex case (and in the real case, ± 1), T is a trigonometric transform, and $R \in \mathbb{R}^{k \times m}$ is a sampling matrix, selecting k rows from the $m \times m$ matrix it multiplies.

Depending on the initial data or other problem parameters, T can be a discrete Fourier transform such that $(T)_{jk} = m^{-\frac{1}{2}} e^{-\frac{2\pi i}{m}(j-1)(k-1)}$ for $j, k = 1, 2, \ldots, m$, the similarly defined discrete sine or cosine transforms, or the discrete Hadamard transform, where $T = H_m$ for $m = 2^n$ and:

$$H_m = \begin{bmatrix} H_{m/2} & H_{m/2} \\ H_{m/2} & -H_{m/2} \end{bmatrix}, \qquad H_2 = \begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix}$$

Thus, the Hadamard transform functions as a sort of analoughe for the Fourier transform for real data, having a distortion $\varepsilon \in \Theta(\sqrt{(n \log n)/d})$ [**magdonismail2019fastfixeddimensionl2subspace**]. It should be noted that although S can be represented as a nested series of matrix multiplications, in practice, the sketching operator can be applied to each column vector x of the initial matrix A by scaling (multiplication by D, applying the trigonometric transform like the DFT (multiplication by T), and then randomly sampling the result (multiplication by R), thereby reducing the sketching operation to be of order at most $\mathcal{O}(mk \log m)$ for each column vector. For all practical applications, this is how the algorithm is implemented.

For a positive constant C and for $k \gtrapprox C(d + \log m)\log d$, then with high probability, for an d-dimensional subspace, the SRFT matrix S is a subspace embedding with distortion $\frac{1}{2}$[**tropp˙2023˙0na16-j0x38**].

Curiously, although theoretically SRTTs need the embedding dimension to grow as $\mathcal{O}(d \log d)$ as above, it is often sufficient to to choose an embedding dimension of only $\mathcal{O}(d)$ [**martinsson2021randomizednumericallinearalgebra**

2.6 Properties of Sketching Matrices and Sketch Quality

Now that we have introduced all of the sketching methods that we will use we can look back at the two properties (**??**) and (**??**) introduced in Section **??**. In practice, it oftentimes suffices to require a relaxed version of property (**??**), namely it suffices if a sketching matrix preserves relative norms [**murray2023randomizednumericallinearalg**
i.e.

$$\frac{\|Su\|_2}{\|Su\|_2} \approx \frac{\|u\|_2}{\|v\|_2} \quad \text{for } u, v \in \text{col}\{A\}. \tag{5}$$

To illustrate that the introduced sketching methods fulfill this property we depict how they impact the relative norm in Figure **??** along side the computed distortion of some of the sketching methods for a matrix $A \in \mathbb{R}^{1024\times 50}$ with entries i.i.d. sampled form a standard normal distribution.

figures/sketching_properties/relative_spread_box_plots34069384.pdf

figures/sketching_properties/sketch_size_vs_distortion3406

Figure 1: Illustration of the embedding property for different sketching methods (left). Distortion for a matrix $A \in \mathbb{R}^{1024\times 50}$ with i.i.d. standard normal entries for different sketching methods (right).

3 Test Problems

3.1 Low-Rank Approximation

The low-rank approximation problem could be stated as follows:

Given an $m \times n$ matrix A, find a matrix A_k such that $\text{rank}(A_k) = k \ll \min(m, n)$ *and* $\|A - A_k\|$ *is minimized.*

There are two primary approaches to solving this problem that are best exemplified through the following algorithms:

- **Singular Value Decomposition**: Factorizing the target matrix as $A = U\Sigma V^T$ with U, V orthogonal and Σ diagonal. By denoting u_i and v_i to be the column vectors of U and V respectively and $\Sigma = \text{diag}(\sigma_1, \ldots, \sigma_n)$ and using the convention that $\sigma_1 \geq \cdots \geq \sigma_n = 0$, we can also write A as follows $A = \sum_{i=1}^{n} \sigma_i u_i v_i^T$ and form A_k by truncating the sum at some $k \leq n$. This approach, which gives the best possible rank k approximation by the Eckart-Young-Mirsky theorem, is often computationally infeasible. Similar algorithms can be used if A has a special structure, such as symmetry.

- CUR **decomposition**: For A an $m \times n$ matrix, the CUR decompositon is formed by selecting small subsets of A's columns in the $m \times k$ matrix C, rows in $k \times n$ matrix R, and linking the two using $k \times k$ matrix U such that $A_k = CUR$. Serving as a representative of submatrix-oriented decompositions, these algorithms, though more prone to numerical instability, can require less storage to compute than SVD-like decompositions, especially when A is sparse.

Nevertheless, using randomized techniques on the low-rank approximation problem can lead to improved efficiency and stability, if the right sketching operator is chosen, especially for large matrices target matrices where these algorithms would prove otherwise infeasible (e.g. the SVD decomposition taking $\mathcal{O}(mn^2)$).

Before we start our low-rank approximation, we first introduce one of the building blocks for low-rank approximations, which is called QB decomposition; here we borrow the practice taken in [**yu2018efficientrandomizedalgorith**
as a representative. See algorithm **??** as a reference. And the

Algorithm 1 The basic QB algorithm to solve rangefinder problem

Input:A,k,s
Output:Q,B

```
1: $\Omega = randn(n, k + s)$
2: $Q = orth(A\Omega)$
3: $B = Q^T A$
```

Algorithm 2 Randomized singular value decomposition(RSVD)

Input:A,k,s
Output:Q,B

```
1: $\Omega = randn(n, k + s)$
2: $Q = orth(A\Omega)$
3: $B = Q^T A$
```

After QB, we could do all sorts of deterministic factorization low-rank approximation algorithms on matrix B [**halko2010findingstructurerandomnessprobabilistic**].

Other than sketching matrices, there are many other questions that need to be considered.

- How do we choose the subspace dimension for the random matrix? The theory[**halko2011finding**] indicates that we can select the subspace dimension l to be just slightly larger than the rank r of the matrix ,say, $l = r + p$ where $p = 5$ or $p = 10$, where the value p is called the oversampling parameter.

- Matrix multiplication Matrix multiplication is a highly optimized primitive on most computer systems.

- Powering Powering iteration could help matrices without a rapidly decaying spectrum. The powering process goes as follows. Let $B \in F^{m\times n}$ be a fixed input matrix, and let q be a natural number. Let $\Omega \in F^{m\times l}$ be a random test matrix. we form the sample matrix

$$Y = (BB^*)^q \Omega$$

by repeated multiplication.

- orthongonalization The columns of the matrix Y tend to be strongly aligned, so it is important to use a numerically stable orthogonalization procedure such as Householder reflectors, double Gram-Schmidt, rank-revealing QR or TSQR algorithm [**DBLP:journals/corr/abs-0806-2159**].

3.2 Overdetermined Least Squares

The overdetermined least squares problem is defined as follows:

For $A \in \mathbb{R}^{m\times n}, x \in \mathbb{R}^n, b \in \mathbb{R}^m$ *such that* $m > n$*, minimize* $\|Ax - b\|_2^2$.

This can be solved by a number of deterministic algorithms with various trade-offs between speed, accuracy, and stability, including:

- **Normal equations**: Conceptually the simplest, it solves the problem by letting $x = (A^TA)^{-1}A^Tb$. Despite requiring the least amount of floating point operations of all the following methods, this method method is not very suitable for practical applications because it is unstable for poorly-conditioned A.
- QR **decomposition**: This method performs the following factorization: $A = QR, Q \in \mathbb{R}^{m\times n}, R \in \mathbb{R}^{n\times n}$ such that Q is orthogonal and R is upper triangular for the solution $x = R^{-1}Q^T$. QR factorization can be achieved through a number of algorithms such as Gram-Schmidt or modified Gram-Schmidt, or those which incrementally construct the result using Householder reflections or Givens rotations. In practice, while all of these algorithms have complexity $\mathcal{O}(mn^2)$, Householder rotations need the least flops among all four, while still being stable and preserving the orthogonality of Q's columns.
- **Singular Value Decomposition**: This method performs the following factorization $A = U\Sigma V^T$, where U, V are orthogonal and Σ is diagonal. The SVD van be used to solve the problem by setting $x = V\Sigma^+U^Tb$ where Σ^+ is the psuedoinverse of Σ. While the most numerically stable, the SVD takes $\mathcal{O}(mn^2)$, not to mention the matrix multiplication required to form x.

However, for m large, the latter two algorithms may become impractically slow.

4 Experiments

4.1 Datasets

For our experiments we use both real life data and synthetic data. We use the dataset from MedMNIST[**medmnistv1**; **medmnistv2**]. In our experiments, we use the dataset of breast, which consists of three parts: training dataset, validation dataset, and test dataset. We just use the training dataset here which has 546 data points, see https://zenodo.org/records/10519652. Another thing that is worth attention is that the data is almost full-rank. It is also worth attention that most of the data points are very ill-conditioned. Here we examine some basic statistical descriptions about condition numbers. See Figure **??** as a reference. In addition to that, we use scikit-learn's dataset on Californian housing[3] , which is suitable for least squares. As the dataset contains 20640 data points we sample a smaller, easier to handle number of 1024 data points. Our synthetically generated data are split into three different categories. First, we use matrices A with all entries sampled i.i.d. from a standard normal distribution, these behave very numerically very nicely and serve as a good basecase. Second we use matrices with a singular values that span a wide range, causing a very high condition number. In particular, we select $A = U\Sigma V^T$ with U, V orthogonal and Σ diagonal with $\sigma_{ii} = e^{k_i}$, where k_i are equidistantly spread in $\{-10, 10\}$. Last, we utilize multicolinear matrices that we generate by sampling a vector a from a i.i.d. standard normal distribution and then setting A's i'th column to $a_i = a + 10^{-6}\theta_i$ where θ_i are also standard normal random vectors with independent components.

4.2 Results

4.2.1 Low Rank Approximation

Our results are shown in Figure **??**, and the table containing concrete numbers is shown in **??**. We also conducted numerical experiments solving the low-rank approximation problem.The result is shown in **??**

[3] `https://scikit-learn.org/1.5/modules/generated/sklearn.datasets.fetch_california_housing.html`

figures/boxplot_BREASTMNIST.png

(a) Randomized SVD results on BreastMMNIST DATASET

splines for breastmnist.png

(b) splines for numerical experiments

Sketching Operators	Uniform	Gaussian	Rademacher	JLT	CWT
Mean	0.001267	**0.001257**	0.001265	0.001274	0.002295
std	0.001146	**0.001111**	0.001123	0.001154	0.002231
min	0.000020	0.000019	0.000019	0.000019	0.000019
25%	0.0000513	0.000524	0.000532	0.000513	0.000874
50%	0.000974	0.000972	0.000971	0.000980	0.0001709
75%	0.001594	0.001600	0.001633	0.001619	0.002860
max	0.011209	0.011091	0.010804	0.010772	0.015175

Table 1: The Performance of Different Sketching Matrices on breastMNIST

4.2.2 Least Squares

We solved the least squares problem $\min_x \|Ax - b\|_2$ for the three types of matrices described in Section **??** using all sketching methods introduced so far using both the QR and the SVD algortithm as outlined in Section **??**. For all choices of A we chose b as a standard normal random vector with independent components. We found both methods to yield the same accuracy so we only depict the results for QR here. In order to measure the accuracy of our the different sketching methods we used the following two metrics. First, the relative norm of the residual, i.e.,

$$\frac{\|Ax_{\text{opt}} - b\|_2 - \|Ax_{\text{sketch}} - b\|_2}{\|Ax_{\text{opt}} - b\|_2},$$

and second, the relative error of the found x_{sketch}, i.e.,

$$\frac{\|x_{\text{opt}} - x_{\text{sketch}}\|_2}{\|x_{\text{opt}}\|_2}.$$

repeating these experiments for many different sizes of A yields genuinely similar behavior. Thus, in Figure **??** we only chose to depict the results for $A \in \mathbb{R}^{256\times 20}$.

We repeated the same process for the Californian housing dataset introduced in Section **??**. Here we found the same behavior for the orthogonal and dense sketching methods. Although and SEE still return good results we observe that CWT's accuracy has significantly gone down, similar to the observations we made for low-rank approximation in Section **??**. Furthermore, we found that all trigonometric transforms performed much worse on this data set, achieving relative errors for the residual of around 2 across all tested sketching sizes.

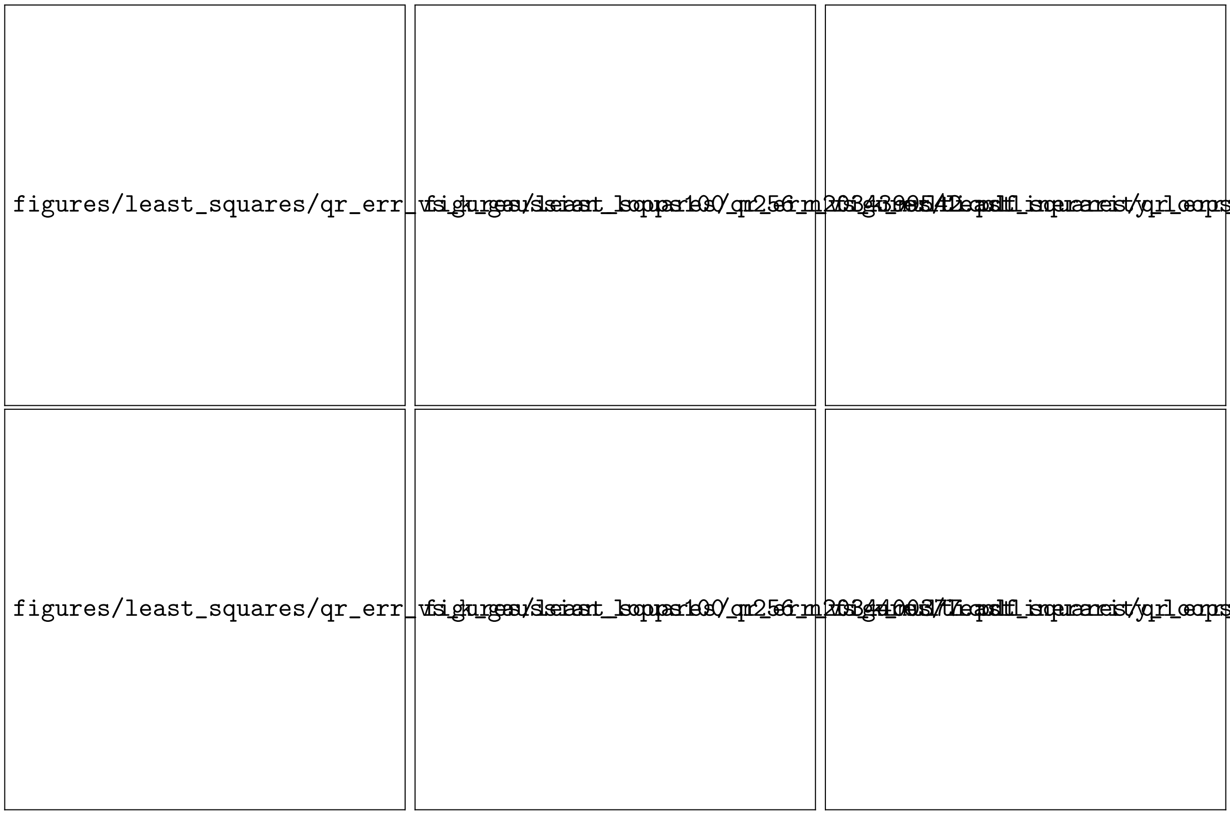

Figure 3: Accuracy of least squares using different sketching methods and different matrices A for i.i.d. Gaussian b averaged over 100 runs each. From left to right A is chosen to be i.i.d. Gaussian, be multicolinear, and have a spectrum spreading 20 orders of magnitude. The relative error of of the residual is depicted at the top and the relative error in x at the bottom.

5 discussion

Our gathered data presented in Section **??** enables us to make the following observations and allows us to draw some interesting conclusions. For one we were able to empirically show that all dense i.i.d. sketching operators behave equivalently as proposed in Section **??**. This can be clearly seen in Figure **??** where they all cluster achieving similar accuracies regardless of the sketching size k. Similarly, the figure suggests that the same might be true for the subsampled random trigonometric transformations, which cluster in a similar manner.

Next, we conclude, that distortion presents a good measure to indicate the quality of a sketching method (at least for least squares). This can be seen as the clusters found in the distortion plot of Figure **??**) are identical to those found in Figure **??**.

Interestingly, we find that the sparse operators do not form as distinct clusters as the before mentioned groups. For the numerically well behaved Matrix A with i.i.d. standard normal entries we see that they perform just as well as the dense operators. In this case they certainly favorable as their computational cost is $O(nnz)$ compared to the $O(n^3)$ of the dense operators. We observe the same for the multicolinear matrices, however, when it comes to the

- Distortion is a good measure to indicate the quality of a sketching method (at least for least squares). This can be seen as the clusters found in the distortion plot of Figure **??**) are identical to those found in Figure **??**.
- For all different types of matrices used to test least squares, we found the same clusters of accuracies. 1. Orthogonal, 2. Dense & Sparse, 3. SRFTs. Since sparse and dense operators yield similar results, sparse operators are clearly favorable.

- For numerically nice matrices A sparse and dense sketching matrices behave alike, clearly favoring sparse matrices. However, for more complicated matrices such as in the third column of Figure **??**, sparse sketching matrices struggle to capture A accurately, especially when the number of non-zero entries is low as for WCT.
- jlt outperformce the other sparse operators and still works well for badly conditioned A See and Figure **??**
- for given dataset for low rank approximation, k = 300 seems to be sweet spot? (Check whether we can find theoretical guarantees that this is just the right size for k)
- TODO: Change low rank approximation results plot to have a logarithmic y-axis