Journal of Computational Physics 447 (2021) 110686

Contents lists available at ScienceDirect

Journal of Computational Physics

www.elsevier.com/locate/jcp

Spectral estimation from simulations via sketching

Zhishen Huang, Stephen Becker *

Department of Applied Mathematics, University of Colorado Boulder, Boulder, CO, 80309, United States of America

ARTICLE INFO

Article history:
Available online 9 September 2021

Keywords:
Autocorrelation
Power spectral density
Molecular dynamics
Sketching
Randomized linear algebra

ABSTRACT

Sketching is a stochastic dimension reduction method that preserves geometric structures of data and has applications in high-dimensional regression, low rank approximation and graph sparsification. In this work, we show that sketching can be used to compress simulation data and still accurately estimate time autocorrelation and power spectral density. For a given compression ratio, the accuracy is much higher than using previously known methods. In addition to providing theoretical guarantees, we apply sketching to a molecular dynamics simulation of methanol and find that the estimate of spectral density is 90% accurate using only 10% of the data.

Large-scale computer simulations are a common tool in many disciplines like astrophysics, cosmology, fluid dynamics, computational chemistry, meteorology and oceanography, to name just a few. In many of these fields, a key goal of the simulation is an estimate of the power spectral density (or equivalently autocorrelation) of some dynamic or thermodynamic state variable or derived function.

Computing a full autocorrelation becomes prohibitively expensive for large-scale simulations since it requires storing the entire dataset in memory. The textbook strategy to combat this problem is to subsample in time, often with clever logarithmic or multi-level spacing strategies [1]. Other simple solutions subsample particles or grid points, or both time and particles/points. Unfortunately, these *ad hoc* methods lack rigorous performance guarantees and can have arbitrarily large error. This article shows how to leverage results from the new field of *randomized linear algebra* to derive subsampling methods that work better in practice and have theoretical guarantees on the accuracy. These new subsampling methods, known as *sketching* methods, essentially exploit the fact that multiplying by a multivariate Gaussian to do compression ensures no worst-case inputs; in comparison, simple subsampling methods do well on some inputs but catastrophically bad on other inputs. Section 1 gives a toy example of this, and the rest of the paper shows how this applies to sampling data for spectral estimation.

Contributions This paper shows how to use existing results from randomized linear algebra results in the context of estimating autocorrelations and power spectral densities. Specifically, we

1. show that the autocorrelation and power spectral density are simple functions of the covariance matrix;
2. convert existing results on covariance matrix estimation to results on estimating autocorrelation and power spectral density; and

* Corresponding author.
E-mail addresses: zhishen.huang@colorado.edu (Z. Huang), stephen.becker@colorado.edu (S. Becker).

https://doi.org/10.1016/j.jcp.2021.110686
0021-9991/

3. numerically demonstrate that the resulting sketching methods are significantly more accurate than baseline methods when applied to the problem of autocorrelation and power spectral density estimation in a typical molecular dynamic simulation.

Throughout the paper, we pay attention to computation and communication costs. In particular, the sketches are linear operators and can be applied to a data stream, so they can be applied during a simulation with negligible memory overhead and in a reasonable time. Our methods are also simple to implement. Indeed, a reason that more sophisticated sampling schemes are not used in practice may be due to the cumbersome book-keeping required for normalizations, but we review a simple trick to deal with this (Remark 7), and other than sampling, our methods do not require any "on-the-fly" computation, as the estimates are formed in post-processing.

Background Spectral estimation arises in molecular dynamic (MD) simulations based on time-dependent density functional theory (TDDFT) [2], which is a prominent methodology for electronic structure calculations. Depending on the original variable (position, velocity, dipole-moment, etc.), applications of spectral estimation in TDDFT include calculating vibrational or rotational modes (as used in infrared and Raman spectroscopy) [3], optical absorption spectra [4], and circular dichroism spectra [5]. Many of these quantities can be experimentally measured, so the spectrum can be used to verify that the simulation matches with reality, as well as predicting properties of novel materials.

Similarly, temporal autocorrelations may be computed during numerical solutions of partial differential equations (PDEs). For one example, in fluid dynamics, the autocorrelations computed via direct numerical simulation of the Navier-Stokes equations can be used to validate large-eddy simulation models [6]. Another example is oceanography where modern simulation codes rely on multi-scale numerical methods that cannot fully resolve the smallest scales, and so use stochastic models to inform the simulation [7,8]. The stochastic process can be constrained to conform to a given autocorrelation function.

MD simulations operate on particles, while standard numerical methods for PDEs operate on (possibly unstructured) grids and elements. In both cases, the exact sample time-autocorrelation function can be computed provided the data (particles or grid points, at all times) is stored. Due to advances in computing power and algorithm design, it is now feasible to run extremely large simulations. A consequence of this is that many large-scale simulations generate more data than can be stored. As an example, running the billion-atom Lennard Jones benchmark on the MD `LAMMPS` software [9] for the equivalent of 1 ns of simulation time on argon atoms [10] takes 4.9 hours on a 288 node GPU computer from 2012 [11], making it a modest large-scale computation. Storing the 6 coordinates of position and velocity in double precision for the 10^5 timesteps would require 4.26 PB, well beyond a typical high-end cluster disk quota of 150 TB. Longer simulations, or simulations of molecules, only exacerbate the problem. Standard compression methods for scientific data, like `fpzip` [12] and `ZFP` [13], improve this by one or two orders of magnitude at best [14].

1. Sketching

Sketching is used to reduce dimensionality from N dimensions to some $m \ll N$. A family of sketches is a probability distribution on the set of real or complex $m \times N$ matrices such that if $\mathbf{\Omega}$ is drawn from this family, for any fixed vectors $\mathbf{v}, \mathbf{w} \in \mathbb{R}^N$, then $\|\mathbf{\Omega v} - \mathbf{\Omega w}\|_2 \approx \|\mathbf{v} - \mathbf{w}\|_2$ with high probability. Hence the sketch preserves distances, and by the polarization formula, preserves inner products as well. The core ideas behind sketching have been in place since the 1980s, and were well-known in theoretical computer science literature, but the field has expanded since 2005 as many applications in scientific computing were developed. In particular, sketching is often used to efficiently find solutions of large least-square regression problems [15–20], and to determine the row and column space of large matrices for low-rank matrix decomposition [21–23].

Formally, a probability distribution on $m \times N$ matrices is a *Johnson-Lindenstrauss Transform* with parameters ε, δ and d if for any fixed set of d vectors $\{\mathbf{v}_i\}_{i=1}^d \subset \mathbb{R}^N$, if $\mathbf{\Omega}$ is drawn from this distribution, then with probability at least $1 - \delta$ it holds that

$$(1-\varepsilon)\|\mathbf{v}_i - \mathbf{v}_j\|_2^2 \le \|\mathbf{\Omega v}_i - \mathbf{\Omega v}_j\|_2^2 \le (1+\varepsilon)\|\mathbf{v}_i - \mathbf{v}_j\|_2^2$$

for all $i, j \in \{1, \dots, d\}$. When no confusion arises, it is common to not distinguish between the random variable and the distribution, and write $\mathbf{\Omega} \in \mathrm{JLT}(\varepsilon, \delta, d)$ to encode the notion. The name Johnson-Lindenstrauss Transform honors Johnson and Lindenstrauss' well-known result which shows that such distributions exist for $m = \mathcal{O}(\varepsilon^{-2}\log(d))$ [24].

Intuition To gain insight, consider the case when $\mathbf{\Omega} \in \mathbb{R}^{1\times N}$ is a sketch that compresses $\mathbf{v} \in \mathbb{R}^N$ to a single number, and without loss of generality, let $\|\mathbf{v}\|_2 = 1$. All sketches we consider will be unbiased, meaning $\mathbb{E}\,\mathbf{\Omega}^T\mathbf{\Omega} = I_{N\times N}$ where I is the identity matrix. We wish to preserve norm, so we look at $\|\mathbf{\Omega v}\|_2^2$, or equivalently $(\mathbf{\Omega v})^2$ when $m = 1$. Then any unbiased sketch has $\mathbb{E}\,(\mathbf{\Omega v})^2 = 1$.

A natural approach to reducing dimension is simple subsampling, meaning that each entry has an equal chance of being selected. Simple subsampling can be written as a sketch by defining $\mathbf{\Omega} = \sqrt{N}\mathbf{e}_i^\top$ where $\mathbf{e}_i$ is the i^{th} canonical basis vector in $\mathbb{R}^N$, and i is chosen uniformly from $\{1, \dots, N\}$; one can easily show this is unbiased. In the lucky event that the input

Table 1
Compressed dimension requirement for JLTs.

Method	Compressed dimension m
Gaussian [27]	$\mathcal{O}(\varepsilon^{-2}\log(d/\delta))$
Haar [35]	$\mathcal{O}(\varepsilon^{-2}\log(d/\delta))$
FJLT [36], [Proposition 3.9]	$\mathcal{O}\left(\varepsilon^{-2}\log(Nd/\delta)\log(d/\delta)\right)$

$\mathbf{v}$ has weight evenly distributed over all coordinates, such that $|v_j| = N^{-1/2}$ for all $j = 1, \ldots, N$, then this is a good sketch, since the variance is $\mathbb{V}\mathrm{ar}((\mathbf{\Omega v})^2) = 0$. However, if the input is $\mathbf{v} = \mathbf{e}_k$ for any fixed k, then an elementary calculation shows that $\mathbb{V}\mathrm{ar}((\mathbf{\Omega v})^2) = N - 1$, which in high dimensions is too large to be useful.

In contrast, the classic example of a *good* sketch is an appropriately scaled Gaussian matrix with independent entries. For this sketch, define $\mathbf{\Omega}$ as $1 \times N$ independent standard normal random variables, then $\mathbf{\Omega}$ is also an unbiased sketch, and furthermore $\mathbb{V}\mathrm{ar}((\mathbf{\Omega v})^2) = 2$ independent of the fixed vector $\mathbf{v}$. In contrast, the variance of the simple subsampling sketch ranges between $[0, N-1]$ depending on $\mathbf{v}$. The Gaussian sketch is not always more efficient than the subsampling sketch, but it is never much worse, and sometimes it is better by a factor of N.

Types of sketches In this work we consider the following three types of distributions of sketching matrices $\mathbf{\Omega}$ (Matlab code available via [25]; some Python implementations are part of the `random_projection` module of scikit learn):

Gaussian sketch Each entry of $\mathbf{\Omega}$ is independently drawn from the scaled normal distribution $\mathcal{N}(0, \frac{1}{m})$.

Haar sketch Draw $\widetilde{\mathbf{\Omega}}$ as in the Gaussian case and then define the rows of $\mathbf{\Omega}$ to be the output of Gram-Schmidt orthogonalization applied to the rows of $\widetilde{\mathbf{\Omega}}$, scaled by $\sqrt{\frac{N}{m}}$. This is equivalent to sampling the first m columns of a matrix from the Haar distribution on orthogonal matrices, and can also be computed via the `QR` factorization algorithm with post-processing [26]. This is essentially the case originally considered by Johnson and Lindenstrauss.

FJLT The Fast Johnson-Lindenstrauss Transformation (FJLT) as is usually implemented [27] is a structured matrix of the form $\mathbf{\Omega} = \sqrt{\frac{N}{m}}\mathbf{P}^\top\mathbf{HD}$ where $\mathbf{D}$ is a diagonal matrix with Rademacher random variables on the diagonal (i.e., independent, ± 1 with equal probability), $\mathbf{H}$ is a unitary or orthogonal matrix, and $\mathbf{P}^\top$ a simple subsampling matrix such that $\mathbf{P}^\top\mathbf{v}$ chooses m of the coordinates from $\mathbf{v}$ uniformly at random (with replacement), so that $\mathbf{P}$ consists of m canonical basis vectors. To be useful, each entry of $\mathbf{H}$ should be as small as possible ($\approx 1/\sqrt{N}$), and $\mathbf{H}$ should be computationally fast to apply to vector. Standard choices for $\mathbf{H}$ are the (Walsh-)Hadamard, discrete Fourier, and discrete Cosine transforms, all of which have fast implementations that take $\mathcal{O}(N\log N)$ flops to apply to a vector. Since applying $\mathbf{D}$ and $\mathbf{P}^\top$ take linear and sub-linear time, respectively, the cost of computing $\mathbf{\Omega v}$ is $\mathcal{O}(N\log N)$, better than the $\mathcal{O}(Nm)$ cost of the Gaussian and Haar sketches. The original FJLT proposed in [28] is a slight variant that uses a different sparse matrix $\mathbf{P}$.

There are other types of sketches such as the count-sketch [29], leverage-score based sketches [30], and entry-wise sampling [31,32] which can be combined with preconditioning [33]. Some of these sketches are not Johnson-Lindenstrauss transforms but are instead the related notion of subspace embeddings. See [27,30,34] for surveys on sketching literature.

Guarantees Table 1 summarizes the required compressed dimension size m for the corresponding sketching matrix to be a JLT(ε, δ, d).

The result for the FJLT holds when $\mathbf{H}$ is a Hadamard matrix, and follows from the observation that a subspace embedding with complexity that depends only logarithmically on the failure probability δ can be turned into a JLT using the union bound. When $\mathbf{H}$ is a discrete Fourier or discrete Cosine transform, similar $\mathcal{O}(\varepsilon^{-2})$ sample complexities hold (with polylog factors in d, N and δ^{-1}) by combining [37, Thm. 3.1] with [38, Thm. 12.31]. The constants hidden in the asymptotic notation are not bad. For example, for the Gaussian sketch, with $d = 10^3$ points (in arbitrary dimension N), for failure probability $\delta \le 0.1$ and error $\varepsilon \le 1/3$, the number of samples required is $m \ge 535$.

2. Approximating autocorrelation with sketching

Throughout the article, we think of the data as a signal $x(t, \varphi)$ in time t and space φ, where φ can encode a grid location or a particle number depending on the type of simulation (for space indices in dimension greater than one, we flatten the indices into a large one-dimensional list). Let t have unit spacing $\Delta T = 1$, $t \in \{1, 2, \ldots, T\}$, and let space be indexed by $\{\varphi_1, \ldots, \varphi_N\}$. We organize the data into a matrix $\mathbf{X} \in \mathbb{R}^{T\times N}$.

In what follows, we consider classical methods for estimating the autocorrelation. There are powerful alternative methods, based on parametric models — most notably, autoregressive-moving-average (ARMA) models [39]. However, these methods excel when T is small, do not clearly extend to $N > 1$, and are not natively suited to on-the-fly calculations during a simulation as they require significant post-processing and parameter tuning.

Autocorrelation and the Wiener-Khinchin theorem For a continuous signal x, the time autocorrelation function of lag τ of signal x is

$$R(\tau) = \mathbb{E}_{\varphi} \lim_{T\to\infty} \frac{1}{2T} \int_{-T}^{T} x(t,\varphi)x(t+\tau,\varphi)\,\mathrm{d}t.$$

For the corresponding discretized signal of length T, the (sample) time autocorrelation of lag τ is defined as

$$\widehat{R}_\tau[\mathbf{X}] = \frac{1}{N}\frac{1}{T-\tau}\sum_{t=1}^{T-\tau}\sum_{i=1}^{N} x(t,\varphi_i)x(t+\tau,\varphi_i) \tag{1}$$

where we change notation slightly to emphasize that this is a function of the data $\mathbf{X}$. As our goal will be to approximate the sample autocorrelation $\widehat{R}_\tau$, we drop the $\widehat{\ }$ notation for clarity and simply write R_τ.

Remark 1 *(Cross-terms).* Calculating Eq. (1) requires storing $N \times T$ parameters. If one instead computed $\sum_{t=1}^{T-\tau}\left(\sum_{i=1}^{N} x(t,\varphi_i)\right)\left(\sum_{i=1}^{N} x(t+\tau,\varphi_i)\right)$ (with appropriate normalization), then only $\mathcal{O}(T)$ storage is required, but unfortunately this is not equivalent to Eq. (1) due to the presence of the cross-terms. One way to view sketching methods is that the sketching adds in suitable randomness so that when using the $\mathcal{O}(T)$ formula, the cross-terms vanish in expectation.

Letting the shifted, unnormalized (sample) covariance matrix be $\mathbf{\Sigma} = \mathbf{X}\mathbf{X}^\top$, our first observation is that R_τ is a linear function of $\mathbf{\Sigma}$, since

$$(\mathbf{\Sigma})_{t,t'} = \sum_{i=1}^{N} x(t,\varphi_i)x(t',\varphi_i)$$

so R_τ is the scaled sum of the τ^{th} diagonal of $\mathbf{\Sigma}$, and hence we use the notation $R_\tau[\mathbf{\Sigma}]$, and also write $\mathbf{R}[\mathbf{\Sigma}] = (R_0[\mathbf{\Sigma}], R_1[\mathbf{\Sigma}], \cdots, R_{T-1}[\mathbf{\Sigma}])^\top$ when working with all T possible lags.

The time autocorrelation is often of interest itself, but it can also be used to derive the power spectral density,

$$S(\omega) = \lim_{T\to\infty} \mathbb{E}_{\varphi} \left| \frac{1}{\sqrt{2T}} \int_{-T}^{T} x(t,\varphi)\mathrm{e}^{-\mathrm{i}\omega t}\,\mathrm{d}t \right|^2.$$

If x is a wide-sense stationary random process, under certain conditions, the Wiener-Khinchin Theorem states that the spectral density is the Fourier transform of $R(\tau)$, and the discrete power spectral density can be estimated by the discrete Fourier transform of $\mathbf{R}$.

Thus both autocorrelation and power spectrum can be reduced to the problem of finding an accurate estimate of $\mathbf{\Sigma}$. Note that $\mathbf{\Sigma}$ is a $T \times T$ matrix that is impractical to store, and is used only for analysis. Our actual software implementation only needs a factored form $\mathbf{\Sigma} = \widehat{\mathbf{X}}\widehat{\mathbf{X}}^\top$ for $\widehat{\mathbf{X}} \in \mathbb{R}^{T\times m}$, and works directly with $\widehat{\mathbf{X}}$. Furthermore, due to linearity, implementations can exploit existing autocorrelation software (which typically use the fast Fourier transform to do convolutions efficiently). Specifically, if the columns of $\widehat{\mathbf{X}}$ are $\mathbf{v}_1, \ldots, \mathbf{v}_m$, then $R_\tau[\mathbf{\Sigma}] = R_\tau[\sum_{i=1}^{m} \mathbf{v}_i\mathbf{v}_i^\top] = \sum_{i=1}^{m} R_\tau[\mathbf{v}_i\mathbf{v}_i^\top]$ and $R_\tau[\mathbf{v}_i\mathbf{v}_i^\top]$ is performed implicitly via an efficient autocorrelation implementation.

In the next section, we use standard results from the sketching literature to create an estimator $\widehat{\mathbf{\Sigma}}$ and bound $\|\mathbf{\Sigma} - \widehat{\mathbf{\Sigma}}\|_F < \varepsilon$, where $\|\cdot\|_F$ denotes the Frobenius (Hilbert-Schmidt) norm. To use those results, we first show that $\mathbf{R}$ is Lipschitz continuous so that a small ε implies an accurate autocorrelation (and hence an accurate power spectrum).

Lemma 2. *Let $\mathbf{\Sigma}$ and $\widehat{\mathbf{\Sigma}}$ both be symmetric $T \times T$ matrices. Then*

$$\|\mathbf{R}[\mathbf{\Sigma}] - \mathbf{R}[\widehat{\mathbf{\Sigma}}]\|_2 \le \|\mathbf{R}[\mathbf{\Sigma}] - \mathbf{R}[\widehat{\mathbf{\Sigma}}]\|_1 \le \frac{\sqrt{1+\log T}}{N}\|\mathbf{\Sigma} - \widehat{\mathbf{\Sigma}}\|_F \tag{2}$$

$$\|\mathbf{R}[\mathbf{\Sigma}] - \mathbf{R}[\widehat{\mathbf{\Sigma}}]\|_\infty \le \frac{1}{N}\|\mathbf{\Sigma} - \widehat{\mathbf{\Sigma}}\|_F \tag{3}$$

where $\|\mathbf{R}[\mathbf{\Sigma}] - \mathbf{R}[\widehat{\mathbf{\Sigma}}]\|_1 = \sum_{\tau=0}^{T-1} \left|R_\tau[\mathbf{\Sigma}] - R_\tau[\widehat{\mathbf{\Sigma}}]\right|$, $\|\mathbf{R}[\mathbf{\Sigma}] - \mathbf{R}[\widehat{\mathbf{\Sigma}}]\|_\infty = \max_{\tau=0,\ldots,T-1} \left|R_\tau[\mathbf{\Sigma}] - R_\tau[\widehat{\mathbf{\Sigma}}]\right|$, *and* $\|\mathbf{R}[\mathbf{\Sigma}] - \mathbf{R}[\widehat{\mathbf{\Sigma}}]\|_2 = \sqrt{\sum_{\tau=0}^{T-1} \left|R_\tau[\mathbf{\Sigma}] - R_\tau[\widehat{\mathbf{\Sigma}}]\right|^2}$.

Proof. Define the difference between true covariance matrix and the estimate as $\mathbf{\Delta} = \mathbf{\Sigma} - \widehat{\mathbf{\Sigma}}$. For the ∞-norm case in Eq. (3), using linearity of $\mathbf{R}$,

$$\|\mathbf{R}[\boldsymbol{\Delta}]\|_\infty = \max_\tau \|R_\tau[\boldsymbol{\Delta}]\| = \frac{1}{N}\max_\tau \left|\frac{1}{T-\tau}\sum_{t=1}^{T-\tau}\Delta_{t,t+\tau}\right| \le \frac{1}{N}\max_{t,t'}|\Delta_{t,t'}| \le \frac{1}{N}\|\boldsymbol{\Delta}\|_F .$$

From this, we immediately have the bound $\|\mathbf{R}[\boldsymbol{\Delta}]\|_1 \le \frac{T}{N}\|\boldsymbol{\Delta}\|_F$, but this is loose, and we show below how to derive a better dependence on T:

$$\begin{aligned}
\left\|\mathbf{R}[\boldsymbol{\Sigma}] - \mathbf{R}[\widehat{\boldsymbol{\Sigma}}]\right\|_1 &= \sum_{\tau=0}^{T-1}\left|R_\tau[\Delta]\right| \le \frac{1}{N}\sum_{\tau=0}^{T-1}\frac{1}{T-\tau}\sum_{t=1}^{T-\tau}|\Delta_{t,t+\tau}| \\
&\overset{①}{\le} \frac{1}{N}\sum_{\tau=0}^{T-1}\sqrt{\frac{1}{T-\tau}\sum_{t=1}^{T-\tau}|\Delta_{t,t+\tau}|^2} \overset{②}{\le} \frac{1}{N}\sqrt{\sum_{\tau=0}^{T-1}\frac{1}{T-\tau}}\sqrt{\sum_{\tau=0}^{T-1}\sum_{t=1}^{T-\tau}|\Delta_{t,t+\tau}|^2} \\
&= \frac{1}{N}\sqrt{\sum_{\tau=1}^{T}\frac{1}{\tau}}\sqrt{\|\Delta\|_F^2 - \sum_{\substack{\alpha\in\text{lower triang.}\\ \text{off-diag elems}}}\Delta_\alpha^2} \le \frac{\sqrt{1+\log T}}{N}\|\Delta\|_F ,
\end{aligned} \tag{4}$$

where ① is due to Jensen's inequality, and ② is due to Cauchy-Schwarz.

The first inequality in Eq. (2) follows from a general property of the $\|\cdot\|_2$ and $\|\cdot\|_1$ norms. □

3. Theoretical guarantees

We give bounds on the error of autocorrelation evaluation due to sketching the rows of $\mathbf{X}$, i.e., $\widehat{\mathbf{X}}^\top = \boldsymbol{\Omega}\mathbf{X}^\top$. Each row consists of the data at a given time t, so this can be trivially implemented in a streaming fashion. The overall compression ratio is $\gamma = \frac{m}{N}$, which for a fixed m is independent of T.

Proposition 3. *For any $\varepsilon > 0$, and for a data matrix $\mathbf{X} \in \mathbb{R}^{T\times N}$, compute $\widehat{\mathbf{X}} = \mathbf{X}\boldsymbol{\Omega}^\top \in \mathbb{R}^{T\times m}$ for a sketch $\boldsymbol{\Omega}$ with enough rows m such that $\boldsymbol{\Omega} \in JLT(\varepsilon, \delta, 2T)$, and define $\boldsymbol{\Sigma} = \mathbf{X}\mathbf{X}^\top$ and $\widehat{\boldsymbol{\Sigma}} = \widehat{\mathbf{X}}\widehat{\mathbf{X}}^\top$. Then with probability at least $1-\delta$, the computed autocorrelation based solely on the data sketch satisfies the following error characterizations:*

$$\frac{\|\mathbf{R}[\widehat{\boldsymbol{\Sigma}}] - \mathbf{R}[\boldsymbol{\Sigma}]\|_2}{\|\mathbf{X}\|_F^2} \le \frac{\|\mathbf{R}[\widehat{\boldsymbol{\Sigma}}] - \mathbf{R}[\boldsymbol{\Sigma}]\|_1}{\|\mathbf{X}\|_F^2} \le \frac{\sqrt{1+\log T}}{N}\varepsilon \tag{5}$$

$$\frac{\|\mathbf{R}[\widehat{\boldsymbol{\Sigma}}] - \mathbf{R}[\boldsymbol{\Sigma}]\|_\infty}{\|\mathbf{X}\|_F^2} \le \frac{1}{N}\varepsilon. \tag{6}$$

In particular, if $\boldsymbol{\Omega}$ is a Gaussian, Haar or FJLT sketch, then $\boldsymbol{\Omega} \in JLT(\varepsilon, \delta, 2T)$ if m is chosen as in Table 1.

Proof. A standard sketching result due to Sarlós [40] gives the error bound for using JLT to estimate matrix products as the following: let $\mathbf{X} \in \mathbb{R}^{T_1\times N}$ and $\mathbf{Y} \in \mathbb{R}^{N\times T_2}$. If $\boldsymbol{\Omega}$ is a JLT$(\varepsilon, \delta, T_1+T_2)$, then

$$\mathbb{P}(\|\mathbf{X}\mathbf{Y} - \mathbf{X}\boldsymbol{\Omega}^\top\boldsymbol{\Omega}\mathbf{Y}\|_F \le \varepsilon\|\mathbf{X}\|_F\|\mathbf{Y}\|_F) \ge 1-\delta$$

Applying Lemma 2 with $\mathbf{Y} = \mathbf{X}$ gives the result immediately. □

To quantitatively characterize how the error in autocorrelation evaluation depends on the compression ratio, we have the following corollary.

Corollary 4. *Under the setting of Theorem 3, assuming the data matrix $\mathbf{X}$ has bounded entries, then the required compression ratio $\gamma = m/N$ to have $\|\mathbf{R}[\widehat{\boldsymbol{\Sigma}}] - \mathbf{R}[\boldsymbol{\Sigma}]\|_1 \le \varepsilon$ with probability greater than $1-\delta$ is $\gamma = \mathcal{O}\big(\frac{T^2\log T\log(T/\delta)}{\varepsilon^2 N}\big)$ for Gaussian or Haar matrix sketches, and $\gamma = \mathcal{O}\big(\frac{T^2\log T\log(Nd/\delta)\log(d/\delta)}{\varepsilon^2 N}\big)$ for FJLT sketches.*

Proof. For Gaussian or Haar matrix sketches as a JLT$(\widetilde{\varepsilon}, \delta, 2T)$, recall from Table 1 that the required compressed dimension $m = \mathcal{O}(\widetilde{\varepsilon}^{-2}\log(T/\delta))$. Then with probability greater than $1-\delta$, $\|\mathbf{R}[\widehat{\boldsymbol{\Sigma}}] - \mathbf{R}[\boldsymbol{\Sigma}]\|_1 \le \frac{\sqrt{1+\log T}}{N}\widetilde{\varepsilon}\|\mathbf{X}\|_F^2$ using the error characterization equation (5) in Theorem 3. Then, to ensure this ℓ_1 norm loss bound is less than some ε, the required compression ratio is $\gamma = m/N = \mathcal{O}(\widetilde{\varepsilon}^{-2}\log(T/\delta))/N = \mathcal{O}\big(\frac{T^2\log T\log(T/\delta)}{\varepsilon^2 N}\big)$, where the last equality exploits $\|\mathbf{X}\|_F^2 = \mathcal{O}(TN)$ since $\mathbf{X}$ has bounded entries. Similar arguments will give the order of the compression ratio γ for FJLT sketches. □

The corollary suggests that as the simulation time $T \to \infty$, our compression ratio grows, until at some point it is not useful. However, T should be seen as inversely proportional to the lowest desired frequency in the power spectrum, not total simulation time. For longer simulation times T_{long}, the data should be blocked into B matrices $\mathbf{X}_{(1)}, \ldots, \mathbf{X}_{(B)}$, each of size $T = T_{\text{long}}/B$, and then form $\boldsymbol{\Sigma} = \frac{1}{B}\sum_{b=1}^{B} \mathbf{X}_{(b)}\mathbf{X}_{(b)}^{\top}$, and similarly for $\widehat{\boldsymbol{\Sigma}}$, with fresh sketches $\boldsymbol{\Omega}_{(b)}$ drawn for each block. If for some reason one needed arbitrarily low frequencies, and wanted the sample time autocorrelation to converge to the true time autocorrelation, then choose $B \propto \sqrt{T_{\text{long}}}$ [41,42], but otherwise choose $B \propto T_{\text{long}}$ and hence the block size T is constant.

Thus given a fixed time T, the corollary says that $\gamma \approx \mathcal{O}(1/N)$ and hence as the amount of data increases, the compression savings are great; in fact, the absolute number of measurements m is independent of the spatial size N for Gaussian and Haar sketches, and only logarithmically dependent on N for the FJLT sketch. For example, this means that if one increases the resolution of a grid or mesh, the amount of data needed to be stored using a Gaussian sketch actually stays constant. This holds not just for 1D grids, but 3D or any dimension grids.

We also note that the matrix $\boldsymbol{\Sigma}$ need not represent all grid points or particles, but could instead represent a subset of grid points or particles, and then the calculations are done independently for each $\boldsymbol{\Sigma}$ and averaged in the end. This may be beneficial in parallel and distributed computing, where each $\boldsymbol{\Sigma}$ might represent just the spatial locations stored in local memory.

Remark 5 *(Error for the power spectral density).* Any bound on $\|\mathbf{R}[\widehat{\boldsymbol{\Sigma}}] - \mathbf{R}[\boldsymbol{\Sigma}]\|_2$ immediately translates to a bound on the error of the discrete power spectral density in the Euclidean norm, since the discrete power spectral density is the discrete Fourier transform (DFT) of autocorrelation, and the DFT operator is unitary.

4. Numerical experiments

The pseudo-code for the proposed sketching algorithm is in Algorithm 1. It exploits existing fast implementations of sample autocorrelation, e.g., `xcorr` in Matlab or `numpy.correlate` in Python. We use Matlab indexing notation, with $\mathbf{X}(:, j)$ meaning the j^{th} column of $\mathbf{X}$, and $\mathbf{X}(i, :)$ the i^{th} row. For our data, the mean was near zero and was not subtracted explicitly. Bartlett windowing [42] was performed to reduce spectral leakage whenever $B > 1$.

Algorithm 1 Sketching for autocorrelation and power density estimation. Requires existing implementation of `autocorr`.

Require: Simulation time T_{long}, number of blocks B, compression size m
$T = T_{\text{long}}/B$
for $b = 0, 1, 2, \ldots, B-1$ **do**
  Draw $\boldsymbol{\Omega} \in \mathbb{R}^{m \times N}$ ▷ One of the sketching operators from §1
  Initialize empty array $\widehat{\mathbf{X}} \in \mathbb{R}^{T \times m}$
  for $t = 1, 2, \ldots, T$ **do**
    Generate data $\mathbf{x}^{\top} \in \mathbb{R}^{1 \times N}$ according to simulation (at time $t + bB$); equivalent to *row* $\mathbf{X}(t, :)$
    Compute and store row $\widehat{\mathbf{X}}(t, :) = (\boldsymbol{\Omega}\mathbf{x})^{\top}$
    Discard $\mathbf{x}$ from memory
  end for
  Compute $\mathbf{R}_{(b)} = \frac{1}{N}\sum_{i=1}^{m} \texttt{autocorr}(\widehat{\mathbf{X}}(:, i))$
end for
$\mathbf{R} = \frac{1}{B}\sum_{b=0}^{B-1} \mathbf{R}_{(b)}$ ▷ autocorrelation
$S = \texttt{FFT}(\mathbf{R})$ ▷ power spectral density

Remark 6. Conceptually, the algorithm forms $\widehat{\mathbf{X}} = \mathbf{X}\boldsymbol{\Omega}$, though the full-size data matrix $\mathbf{X}$ is never actually formed, as $\widehat{\mathbf{X}}$ is built up row-by-row (and old rows of $\mathbf{X}$ are discarded). Similarly, the estimated covariance matrix $\widehat{\boldsymbol{\Sigma}}$, which is introduced for discussion on theoretical properties of sketching methods, is never explicitly constructed for computation, as discussed in Section 2.

4.1. Baseline methods

Many existing algorithms for computing autocorrelation require complete data, such as the utility routines provided with the popular MD simulator `LAMMPS` [9], so we do not compare with these since they work with the full data. Among subsampling approaches, we compare with the following three types of subsampling (recall the data matrix is structured as $\mathbf{X} \in \mathbb{R}^{T \times N}$, where T is the total length of time and N is the total number of particles or grid size), all of which sample with replacement:

Time dimension compression Given a compression ratio γ, sample time points $\mathcal{I} \subset \{1, \ldots, T\}$ with size $|\mathcal{I}| = \lceil \gamma T \rceil$ (where $\lceil a \rceil$ rounds a up to the nearest integer) by selecting **rows** from the data matrix $\mathbf{X}$. The natural unbiased estimator for the autocorrelation $R_\tau[\mathbf{X}]$ is

$$\frac{1}{N}\frac{1}{z_\tau^{\mathcal{I}}} \sum_{t | t, t+\tau \in \mathcal{I}} \sum_{i=1}^{N} \mathbf{X}(t, i)\mathbf{X}(t + \tau, i) \tag{7}$$

where $z_\tau^{\mathcal{I}}$ is a normalization coefficient that is the number of t such that $t \in \mathcal{I}$ and $t+\tau \in \mathcal{I}$ (for full sampling, this is $z_\tau^{\mathcal{I}} = T-\tau$ as in (1)). Efficient computation of this autocorrelation estimate is discussed in Remark 7. When the index $\mathcal{I}$ is sufficiently small, not all lags τ will have an estimate, thus making computation of the PSD unclear. In these cases, we interpolate the missing lag values using cubic splines.

There are several common choices for $\mathcal{I}$:

1. Choosing $\mathcal{I}$ (pseudo-)randomly according to the uniform distribution. This is the method we use in the experiments unless otherwise noted, as it has the best performance among these types of methods.
2. Choosing $\mathcal{I}$ via a power-series sampling scheme that is common in simulation of polar liquids (where $R_\tau[\mathbf{X}]$ is only needed for short lags τ due to the rapid decorrelation). Given a block length k, let $\mathcal{I}_0 = \{1, 2, 4, 8, \ldots, 2^k\}$, and then the index set $\mathcal{I}$ is divided into blocks $\mathcal{I} = \mathcal{I}_0 \cup (2^k + \mathcal{I}_0) \cup (2^{k+1} + \mathcal{I}_0) \cup \ldots$. This scheme is intended to give dense sampling for low lags, and some sampling for higher lags while still allowing for reasonable book-keeping due to its structured nature. See Fig. 1 for a comparison of this scheme with random sampling; it generally underperforms random sampling, so we do not present further comparisons.
3. Sparse ruler sampling. As shown in Fig. 1, the power-series scheme does not generate all possible lags. Sampling schemes that do generate all possible lags (up to some point) are known as *rulers*, and rulers with only a few samples are *sparse rulers*, and are used in signal processing [43]. One can modify the power-series scheme so that each block $\mathcal{I}_0$ is a sparse ruler (we used Wichmann Rulers). The scheme still underperforms random sampling; see Appendix A.1 for more details.
4. Sampling blocks (Algorithm 8 in [1]), which gives good estimates of $R_\tau[\mathbf{X}]$ for small τ, but does not attempt to estimate $R_\tau[\mathbf{X}]$ for τ larger than the block size. This does not perform well and details in left for the supplementary information section 1.A.
5. Hierarchical sampling schemes (Algorithm 9 in [1]), designed to improve on block sampling by giving a small amount of large lag information. This method is exact for some derived quantities (like diffusion coefficients) but *ad-hoc* for estimating the large-lag autocorrelation. This method has high errors (see Appendix A.1 for details).

These last two methods (4 and 5) are different than all the other baseline methods we discuss as they require "on-the-fly" computation to record the estimate of $R_\tau[\mathbf{X}]$ for a subset of the lags τ, and this estimate is then updated. These methods do not simply sample $\mathbf{X}$ and then postprocess. Both method 4 and 5 do not give accurate estimates for large lags, hence we do not present further simulation results with these methods.

Particle dimension compression Given a compression ratio γ, randomly sample particles (or grid points) to form $\mathcal{I} \subset \{1, \ldots, N\}$ with size $|\mathcal{I}| = \lceil \gamma N \rceil$ by uniformly selecting **columns** from the data matrix $\mathbf{X}$. The natural unbiased estimator of $R_\tau[\mathbf{X}]$ is then

$$\frac{1}{|\mathcal{I}|} \frac{1}{T-\tau} \sum_{t=1}^{T-\tau} \sum_{i \in \mathcal{I}} \mathbf{X}(t,i)\mathbf{X}(t+\tau, i).$$

Naïve uniform sparsification (both time and particles) Given a compression ratio γ, uniformly sample $\lceil \gamma TN \rceil$ **entries** from $\mathbf{X}$. This approach has the same estimator for autocorrelation of lag τ as the case time dimension compression, except that the sampling set $\mathcal{I}$ and normalization constant now depend on the column i. We refer to this as "naïve" since it uses a uniform distribution, in contrast to complicated weighted sampling schemes like [32] used in the sampling literature. With an appropriate normalization $z_{\tau,i}^{\mathcal{I}}$, the unbiased estimate of $R_\tau[\mathbf{X}]$ is

$$\frac{1}{z_{\tau,i}^{\mathcal{I}}} \sum_{i=1}^{N} \sum_{\substack{t, \text{ such that} \\ (t,i),(t+\tau,i) \in \mathcal{I}}} \mathbf{X}(t,i)\mathbf{X}(t+\tau,i),$$

which can be calculated via the above formula or via Remark 7.

One can combine time dimension and particle dimension compression (doing time-then-particle, or particle-then-time), but for a given overall compression level, we did not find that this improved accuracy, and therefore do not include it in the results.

Remark 7. To efficiently compute the estimate of the autocorrelation for any time dimension compression scheme, i.e., Eq. (7), one can use existing fast autocorrelation functions. Specifically, set the non-sampled entries to zero, so they do not contribute to the sum, and put each column of $\mathbf{X}$ through a standard autocorrelation function and then average the results. To find the normalization factor $z_\tau^{\mathcal{I}}$, one can create an indicator vector $\boldsymbol{\xi}$ where $\xi_t = 1$ if $t \in \mathcal{I}$ and $\xi_t = 0$ if $t \notin \mathcal{I}$ (think of this as a "book-keeping" particle that can be stored as an extra particle or grid-point), and then compute the autocorrelation of $\boldsymbol{\xi}$ to get the normalization $z_\tau^{\mathcal{I}}$. Computing the value by hand is possible but tedious and the programming is error-prone, which may be a reason why simple (non-random) time compression schemes have historically been favored.

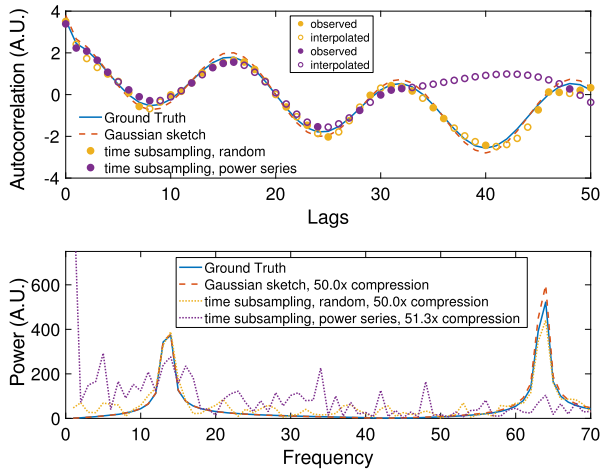

Fig. 1. Autocorrelation (top) and power spectral density (bottom) for the two frequency simulation. (For interpretation of the colors in the figure(s), the reader is referred to the web version of this article.)

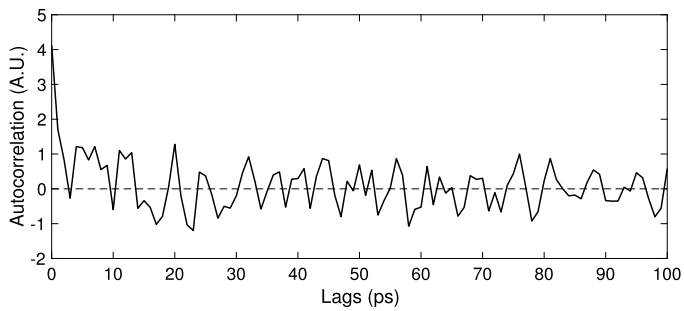

Fig. 2. Ground truth of autocorrelation of the velocity of methanol molecules up to $\tau = 100$.

To illustrate the different types of time dimension compression schemes, we conduct a basic experiment of $N = 10^4$ particles and $T = 2000$ time points with unit spacing, where each particle is randomly assigned one of two possible frequencies (one fast, one slow), and with a random phase; the autocorrelation is the fast sinusoid modulated by the slow sinusoid. The power spectral density ranges up to 500 Hz, of which the first 70 Hz are shown in the bottom of Fig. 1. The ground truth would show two delta functions if $T = \infty$ but are spectrally broadened by the finite time sample. Fig. 1 shows that, at $50\times$ compression, the time sampling approaches have no observations for some lags and must be interpolated. The random time subsampling is more accurate than the power series approach. The Gaussian sketching method requires no interpolation and the PSD it computes is significantly more accurate.

4.2. Methanol ensemble simulation data

Our dataset is a MD simulation using the `LAMMPS` software [9] for $N = 384$ methanol molecules with time step 1 fs for 10 ps, with potentials between pairs of bonded atoms, between triplets and between quadruplets of atoms set as harmonic, and potential for pairwise interactions set as the hybrid of the "DREIDING" hydrogen bonding Lennard-Jones potential and the Lennard-Jones with cut-off Coulombic potential [44]. The quantity of interest is the power spectral density of the velocity of the molecules. Except in Fig. 5, no blocking was performed, so $B = 1$ and $T = T_{\text{long}} = 10000$. The true sample autocorrelation, up to $\tau = 100$, is shown in Fig. 2. The actual simulation was run for 20000 time steps (20 ps) but the first 10 ps are ignored as the simulation was equilibrating.

Fig. 3 shows the corresponding true power spectral density (PSD), as well as the PSD computed via the three proposed sketching methods (with Gaussian, Haar and FJLT sketches), as well as the three benchmark methods, using only about 1% of the data. The three sketching methods faithfully recover the true peaks of the spectrum, while the baseline methods (in blue) either have spurious peaks (time compression and naive uniform compression) or miss/distort peaks (particle compression).

For systematic and quantitative comparison, we consider three metrics for evaluating the estimated PSD $\hat{\mathbf{s}} = \widehat{S}(\omega)$ compared to the true PSD $\mathbf{s} = S(\omega)$. First, we use the relative ℓ_2 norm $\|\hat{\mathbf{s}} - \mathbf{s}\|_2 / \|\mathbf{s}\|_2$ which also captures the relative ℓ_2 error for the autocorrelation (since the Fourier transform is unitary, i.e., Parseval's identity). Second, we use the relative ℓ_∞ error, which is defined as $\max_{i, s_i \neq 0} \frac{|\hat{s}_i - s_i|}{|s_i|}$. Third, we use a relative ℓ_1 error, defined as $\|\hat{\mathbf{s}} - \mathbf{s}\|_1 / \|\mathbf{s}\|_1$, where $\|\mathbf{s}\|_1 = \sum_i |s_i|$.

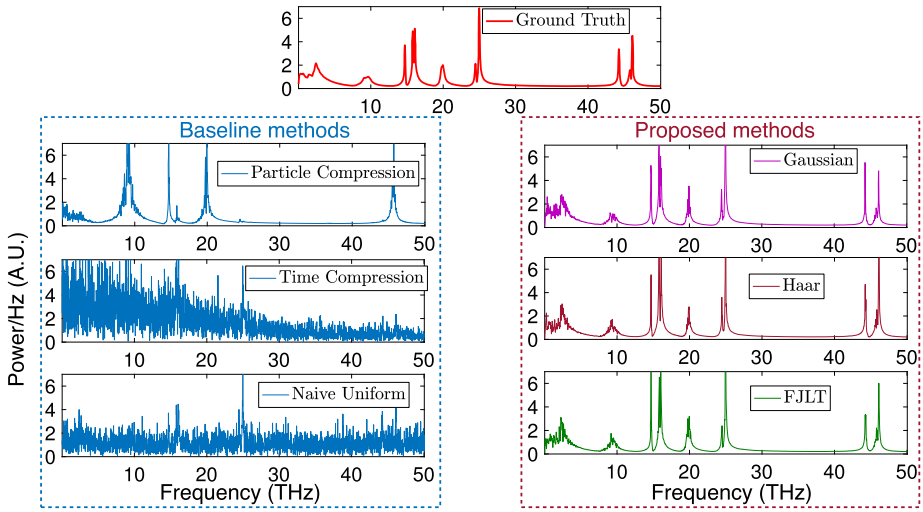

Fig. 3. Power spectral density for methanol data. The compression ratio is 1% for each method.

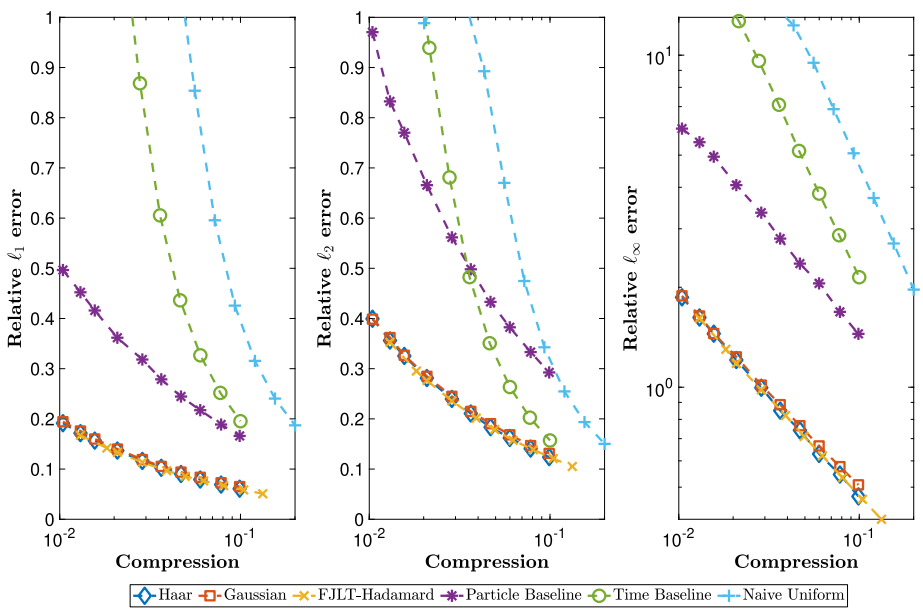

Fig. 4. The error due to approximating the PSD for the proposed methods (Haar, Gaussian, and FJLT-Hadamard) compared to baselines, on the methanol data. Left: relative ℓ_1 error. Middle: relative ℓ_2 error. Right: relative ℓ_∞ error.

When computing the compression ratio, a sketching method with $\mathbf{\Omega} \in \mathbb{R}^{m \times N}$ achieves a $\gamma = m/N$ compression ratio, as no meta-data needs to be stored. The time dimension and particle dimension subsampling methods must also save the time or particle/space indices $\mathcal{I}$ as meta-data, though this is typically insignificant, so they achieve approximately $|\mathcal{I}|/T$ and $|\mathcal{I}|/N$ compression ratios, respectively. The naïve uniform sparsification, which samples in both space and time, must save both time and particle/space indices; this is done implicitly by storing the data as a sparse matrix in compressed sparse column format. The overhead of storing these indices can be significant, which is why the compression ratio for "naïve uniform" is slightly worse than the target of $|\mathcal{I}|/(TN)$.

Fig. 4 shows the error metrics as a function of compression ratio γ in the interesting regime where $\gamma \ll 1$. We see that sketching methods perform better than baseline methods in the ℓ_1, ℓ_2 and ℓ_∞ metrics, and the advantage is most significant when the compression ratio is small.

Fig. 5 shows that the ℓ_1, ℓ_2 and ℓ_∞ errors decay to zero as the time series becomes arbitrarily long. Specifically, we take the total simulation time $T_{\text{long}} \to \infty$, and set $B = T = \sqrt{T_{\text{long}}}$ (this is necessary, since the simpler choice of $B = 1$ and $T = T_{\text{long}}$ does not give a consistent estimator even with fully sampled data). The evaluation of the errors of the autocorrelation is with respect to the first 15 lags. The compression ratio of all sketching methods is fixed as 10%. The

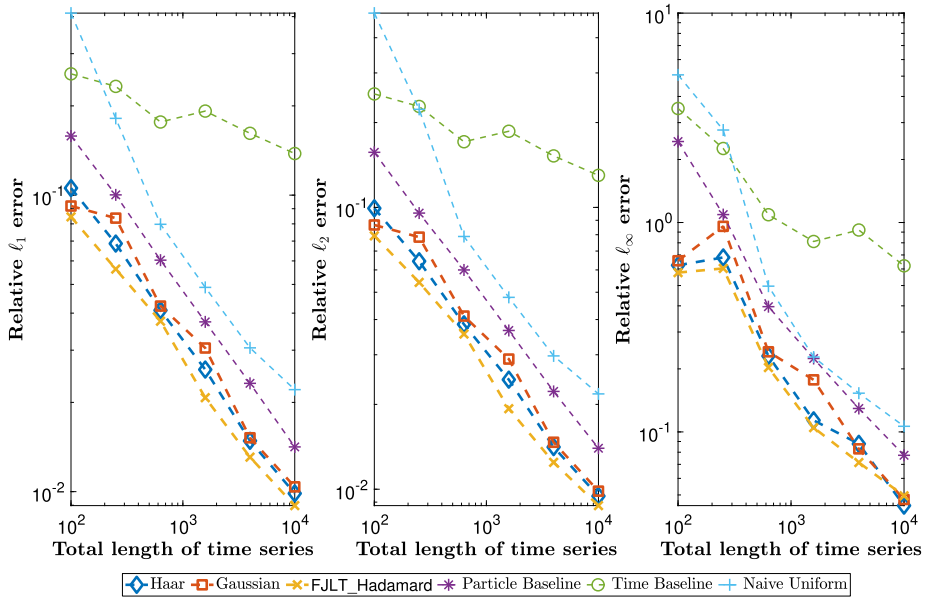

Fig. 5. Three metrics characterizing the discrepancy between estimated autocorrelation of first 15 lags and the ground truth vs. total length of time signals. The full time signal is divided into $B = \sqrt{T_{\text{long}}}$ blocks, each of which is used to evaluate the first 15 lags of autocorrelation.

figure shows that all methods appear to be consistent, with the sketching methods significantly more accurate compared to the *ad hoc* baselines.

Synthetic data The performance of the sketching methods over the classical benchmark methods is significant, but in fact the discrepancy can be arbitrarily large. Appendix A.2 shows a synthetic data set created to be adversarial for the classical methods, for which they perform poorly, whereas the sketching methods do well. The data is created to have a few "special" particles which contribute significantly but are unlikely to be sampled by the particle sampling methods, and to have a few short pulses, so that the relevant time dynamics is likely to be missed by the time sampling methods. The sketching methods are not susceptible to such adversarial examples.

5. Conclusions

Since second order statistics like autocorrelation and power density spectral can be computed via the empirical covariance matrix, this means that sketching methods can be used to preserve statistical properties of the data. These sketching methods come with well-understood theory, little extra computational burden, straightforward implementation, and excellent practical performance. For these reasons, we hope they find their place in the numerical simulation toolkit. An interesting future question is whether even more powerful practical estimators of autocorrelation can be achieved by bypassing the estimation of the covariance matrix.

CRediT authorship contribution statement

Zhishen Huang: Formal analysis, Software, Visualization, Writing – original draft, Writing – review & editing. **Stephen Becker:** Conceptualization, Funding acquisition, Supervision, Visualization, Writing – review & editing.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Acknowledgements

The authors thank Michael Wakin for helpful discussions on fast computation of autocorrelation, Marc Thomson for providing the molecular dynamics data, Francis Starr for discussions of sampling schemes for water, and anonymous reviewers for helpful comments (notably, pointing out a better sample complexity bound for the FJLT). This material is based upon work supported by the National Science Foundation under grant no. 1819251.

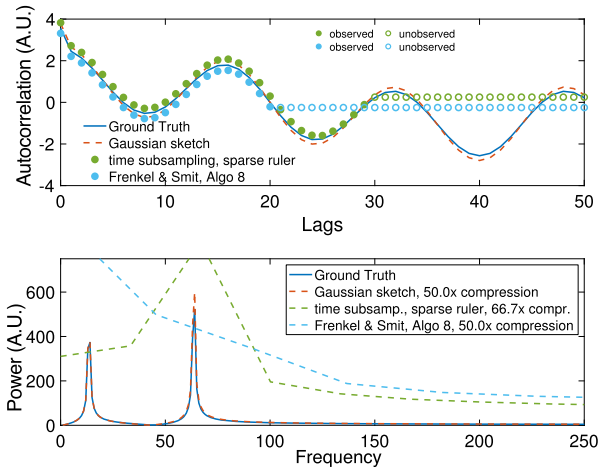

Fig. A.6. Top: autocorrelation, and bottom: Power spectral density (PSD) for a synthetic simulation. The sparse ruler subsampling and the block (Algorithm 8) subsampling miss sampling the autocorrelation at long lags, with the effect of making the PSD estimate have low resolution. Y-axis in arbitrary units for both plots.

Appendix A. Further experiments

A.1. Alternative baseline methods

We expand on other alternatives for time-dimension compression (beyond the (1) random and (2) power-series sampling), namely

3. Sparse ruler sampling. The power-series scheme does not generate all possible lags. Sampling schemes that do generate all possible lags (up to some point) are known as *rulers*, and rulers with only a few samples are *sparse rulers*. One can modify the power-series scheme so that each block $\mathcal{I}_0$ is a sparse ruler (we used Wichmann Rulers).
4. Sampling blocks (Algorithm 8 in [1]), which gives good estimates of $R_\tau[\mathbf{X}]$ for small τ, but does not attempt to estimate $R_\tau[\mathbf{X}]$ for τ larger than the block size.
5. Hierarchical sampling schemes (Algorithm 9 in [1]), designed to improve on block sampling by giving a small amount of large lag information. This method is exact for some derived quantities (like diffusion coefficients) but *ad-hoc* for estimating the large-lag autocorrelation. This method has high errors.

Fig. A.6 compares the sparse ruler sampling and block sampling (Algorithm 8), as well as using the Gaussian sketch. This uses the same $N = 10000$ and $T = 2000$ synthetic data as in Fig. 1 in the main text. Both the sparse ruler sampling and block sampling only observe the autocorrelation for short lags. For this reason, the autocorrelation cannot even be interpolated at missing lags, but rather these values must be extrapolated. Rather than do this, the PSD is computed using only the short time lags, but this has the effect of lowering the resolution of the PSD. The bottom part of the figure shows the PSD.

Fig. A.7 demonstrates the hierarchical sampling scheme on the same data. This scheme samples in blocks (giving a good estimate of short-time autocorrelation lags, much like the block sampling scheme), but then also aggregates blocks to estimate longer lag autocorrelation. For some quantities, such as the diffusion constant when defined as the integral of autocorrelation (e.g., in the discrete case, this is just a sum), this aggregation-by-averaging results in no loss. However, for estimating the autocorrelation itself, the estimate is highly inaccurate. The corresponding PSD is not shown as it is considerably inaccurate.

A.2. Synthetic data

The main paper presents realistic data and shows that newly proposed sketching methods outperform classical methods. Here, we show that the difference in performance can be made almost arbitrarily large by choosing adversarial synthetic data. The specific random nature of the sketching methods makes it impossible to create generic adversarial examples, whereas the classical methods which rely on weaker notions of randomness are much more susceptible.

Creation of the data set Consider a collection of $N = 10{,}000$ particles among which 9997 of them share the same eigenfrequency ω while 3 particles have an additional eigenfrequency ω'. The existence of special particles contributes to the inhomogeneity of the ensemble dynamics. Furthermore, there are 2 pulses in the time range for every particle in the ensemble. Each pulse can be represented by $p_1(t) = p(t - t_1)$, $p_2(t) = p(t - t_2)$ and $p(t) = 10\sin\left(\frac{\pi}{\delta}t\right)\mathbb{1}(-\frac{\delta}{2} \le t \le \frac{\delta}{2})$, where

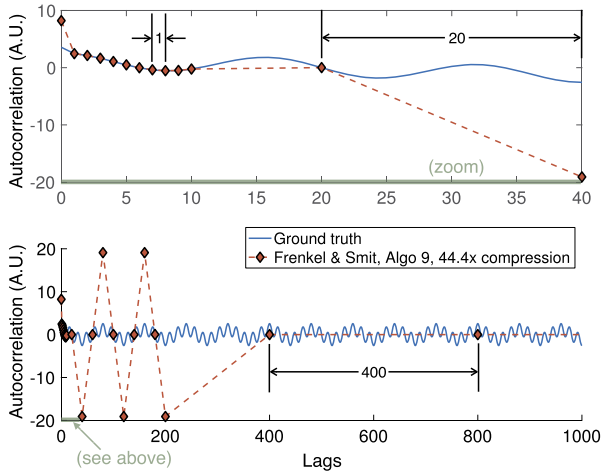

Fig. A.7. Autocorrelation, demonstrating the hierarchical sampling scheme of Algorithm 9. The top plot is a zoomed in version of the bottom plot. The estimate of the autocorrelation at long lags is inaccurate, and the resulting PSD is unusable.

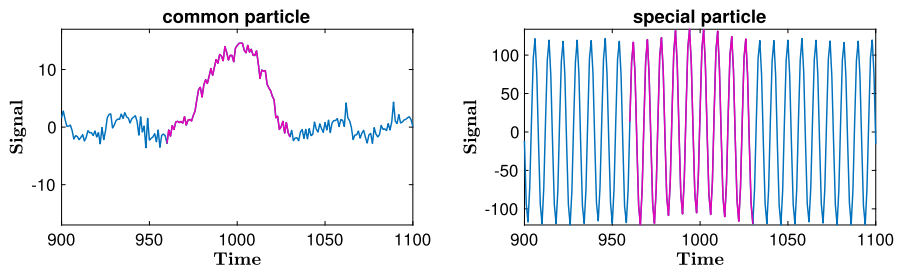

Fig. A.8. Example of particle dynamics in synthetic data. The left subfigures shows the signal of a common particle and the right subfigure shows the signal of a particle with two eigen-frequencies. 2 pulses exist in the synthetic signal and are introduced apart from each other thus not merging their peaks, while we show the zoomed version of one pulse, which is marked in the color of magenta.

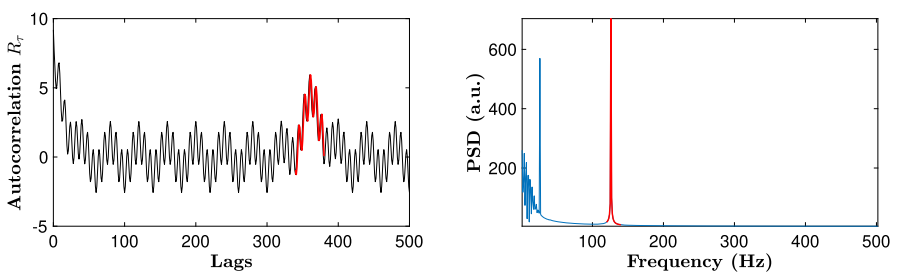

Fig. A.9. Autocorrelation and power spectral density of the synthetic data. The red peak in the power spectral density exists because of special particles, and the red lags in autocorrelation are due to existence of pulses.

$\delta \approx 0.6 \cdot \frac{2\pi}{\omega}$ which accounts for more than half of a period of the signal with common eigenfrequency, and $\mathbb{1}$ is the 0-1 indicator function. Each particle has a random phase $\varphi_i \in [0, 2\pi)$. Specifically, 9997 particles have the "common" dynamics

$$(i = 1, \ldots, 9997) \quad x_i^{\text{common}}(t) = \sin(\omega t + \varphi_i) + p_1(t) + p_2(t) + \varepsilon_i(t)$$

while 3 "special" particles have one more ingredient in their dynamics

$$(j = 9998, 9999, 10000) \quad x_j^{\text{special}}(t) = \sin(\omega t + \varphi_j) + 80\sin(\omega' t + \varphi_j') + p_1(t) + p_2(t) + \varepsilon_j(t)$$

so that when taking the expectation the additional frequency component demonstrates significant importance in the overall spectrum, and $\varepsilon(t)$ is white noise. Fig. A.8 shows the signal example of a common particle and a special particle, while the ground truth autocorrelation and power spectral density are shown in Fig. A.9.

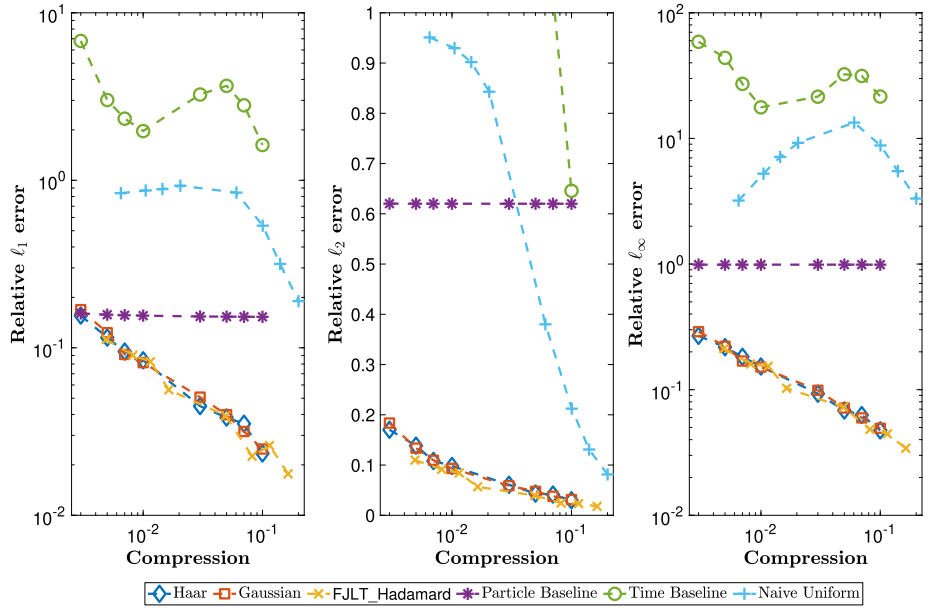

Fig. A.10. Three metrics characterizing accuracy of sketching methods on the PSD in the case of adversarial synthetic data.

Fig. A.10 shows the performance of each sketching method on evaluating the power spectral density of the synthetic data set. The sketching methods perform well, whereas the classical baseline methods perform so poorly as to be unusable. For the sketching methods, even when compression is around 1%, the characteristic peak in the PSD formed by the 3 special particles is still correctly identified, whereas it is completely missed by all 3 classical methods. This is mostly demonstrated by the relative ℓ_∞ error which captures the largest discrepancy in PSD evaluation at any frequency. In fact, all the baseline methods have over 100% relative error on the ℓ_∞ error, regardless of compression.

Fig. A.11 is the same experiment as Fig. A.10 but also reports information on the variance with respect to the ℓ_1 errors. Specifically, box plots are shown, with the middle red line showing the median, and the top and bottom of the box are the 75% and 25% percentiles, respectively. The boxes for the sketching approaches appear large, but due to the logarithmic scale of the y-axis, there is actually not too much spread. The time baseline is inaccurate and has large spread; the naive uniform baseline has less spread but is also inaccurate. The particle baseline shows reasonable good performance for the median, but has worrisome outliers (as indicated by the red $+$ symbols). This is expected for this particular synthetic setup, since the method is reasonable at capturing most of the behavior as long as it does *not* sample one of the three "special" particle. In the cases when it does sample a "special" particle, the method has no way to know that these particles are rare, so due to the normalization, it heavily weights these particles and incorrectly estimates their effect. These are the outliers shown in the figure, and their effect gets larger as $\gamma \to 0$ since the normalization factor grows. The variation with respect to the ℓ_∞ and ℓ_2 metrics is similar.

A.3. Variance information

The following plots show the variation of errors (as reported respectively in Figs. 4 and 5) when sketching methods are used to evaluate PSD/autocorrelation. On each box, the central mark indicates the median, and the bottom and top edges of the box indicate the 25th and 75th percentiles, respectively. The whiskers extend to the most extreme data points not considered outliers, and the outliers are plotted individually using the '+' symbol, if any.

We only show data for the relative ℓ_2 norm errors, but the results for ℓ_1 and ℓ_∞ norm errors are similar.

Fig. A.12 shows that for approximating the PSD, the sketching methods have a reasonably small spread. The variance seems to increase as the compression ratio $\gamma \to 0$ which makes sense since there is less averaging when there are fewer samples. The time and naive subsampling baseline methods have reasonably low spread too, but very large errors. The particle baseline has a large variance in all compression regimes.

Fig. A.13 shows the variability when approximating the first 15 lags of the autocorrelation. All methods have somewhat similar variance at a given error level. However, note that the y-axis is log scale, so if two boxes seem the same size but one is centered at a lower relative error, then that box represents less spread of the data. Hence we again see the trend that most of the methods have lower variance when there is more data (larger γ) since they are also more accurate in this regime.

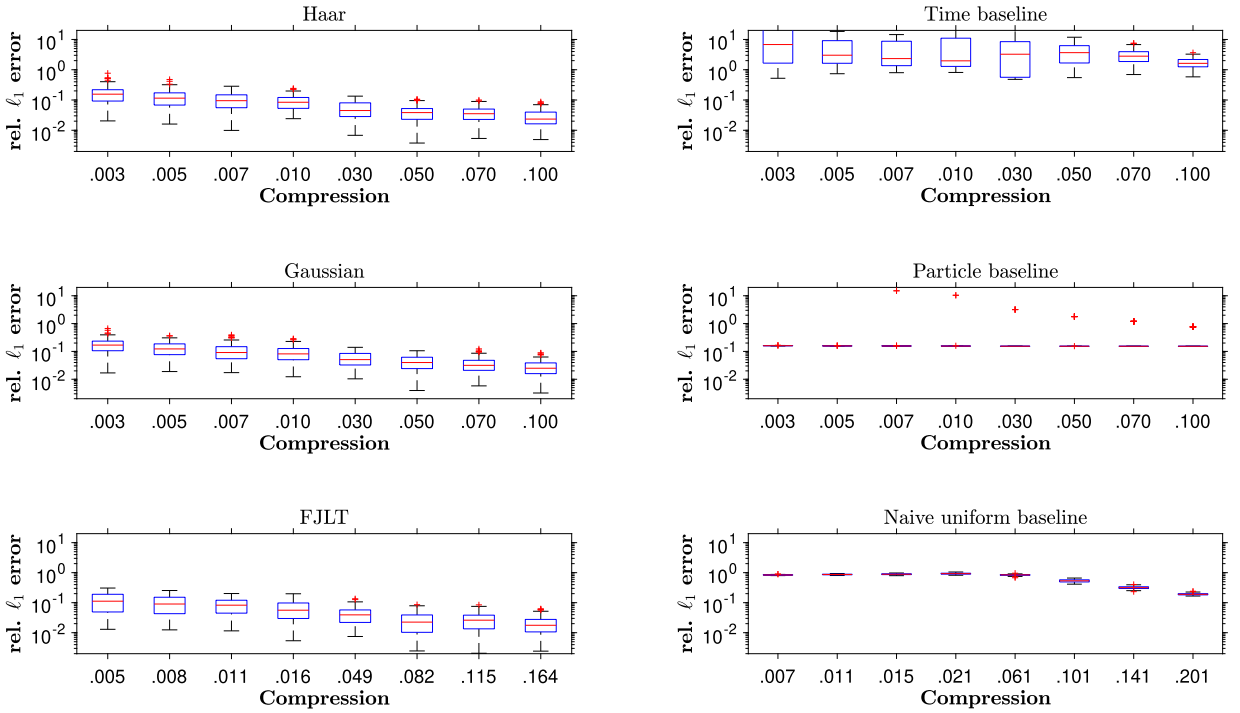

Fig. A.11. Variability of relative ℓ_1 errors (as reported in Fig. A.10) due to approximating the PSD for the proposed methods (Haar, Gaussian, and FJLT-Hadamard) compared to baselines, on the methanol data. With respect to each compression mark and each sketching method, the experiment is repeated for 100 trials.

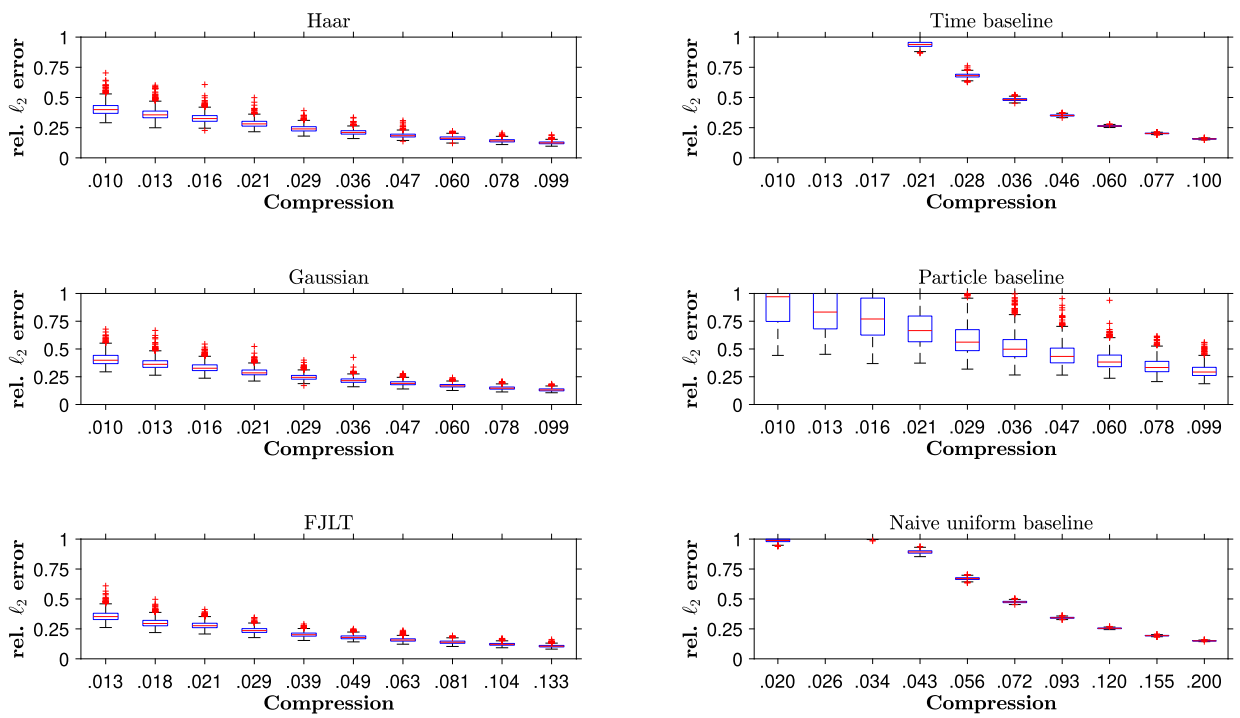

Fig. A.12. Variability of relative ℓ_2 errors (as reported in Fig. 4) due to approximating the PSD for the proposed methods (Haar, Gaussian, and FJLT-Hadamard) compared to baselines, on the methanol data. With respect to each compression mark and each sketching method, the experiment is repeated for 1000 trials.

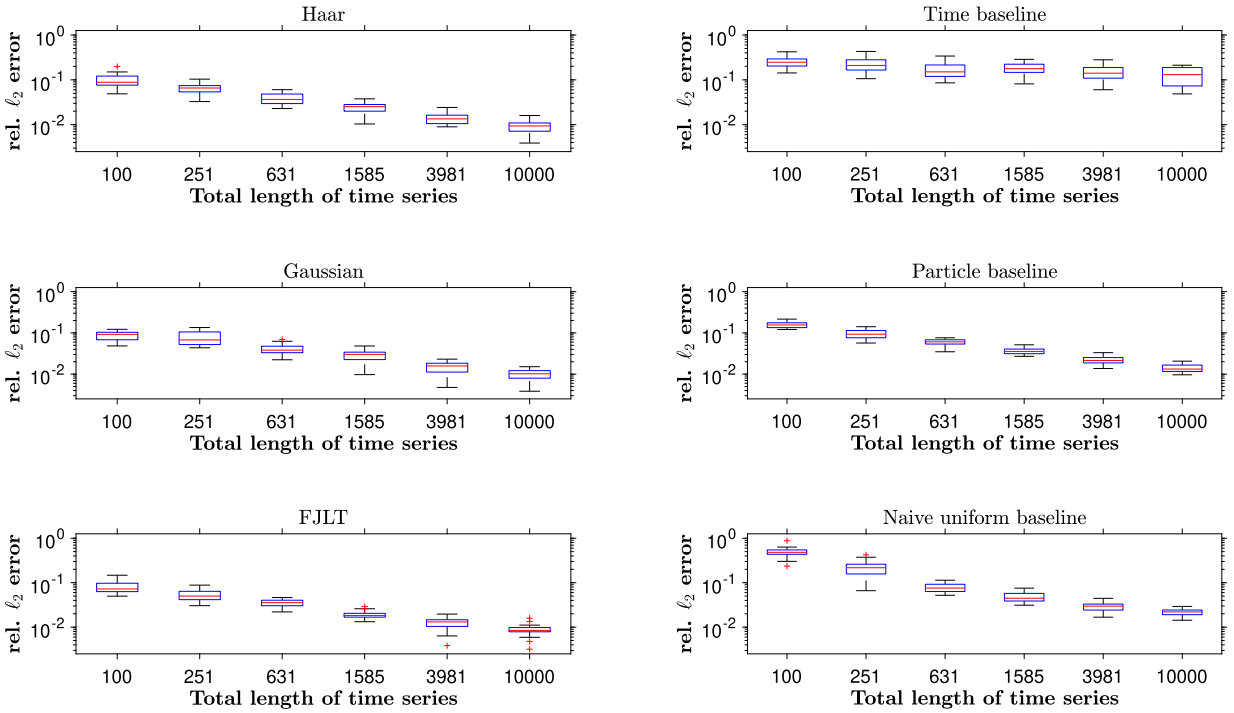

Fig. A.13. Variability of ℓ_2 error of the estimated autocorrelation of first 15 lags (as reported in Fig. 5) vs. total length of time signals. With respect to each fixed length of time series and each sketching method, the experiment is repeated for 20 trials.

References

[1] D. Frenkel, B. Smit, Chapter 4 - molecular dynamics simulations, in: D. Frenkel, B. Smit (Eds.), Understanding Molecular Simulation, 2nd edition, Academic Press, San Diego, 2002, pp. 63–107, http://www.sciencedirect.com/science/article/pii/B9780122673511500067.

[2] E. Runge, E.K.U. Gross, Density-functional theory for time-dependent systems, Phys. Rev. Lett. 52 (1984) 997–1000, https://doi.org/10.1103/PhysRevLett.52.997, https://link.aps.org/doi/10.1103/PhysRevLett.52.997.

[3] A.P. Scott, L. Radom, Harmonic vibrational frequencies: an evaluation of Hartree-Fock, Møller-Plesset, quadratic configuration interaction, density functional theory, and semiempirical scale factors, J. Phys. Chem. 100 (41) (1996) 16502–16513.

[4] K. Yabana, G.F. Bertsch, Time-dependent local-density approximation in real time, Phys. Rev. B 54 (1996) 4484–4487, https://doi.org/10.1103/PhysRevB.54.4484, https://link.aps.org/doi/10.1103/PhysRevB.54.4484.

[5] D. Varsano, D.A. Espinosa-Leal, X. Andrade, M.A.L. Marques, R. di Felice, A. Rubio, Towards a gauge invariant method for molecular chiroptical properties in TDDFT, Phys. Chem. Chem. Phys. 11 (2009) 4481–4489, https://doi.org/10.1039/B903200B.

[6] I. Rodríguez, O. Lehmkuhl, R. Borrell, C. Pérez-Segarra, On DNS and LES of natural convection of wall-confined flows: Rayleigh-Bénard convection, in: H. Kuerten, B. Geurts, V. Armenio, J. Fröhlich (Eds.), Direct and Large-Eddy Simulation VIII, in: ERCOFTAC Series, vol. 15, Springer, 2011, pp. 389–394.

[7] I. Grooms, A.J. Majda, Efficient stochastic superparameterization for geophysical turbulence, Proc. Natl. Acad. Sci. 110 (12) (2013) 4464–4469.

[8] I. Grooms, W. Kleiber, Diagnosing, modeling, and testing a multiplicative stochastic Gent-McWilliams parameterization, Ocean Model. 133 (2019) 1–10.

[9] S. Plimpton, Fast parallel algorithms for short-range molecular dynamics, J. Comput. Phys. 117 (1995) 1–19, http://lammps.sandia.gov.

[10] D. Rapaport, The Art of Molecular Dynamics Simulation, Cambridge University Press, 2004.

[11] LAMMPS benchmarks, https://lammps.sandia.gov/bench.html#billion, 2012. (Accessed 13 February 2020).

[12] P. Lindstrom, M. Isenburg, Fast and efficient compression of floating-point data, IEEE Trans. Vis. Comput. Graph. 12 (5) (2006) 1245–1250.

[13] P. Lindstrom, Fixed-rate compressed floating-point arrays, IEEE Trans. Vis. Comput. Graph. 20 (12) (2014) 2674–2683.

[14] M. Salloum, N.D. Fabian, D.M. Hensinger, J. Lee, E.M. Allendorf, A. Bhagatwala, M.L. Blaylock, J.H. Chen, Jeremy A. Templeton, I. Tezaur, Optimal compressed sensing and reconstruction of unstructured mesh datasets, Data Sci. Eng. 3 (1) (2018) 1–23.

[15] K.L. Clarkson, P. Drineas, M. Magdon-Ismail, M.W. Mahoney, X. Meng, D.P. Woodruff, The fast Cauchy transform and faster robust linear regression, in: Proceedings of the Twenty-Fourth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA '13, Society for Industrial and Applied Mathematics, Philadelphia, PA, USA, 2013, pp. 466–477, http://dl.acm.org/citation.cfm?id=2627817.2627851.

[16] K.L. Clarkson, Subgradient and sampling algorithms for l1 regression, in: Proceedings of the Sixteenth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA '05, Society for Industrial and Applied Mathematics, Philadelphia, PA, USA, 2005, pp. 257–266, http://dl.acm.org/citation.cfm?id=1070432.1070469.

[17] X. Meng, M.W. Mahoney, Low-distortion subspace embeddings in input-sparsity time and applications to robust linear regression, in: Proceedings of the Forty-Fifth Annual ACM Symposium on Theory of Computing, STOC '13, ACM, New York, NY, USA, 2013, pp. 91–100, http://doi.acm.org/10.1145/2488608.2488621.

[18] C. Sohler, D.P. Woodruff, Subspace embeddings for the l1-norm with applications, in: Proceedings of the Forty-Third Annual ACM Symposium on Theory of Computing, STOC '11, ACM, New York, NY, USA, 2011, pp. 755–764, http://doi.acm.org/10.1145/1993636.1993736.

[19] D.P. Woodruff, Q. Zhang, Subspace embeddings and ℓ_p-regression using exponential random variables, in: S. Shalev-Shwartz, I. Steinwart (Eds.), COLT - The 26th Annual Conference on Learning Theory, in: JMLR Workshop and Conference Proceedings, vol. 30, JMLR.org, Princeton University, NJ, USA, 2013, pp. 546–567, http://proceedings.mlr.press/v30/Woodruff13.html.

[20] K.L. Clarkson, D.P. Woodruff, Low rank approximation and regression in input sparsity time, J. ACM 63 (6) (2012) 1–45, https://doi.org/10.1145/2488608.2488620, arXiv:1207.6365, http://arxiv.org/abs/1207.6365.

[21] N. Halko, P.-G. Martinsson, J.A. Tropp, Finding structure with randomness: probabilistic algorithms for constructing approximate matrix decompositions, SIAM Rev. 53 (2) (2011) 217–288.
[22] P. Drineas, M.W. Mahoney, S. Muthukrishnan, Relative-error cur matrix decompositions, SIAM J. Matrix Anal. Appl. 30 (2) (2008) 844–881, https://doi.org/10.1137/07070471X.
[23] M.W. Mahoney, P. Drineas, CUR matrix decompositions for improved data analysis, Proc. Natl. Acad. Sci. 106 (3) (2009) 697–702, https://doi.org/10.1073/pnas.0803205106, https://www.pnas.org/content/106/3/697.full.pdf, https://www.pnas.org/content/106/3/697.
[24] W.B. Johnson, J. Lindenstrauss, Extensions of Lipschitz mappings into a Hilbert space, Contemp. Math. 26 (189–206) (1984) 1.
[25] S. Becker, Matlab code for sketching, https://github.com/stephenbeckr/randomized-algorithm-class/blob/master/Code/sketch.m, 2019.
[26] F. Mezzadri, How to generate random matrices from the classical compact groups, Not. Am. Math. Soc. 54 (5) (2007) 592–604.
[27] D.P. Woodruff, Sketching as a tool for numerical linear algebra, Found. Trends Theor. Comput. Sci. 10 (1–2) (2014) 1–157, https://doi.org/10.1561/0400000060.
[28] N. Ailon, B. Chazelle, The fast Johnson-Lindenstrauss transformation and approximate nearest neighbors, SIAM J. Comput. 39 (1) (2009) 302–322.
[29] G. Cormode, Sketch techniques for approximate query processing, in: Synposes for Approximate Query Processing: Samples, Histograms, Wavelets and Sketches, Foundations and Trends in Databases, NOW Publishers, 2011.
[30] M. Mahoney, Randomized algorithms for matrices and data, Found. Trends Mach. Learn. 3 (2) (2011) 123–224.
[31] D. Achlioptas, F. Mcsherry, Fast computation of low-rank matrix approximations, J. ACM 54 (2) (Apr. 2007), https://doi.org/10.1145/1219092.1219097, http://doi.acm.org/10.1145/1219092.1219097.
[32] D. Achlioptas, Z. Karnin, E. Liberty, Near-optimal entrywise sampling for data matrices, in: Proceedings of the 26th International Conference on Neural Information Processing Systems - Volume 1, NIPS'13, Curran Associates Inc., USA, 2013, pp. 1565–1573, http://dl.acm.org/citation.cfm?id=2999611.2999786.
[33] F. Pourkamali-Anaraki, S. Becker, Preconditioned data sparsification for big data with applications to PCA and K-means, IEEE Trans. Inf. Theory 63 (5) (2017) 2954–2974, https://doi.org/10.1109/TIT.2017.2672725.
[34] P.-G. Martinsson, J. Tropp, Randomized numerical linear algebra: foundations & algorithms, arXiv preprint, arXiv:2002.01387, 2020.
[35] R. Vershynin, High-Dimensional Probability: An Introduction with Applications in Data Science, Cambridge Series in Statistical and Probabilistic Mathematics, Cambridge University Press, 2018.
[36] O. Balabanov, A. Nouy, Randomized linear algebra for model reduction. Part I: Galerkin methods and error estimation, Adv. Comput. Math. 45 (5–6) (2019) 2969–3019.
[37] F. Krahmer, R. Ward, New and improved Johnson–Lindenstrauss embeddings via the restricted isometry property, SIAM J. Math. Anal. 43 (3) (2011) 1269–1281, https://doi.org/10.1137/100810447.
[38] S. Foucart, H. Rauhut, A Mathematical Introduction to Compressive Sensing, Birkhäuser, New York, NY, 2013.
[39] P. Broersen, Automatic Autocorrelation and Spectral Analysis, Springer Science & Business Media, 2006.
[40] T. Sarlos, Improved approximation algorithms for large matrices via random projections, in: 2006 47th Annual IEEE Symposium on Foundations of Computer Science (FOCS'06), 2006, pp. 143–152.
[41] P.J. Brockwell, R.A. Davis, Time Series: Theory and Methods, Springer, 1987.
[42] J.G. Proakis, D.K. Manolakis, Digital Signal Processing, 4th edition, Prentice-Hall, Inc., Upper Saddle River, NJ, USA, 2006.
[43] D. Romero, G. Leus, Compressive covariance sampling, in: 2013 Information Theory and Applications Workshop (ITA), 2013, pp. 1–8.
[44] S. Plimpton, A. Thompson, S. Moore, A. Kohlmeyer, R. Berger, Lammps dreiding example, https://github.com/lammps/lammps/tree/master/examples/dreiding, 2011.