Sparse Recovery for Orthogonal Polynomial Transforms

Anna Gilbert
Department of Mathematics, Yale University, New Haven, CT, USA
anna.gilbert@yale.edu

Albert Gu
Department of Computer Science, Stanford University, CA, USA
albertgu@stanford.edu

Christopher Ré
Department of Computer Science, Stanford University, CA, USA
chrismre@cs.stanford.edu

Atri Rudra
Department of Computer Science and Engineering, University at Buffalo, NY, USA
atri@buffalo.edu

Mary Wootters
Departments of Computer Science and Electrical Engineering, Stanford University, CA, USA
marykw@stanford.edu

Abstract

In this paper we consider the following *sparse recovery* problem. We have query access to a vector $\mathbf{x} \in \mathbb{R}^N$ such that $\hat{\mathbf{x}} = \mathbf{F}\mathbf{x}$ is k-sparse (or nearly k-sparse) for some orthogonal transform $\mathbf{F}$. The goal is to output an approximation (in an ℓ_2 sense) to $\hat{\mathbf{x}}$ in sublinear time. This problem has been well-studied in the special case that $\mathbf{F}$ is the Discrete Fourier Transform (DFT), and a long line of work has resulted in sparse Fast Fourier Transforms that run in time $O(k \cdot \text{polylog} N)$. However, for transforms $\mathbf{F}$ other than the DFT (or closely related transforms like the Discrete Cosine Transform), the question is much less settled.

In this paper we give sublinear-time algorithms – running in time $\text{poly}(k \log(N))$ – for solving the sparse recovery problem for orthogonal transforms $\mathbf{F}$ that arise from *orthogonal polynomials*. More precisely, our algorithm works for any $\mathbf{F}$ that is an orthogonal polynomial transform derived from *Jacobi polynomials*. The Jacobi polynomials are a large class of classical orthogonal polynomials (and include *Chebyshev* and *Legendre* polynomials as special cases), and show up extensively in applications like numerical analysis and signal processing. One caveat of our work is that we require an assumption on the sparsity structure of the sparse vector, although we note that vectors with random support have this property with high probability.

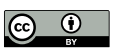
Our approach is to give a very general reduction from the k-sparse sparse recovery problem to the 1-sparse sparse recovery problem that holds for any flat orthogonal polynomial transform; then we solve this one-sparse recovery problem for transforms derived from Jacobi polynomials. Frequently, sparse FFT algorithms are described as implementing such a reduction; however, the technical details of such works are quite specific to the Fourier transform and moreover the actual implementations of these algorithms do not use the 1-sparse algorithm as a black box. In this work we give a reduction that works for a broad class of orthogonal polynomial families, and which uses any 1-sparse recovery algorithm as a black box.

2012 ACM Subject Classification Theory of computation → Streaming, sublinear and near linear time algorithms; Theory of computation → Algorithm design techniques

Keywords and phrases Orthogonal polynomials, Jacobi polynomials, sublinear algorithms, sparse recovery

Digital Object Identifier 10.4230/LIPIcs.ICALP.2020.58

Category Track A: Algorithms, Complexity and Games



47th International Colloquium on Automata, Languages, and Programming (ICALP 2020).
Editors: Artur Czumaj, Anuj Dawar, and Emanuela Merelli; Article No. 58; pp. 58:1–58:16

Leibniz International Proceedings in Informatics
Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

Related Version https://arxiv.org/abs/1907.08362

Funding *Anna Gilbert*: ACG is partially funded by a Simons Foundation Fellowship.
Albert Gu: AG and CR gratefully acknowledge the support of DARPA under Nos. FA87501720095 (D3M) and FA86501827865 (SDH), NIH under No. U54EB020405 (Mobilize), NSF under Nos. CCF1763315 (Beyond Sparsity) and CCF1563078 (Volume to Velocity), ONR under No. N000141712266 (Unifying Weak Supervision), the Moore Foundation, NXP, Xilinx, LETI-CEA, Intel, Google, NEC, Toshiba, TSMC, ARM, Hitachi, BASF, Accenture, Ericsson, Qualcomm, Analog Devices, the Okawa Foundation, and American Family Insurance, Google Cloud, Swiss Re, Stanford Bio-X SIG Fellowship, and members of the Stanford DAWN project: Intel, Microsoft, Teradata, Facebook, Google, Ant Financial, NEC, SAP, VMWare, and Infosys. The U.S. Government is authorized to reproduce and distribute reprints for Governmental purposes notwithstanding any copyright notation thereon. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the authors and do not necessarily reflect the views, policies, or endorsements, either expressed or implied, of DARPA, NIH, ONR, or the U.S. Government.
Atri Rudra: AR is partially funded by NSF grant CCF-1763481.
Mary Wootters: MW is partially funded by NSF CAREER award CCF-1844628.

Acknowledgements We would like to thank Mark Iwen for useful conversations. Thanks also to Stefan Steinerberger for showing us the proof of a number theory lemma we used in the 1-sparse Jacobi solver and graciously allowing us to use it, and to Clément Canonne for helpful comments on our manuscript. We also thank Eric Winsor for pointing out an error in a lemma in an earlier version of this paper. Thanks also to anonymous reviewers who pointed out some missing references and whose comments improved the presentation of the paper.

1 Introduction

In this paper, we consider the following *sparse recovery* problem. Suppose that we have query access to a vector $\mathbf{x} \in \mathbb{R}^N$, which has the property that for a fixed *orthogonal transform matrix* $\mathbf{F}$, $\hat{\mathbf{x}} = \mathbf{F}\mathbf{x}$ is k-sparse (or approximately k-sparse, in the sense that $\hat{\mathbf{x}}$ is close in ℓ_2 distance to a k-sparse vector). The goal is to recover an approximation $\hat{\mathbf{z}}$ to $\hat{\mathbf{x}}$, so that $\|\hat{\mathbf{x}} - \hat{\mathbf{z}}\|_2$ is small with high probability, as quickly as possible.

Variants of this problem have been studied extensively over several decades – we refer the reader to the book [16] for many examples and references. One particularly well-studied example is the *sparse Fast Fourier Transform* (sFFT) – see the survey [18] and the references therein. In this case, the matrix $\mathbf{F}$ is taken to be the Discrete Fourier Transform (DFT) and a long line of work has produced near-optimal results: algorithms with running time $O(k\,\mathrm{polylog}(N))$ and sample complexity $O(k \log N)$ [8, 9, 19, 23–27]– though not all of these works achieve both the claimed sample complexity and runtime at the same time.

We study the sparse recovery problem for a more general class of transforms $\mathbf{F}$ called *orthogonal polynomial transforms,* and in particular those that arise from *Jacobi polynomials,* a broad class of orthogonal polynomials (OPs). Jacobi polynomials include as special cases many familiar families of OPs, including Gegenbauer and in particular Chebyshev, Legendre, and Zernike[1] polynomials, and the corresponding OP transforms appear throughout numerical analysis and signal processing.

Despite the progress on the sFFT described above, much remains unknown for general orthogonal polynomial transforms. As discussed more in Section 1.2 below, the *sample complexity* of the sparse recovery problem is well understood, and the "correct" answer

[1] To be more precise the *radial* component of a Zernike polynomial is a Gegenbauer and hence, a Jacobi polynomial.

is known to be $\Theta(k \operatorname{polylog}(N))$ queries to $\mathbf{x}$. However, the algorithmic results that go along with these sample complexity bounds result in $\operatorname{poly}(N)$ time algorithms. Our goal in this work will be *sublinear* time algorithms as well as sublinear sample complexity. There are sublinear-time algorithms available for the special cases of Chebyshev and Legendre polynomials that work by essentially reducing to the Fourier case [21]. For general Jacobi polynomials, such reductions are not available. We elaborate in the full version of the paper why reducing general Jacobi polynomials to the Fourier case does not seem easy. There are also algorithms based on Prony's method, some of which work for quite general families of OPs [29]. However these general results require exact sparsity; to the best of our knowledge versions of Prony's method that are provably robust to noise are restricted to classes of OPs similar to the Fourier transform.

Results

In this work, we give the first (to the best of our knowledge) sublinear-time algorithms with provable guarantees for the (approximately-)sparse recovery problem for general orthogonal transforms derived from Jacobi polynomials. We discuss our results in more detail in Section 3 and briefly summarize them here. Our algorithms run in time $\operatorname{poly}(k \log(N))$ and given query access to $\mathbf{v} = \mathbf{F}^{-1}\hat{\mathbf{v}}$, can find approximations to $\hat{\mathbf{v}}$ when $\hat{\mathbf{v}}$ is approximately k-sparse of an appropriate form. More precisely, we can handle vectors $\hat{\mathbf{v}} = \hat{\mathbf{x}} + \hat{\mathbf{w}}$ where $\hat{\mathbf{x}}$ is k-sparse with a "spread-out" support (made precise in Definition 2.3), and $\hat{\mathbf{w}}$ is an adversarial noise vector with sufficiently small ℓ_2 norm. We obtain guarantees of the following flavor: for any such vector $\mathbf{v}$, we can find $\hat{\mathbf{z}}$ such that $\|\hat{\mathbf{z}} - \hat{\mathbf{x}}\|_2 \leq 0.01\|\hat{\mathbf{x}}\|_2$ with high probability.

We note that these results are weaker than the results for the sFFT: our sample complexity and running time are polynomially larger, and we need stronger assumptions on the sparse signals. However, we also note that the decade or so of work on the sFFT culminating in the results above began with similar results (see [17], for example, in which the dependence on k is an unspecified polynomial) and we hope that this work will similarly be a first step towards near-optimal algorithms for general orthogonal polynomial transforms.

Techniques

Our techniques follow the outline of existing algorithms for the sFFT, although as we elaborate on in Section 1.3, the situation for general Jacobi polynomials is substantially more complicated. More precisely, we first give a very general reduction, which reduces the k-sparse case to the 1-sparse case. The idea of such a reduction was implicit in the sFFT literature, but previous work has relied heavily on the structure of the DFT. Our reduction applies to a broad class of OPs including Jacobi polynomials. Next, we show how to solve the 1-sparse recovery problem for general Jacobi polynomials. The basic idea is to use known approximations of Jacobi polynomial evaluations by certain cosine evaluations [35] in order to iteratively narrow down the support of the unknown 1-sparse vector. We give a more detailed overview of our techniques in Section 1.3.

Organization

For the rest of the introduction, we briefly introduce orthogonal polynomial transforms, discuss previous work, and give a high-level overview of our approach. After that we introduce the formal notation and definitions we need in Section 2, after which we state our results more formally in Section 3.

Due to space constraints, proofs of our main results are in the full version of the paper, including the reduction from k to 1-sparse recovery, the 1-sparse recovery algorithm for Jacobi polynomials, and the resulting k-sparse recovery algorithm for Jacobi polynomials.

1.1 Orthogonal Polynomial Transforms

Orthogonal polynomials (OPs) play an important role in classical applied mathematics, mathematical physics, and the numerical analysis necessary to simulate solutions to such problems. We give more precise definitions in Section 2; briefly, a family of orthogonal polynomials $p_0(X), p_1(X), \ldots$ is a collection of polynomials defined on an interval $\mathcal{D}$ of $\mathbb{R}$, that are pairwise orthogonal with respect to a (non-negative) weight function w.

Orthogonal polynomials naturally give rise to (discrete) orthogonal polynomial transforms. In particular, we define the transform as follows– $\mathbf{F}$ is an $N \times N$ matrix, with each column corresponding to an orthogonal polynomial $p_0, \ldots, p_{N-1}$ and each row an evaluation point $\lambda_0, \ldots, \lambda_{N-1}$ in a suitable domain and suitably normalized so that it is an orthogonal matrix (Definition 2.1). A familiar example might be the DFT: in this language, the DFT matrix is defined by the polynomials $1, X, X^2, \ldots, X^{N-1}$, evaluated at points $\lambda_j = \omega^j$ where ω is the Nth root of unity.[2] Like the Fourier Transform, it is known that all OP transforms admit "fast" versions, allowing matrix-vector multiplication in time $O(N \log^2(N))$ [15].[3] Thus, our problem of sparse recovery for OP transforms is a natural extension of the sFFT problem, with applications to several areas mentioned below.

In this work we study *Jacobi polynomials* (defined formally in Section 2), which are a very general class of orthogonal polynomials. These include Chebyshev polynomials, Legendre polynomials, Zernike polynomials and more generally Gegenbauer polynomials. These OP families show up in many places. For example, Zernike polynomials are a family of orthogonal polynomials on the unit disk that permit an analytic expression of the 2D Fourier transform on the disk. They are used in optics and interferometry [36]. They can be utilized to extract features from images that describe the shape characteristics of an object and were recently used for improved cancer imaging [39]. Different families of orthogonal polynomials give rise to different quadrature rules for numerical integration [12, 33]. Specifically, Chebyshev polynomials are used for numerical stability (see e.g. the ChebFun package [3]) as well as approximation theory (see e.g. Chebyshev approximation [1]). Chebyshev polynomials also have certain optimal *extremal* properties, which has resulted in many uses in theoretical computer science, including in learning theory, quantum complexity theory, linear systems solvers, eigenvector computation, optimization, and more [28]. Further, Jacobi polynomials form solutions of certain differential equations [2].

More recently, orthogonal polynomials and orthogonal polynomial transforms have found applications in various facets of machine learning. For example, Dao et al. [13] leverage the connection between orthogonal polynomials and quadrature to derive rules for computing

[2] We note that in this work we consider a setting slightly different than this example, where $\mathcal{D} = [-1, 1]$ rather than $\mathbf{S}^1$.

[3] We note that even though the work of [15] has in some sense solved the problem of computing any OP transform in near-linear time, many practical issues still remain to be resolved and the problem of computing OP transforms in near-linear time has seen a lot of research activity recently. We just mention two recent works [6, 7] that present near-linear time algorithms for the Jacobi polynomial transforms (and indeed their notion of *uniform Jacobi transform* corresponds exactly to the Jacobi polynomial transform that we study in this paper). However, these algorithms inherently seem to require at least linear-time and it is not clear how to convert them into sub-linear algorithms, which is the focus of our work.

kernel features in machine learning. The Legendre Memory Unit [38] augments recurrent neural networks by orthogonalizing the history of features on a sliding Legendre basis; mathematically, this is essentially an online update of the discrete Legendre Transform. More directly, Thomas et al. [37] apply parametrized families of structured matrices directly inspired by orthogonal polynomial transforms (De Sa et al. [14]) as layers in neural networks. In this context, *any* form of structured matrix that admits fast operations is valuable, such as those considered in this work. Although not directly applied yet, all of these applications have a natural way of incorporating *sparsity* if the appropriate sparse transforms exist, which is a particular focus of modern ML in the face of sharply increasing trends in computation.

1.2 Related Work

As previously described, there has been a great deal of work on the sFFT; we refer the reader to the survey [18] for an overview. There has also been work on non-Fourier OP transforms. We break up our discussion below into discussion on the *sample complexity* (which as mentioned above is largely settled) and the *algorithmic complexity* (which remains largely open).

Sample complexity

The sample complexity of OP transforms $\mathbf{F}$ has been largely pinned down by the *compressed sensing* literature. For example, suppose that $\mathbf{F} \in \mathbb{R}^{N\times N}$ is any orthogonal and sufficiently flat matrix, in the sense that none of the entries of $\mathbf{F}$ are too large. Then a result of Haviv and Regev (sharpening of results by Bourgain, and Rudelson and Vershynin) shows that $m = O(k \log^2 k \log N)$ samples suffice to establish that the matrix $\Phi \in \mathbb{R}^{m\times N}$ (which is made up of m sampled rows from $\mathbf{F}^T$) has the *Restricted Isometry Property* (RIP) [5,20,34]. Finding $\hat{\mathbf{x}} = \mathbf{F}\mathbf{x}$ from samples of $\mathbf{F}$ of corresponds to the problem of finding an (approximately) k-sparse vector $\hat{\mathbf{x}}$ from the linear measurements $\Phi\hat{\mathbf{x}}$, which is precisely the compressed sensing problem. It is known that if Φ satisfies the RIP, then this can be solved (for example with ℓ_1 minimization) in time $N^{O(1)}$. On the other hand, recent results by Błasiok et al. show that this is essentially tight when $\mathbf{F}$ are certain Fourier matrices over constant sized prime finite fields, such as the Hadamard matrix, in that $O(k \log k \log N)$ queries (for a certain range of k) to $\mathbf{x}$ are needed to compute a k-sparse approximation of $\mathbf{F}\mathbf{x}$ [4].

Foucart and Rauhut [16] show that if the orthogonal polynomials satisfy a Bounded Orthogonal System (BOS) that are suitably flat, then if the m evaluation points λ_j are chosen uniformly at random proportional to the weight function w, then the $m \times N$ matrix Φ defined by normalizing $\mathbf{P}_N[i,j] = p_j(\lambda_i)$ appropriately satisfies the RIP with high probability provided that m has an appropriate dependence on N, k, ϵ, and the flatness of the matrix, and this again gives an $N^{O(1)}$-time algorithm to solve the sparse recovery problem.

Rauhut and Ward [32] show that for Jacobi polynomial transforms if the evaluation points were picked according to the *Chebyshev measure*, then with $O(k \operatorname{polylog} N)$ random measurements, the corresponding matrix has the RIP (note that the Foucart and Rauhut sample the evaluation points according to the measure of orthogonality for the Jacobi polynomials, which in general is *not* the Chebyshev measure). This result again does not give a sub-linear time algorithm but was used in the result of [21] which we describe below.

While these approaches can give near-optimal sample complexity, they do not give sublinear-time algorithms. In fact, it is faster to compute $\hat{\mathbf{x}}$ exactly by computing $\mathbf{F}\mathbf{x}$, if we care only about the running time and not about sample complexity [15]. Thus, we turn our attention to sublinear-time algorithms.

Sublinear-time algorithms for OP transforms

There have been several works generalizing and building on the sFFT results mentioned above. One direction is to the multi-dimensional DFT (for example in [23, 27]). Another direction is to apply the sFFT framework to orthogonal polynomials with similar structure. One example is Chebyshev polynomials and the Discrete Cosine Transform (DCT). It was observed in [21] that this can be reduced to sFFT in a black box manner, solving the sparse recovery problem for Chebyshev polynomials and the DCT. A second example of OP transforms which can essentially be reduced to the sFFT is *Legendre polynomials.* Hu et al. [21] seek to recover an unknown k-term Legendre polynomial (with highest magnitude degree limited to be $N/2$), defined on $[-1, 1]$, from samples. They give a sublinear two-phase algorithm: in the first phase, they reduce k-sparse-Legendre to sFFT to identify a set of candidate Legendre polynomials. The second phase uses the RIP result for BOS to produce a matrix that is used to estimate the coefficients of the candidate Legendre polynomials. We note that in this work the setting is naturally continuous, while ours is discrete.

Choi et al. [10, 11] study higher dimensions and obtain sublinear-time algorithms for more general harmonic expansions in multiple dimensions. These results complement our work. More precisely, that work shows how to use any algorithm for a univariate polynomial transform (the work in [11] needs these algorithms to have certain specific properties) to design an algorithm for a multi-variate polynomial transform where the multi-variate polynomials are products of univariate polynomials in the individual variables. Thus our improvements for univariate polynomial transforms can (potentially) be used with [10, 11].

Finally, there are sparse OP transforms based on Prony's method. The work [29] extends Prony's method to a very general setting, including Jacobi polynomials, and gives an algorithm that requires only $O(k)$ queries to recover exactly k-sparse polynomials. However, these general results work only for exact sparsity and are in general not robust to noise. There has been work extending and modifying these techniques to settings with noise (for example, [22, 30]), but to the best of our knowledge the only provable results for noise are for either the sFFT or closely related polynomial families. We note that [31] presents a Prony-like algorithm for Legendre and Gegenbauer polynomials and demonstrates empirically that this algorithm is robust to noise, although they do not address the question theoretically.

1.3 Technical overview

Our technical results have two main parts. First, inspired by existing approaches to the sFFT, we provide a general reduction from the k-sparse recovery problem to the 1-sparse recovery algorithm, which works for any family of OPs that is sufficiently "flat": that is, no entry of the matrix $\mathbf{F}$ is too large. Second, we provide a 1-sparse recovery algorithm for Jacobi polynomials. We give an overview of both parts below.

For what follows, let $\mathbf{F}$ be an orthogonal matrix. For simplicity in this overview we will assume that there is no noise. That is, we want to compute the exactly k-sparse $\hat{\mathbf{x}} = \mathbf{F}\mathbf{x}$ given query access to $\mathbf{x}$. However, we note that our final results do work for approximately k-sparse vectors $\hat{\mathbf{v}} = \hat{\mathbf{x}} + \hat{\mathbf{w}}$ provided that $\|\hat{\mathbf{w}}\|_2$ is sufficiently small.

1.3.1 Reduction to one-sparse recovery

We give a general reduction from the k-sparse recovery problem to the one-sparse recovery problem, which works for a broad class of OP families defined on a finite interval.[4] At a high level, the idea is as follows. Suppose that $\hat{\mathbf{x}} = \mathbf{F}\mathbf{x}$ is k-sparse and $\mathbf{b} \in \mathbb{R}^N$ is a "filter": at this stage it is helpful to think of it like a boxcar filter, so $\mathbf{b}$ is 1 on some interval I and zero outside of that interval. If we choose this interval randomly, we might hope to isolate a single "spike" of $\hat{\mathbf{x}}$ with $\mathbf{b}$: that is, we might hope that $\mathbf{D_b}\hat{\mathbf{x}}$ is one-sparse, where $\mathbf{D_b}$ is the diagonal matrix with $\mathbf{b}$ on the diagonal. Suppose that this occurs, so $\mathbf{y} = \mathbf{D_b}\hat{\mathbf{x}}$ is one sparse. In order to take advantage of this with a black-box solution to the one-sparse recovery problem $\hat{\mathbf{y}} = \mathbf{F}\mathbf{y}$, we would need query access to the vector $\mathbf{y} = \mathbf{F}^{-1}\mathbf{D_b}\hat{\mathbf{x}} = \mathbf{F}^{-1}\mathbf{D_b}\mathbf{F}\mathbf{x}$, while what we have is query access to $\mathbf{x}$. Thus, we would like to design $\mathbf{b}$ so that $\mathbf{F}^{-1}\mathbf{D_b}\mathbf{F}$ is *row-sparse*. This would allow us to query a position of $\mathbf{y} = \mathbf{F}^{-1}\mathbf{D_b}\hat{\mathbf{x}}$ using only a few queries from $\mathbf{x}$.

One of our main technical contributions is showing how to design such a vector $\mathbf{b}$, so that $\mathbf{b}$ approximates a boxcar filter and so that $\mathbf{F}^{-1}\mathbf{D_b}\mathbf{F}$ is row-sparse for *any* OP transform $\mathbf{F}$.

Then, given this filter, we can iteratively identify and subtract off "spikes" in $\hat{\mathbf{x}}$ until we have recovered the whole thing. Of course, the actual details are much more complicated than the sketch above. First, the one-sparse solver might have a bit of error, which will get propagated through the algorithm. Second, in our analysis the vector $\hat{\mathbf{x}}$ need not be exactly k-sparse. Third, $\mathbf{b}$ will only approximate a boxcar filter, and this is an additional source of error that needs to be dealt with. Complete details are in the full version of the paper.

For the reader familiar with the sFFT, this approach might look familiar: most sFFT algorithms work by using some sort of filter to isolate single spikes in an approximately sparse signal. Below, we highlight some of the challenges in extending this idea beyond the Fourier transform. Some of these challenges we have overcome, and one we have not (yet) overcome. We mention this last open challenge both because it explains the assumption we have to make on the sparsity structure of $\hat{\mathbf{x}}$, and also because we hope it will inspire future work.

Challenge 1: Choice of filter

One key difficulty in extending sFFT algorithms to general orthogonal polynomials is that the filters used in the sFFT approach are very specific to the Fourier transform. Indeed, much of the progress that has been made on that problem has been due to identifying better and better choices of filter specialized to the Fourier transform. In order to find filters that work for *any* OP family, we take a different approach and construct a filter out of low-degree Chebyshev polynomials. Then we use the orthogonality properties of the OP family to guarantee that $\mathbf{F}^{-1}\mathbf{D_b}\mathbf{F}$ has the desired sparsity properties.

Challenge 2: Explicit black-box reduction

Because our goal is generality (to as broad a class of OPs as possible), we give an explicit reduction that uses a 1-sparse solution as a black box. To the best of our knowledge, existing work on the sFFT does not explicitly do this: a reduction of this flavor is certainly implicit in many of these works, and even explicitly given as intuition, but we are not aware of an sFFT algorithm which actually uses a 1-sparse recovery algorithm as a black box.

[4] We note that our results do not (yet) work for the case when the orthogonality is defined over an infinite interval. In particular, our reduction does not work for the Hermite and Laguerre polynomials.

Challenge 3: Equi-spaced evaluation points

The evaluations points in DFT and the DCT are equispaced (in the angular space). This fact is crucially exploited in sFFT algorithms (as well as the reduction of DCT to DFT – see the full version of this paper for more details on the reduction). Unfortunately, the roots of Jacobi polynomials are no longer equally spaced. However, it is known that the roots of Jacobi polynomials are "spread out" (in a sense made below precise in Definition 2.2), and we show that this property is enough for our reduction. In fact, our reduction from k-sparse recovery to 1-sparse recovery works generally for any "flat" OP family with "spread out" roots.

(Open) Challenge 4: Permuting the coordinates of $\hat{\mathbf{x}}$

In the approach described above, we hoped that an interval I would "isolate" a single spike. In the sFFT setting, this can be achieved through a permutation of the coordinates of $\hat{\mathbf{x}}$. In our language, in the sFFT setting it is possible to define a random (enough) permutation matrix $\mathbf{P}$ so that $\mathbf{P}\hat{\mathbf{x}}$ has permuted coordinates, and so that $\mathbf{F}^{-1}\mathbf{D_b}\mathbf{P}\mathbf{F}$ is row-sparse – this argument crucially exploits the fact that the roots of unity are equispaced in the angle space. This means that not only can we sample from the one-sparse vector $\mathbf{D_b}\hat{\mathbf{x}}$, but also we can sample from $\mathbf{D_b}\mathbf{P}\hat{\mathbf{x}}$, and then there is some decent probability that any given spike in $\hat{\mathbf{x}}$ is isolated by $\mathbf{b}$. However, we have not been able to come up with (an approximation to) such a $\mathbf{P}$ that works in the general OP setting. This explains why we require the assumption that the support of $\hat{\mathbf{x}}$ be reasonably "spread out," so that we can hope to isolate the spikes by $\mathbf{b}$. This assumption is made precise in Definition 2.3. We note that if such a $\mathbf{P}$ were found in future work, this would immediately lead to an improved k-sparse recovery result for Jacobi polynomials, which would work for arbitrary sparse signals $\hat{\mathbf{x}}$.

1.3.2 A one-sparse recovery algorithm for Jacobi polynomials

With the reduction complete, to obtain a k-sparse recovery algorithm for general Jacobi polynomials we need to solve the one-sparse case. We give an overview of the basic idea here, with full details in the full version of the paper. First, we note that via well-known approximations of Jacobi polynomials [35], one can approximate the evaluation of any Jacobi polynomial at a point in $(-1, 1)$ by evaluating the cosine function at an appropriate angle. Using some standard local error-correcting techniques (for example, computing $\cos(A)$ via $\cos A = \frac{\cos(A+B)+\cos(A-B)}{2\cos B}$ for a random B), we reduce the 1-sparse recovery problem to computing some unknown value θ, corresponding to the index of the spike, from noisy values of $\cos(w\theta)$ for some integers $w \geq 1$, corresponding to evaluations of the Jacobi polynomials for this index. Since the reduction is approximate, some care has to be taken to handle some corner cases where the approximation does not hold. In particular, we have to figure out for which real numbers $y \in [0, N)$ does its orbit $\langle xy \rangle$ for $x \in \mathbb{Z}_N$ have small order. We give a result to handle this, which to the best of our knowledge (and somewhat surprisingly) seems to be new.[5] With this out of the way, our algorithm to compute the value of θ from the evaluations $\cos(w\theta)$ is based on the following idea. Assuming we already know $\cos(\theta)$ up to $\pm\epsilon$, we get a noisy estimate of θ (which lives in the range $\arccos(\cos(\theta) \pm \epsilon)$) and then use the evaluations at $w > 1$ to "dilate" the range where we know θ lies, reducing ϵ. We

[5] We thank Stefan Steinerberger for showing us a much simpler proof than our original more complicated proof, which also gave worse parameters.

proceed iteratively until the region of uncertainty is small enough that there are only $O(1)$ possibilities remaining, which we then prune out using the fact that $\mathbf{F}$ is orthogonal and flat, in the sense that none of its entries are too large. (We note that proving $\mathbf{F}$ is flat needs a bit of care. In particular, we need a sharper bound on Jacobi polynomials (than the cosine approximation mentioned above) in terms of Bessel functions to prove that *all* entries of $\mathbf{F}$ are small.) Similar ideas have been used for 1-sparse recovery for the DFT (for example, in [19]), although our situation is more complicated than the DFT because working with cosines instead of complex exponentials means that we lose sign information about θ along the way (though it is similar in spirit to the one-sparse recovery algorithm for DFT in [19]).

2 Background and Preliminaries

2.1 Notation

We use bold lower-case letters $(\mathbf{x}, \mathbf{y})$ for vectors and bold upper-case letters $(\mathbf{P}, \mathbf{F})$ for matrices. Non-bold notation x, y, U is used for scalars in $\mathbb{R}$. In general, if there is a given transform $\mathbf{F}$ we are considering, then the notation $\hat{\mathbf{x}} \in \mathbb{R}^N$ indicates $\mathbf{F} \cdot \mathbf{x}$. We use the notation $\mathbf{x}[i]$ or $\mathbf{X}[i, j]$ to index into a vector or matrix, respectively. All of our vectors and matrices are 0-indexed, i.e. the entries of a vector $\mathbf{x} \in \mathbb{R}^N$ are $\mathbf{x}[0], \ldots, \mathbf{x}[N-1]$. We use $[N]$ to denote the set $\{0, \ldots, N-1\}$. Given a subset $S \subset [N]$, we will denote the complement set (i.e. $[N] \setminus S$) by S^c.

Given a vector $\mathbf{x} \in \mathbb{R}^N$ and an integer $1 \leq s \leq N$, we define $\textsc{large}(s, \mathbf{x})$ to be the magnitude of the sth largest value in $\mathbf{x}$ (by absolute value).

For any vector $\mathbf{u} \in \mathbb{R}^N$, we define $\mathbf{D}_{\mathbf{u}} \in \mathbb{R}^{N \times N}$ as the diagonal matrix with $\mathbf{u}$ on its diagonal. For a diagonal matrix $\mathbf{D}$, and any real α we denote $\mathbf{D}^\alpha$ to denote the diagonal matrix with the (i, i) entry being $(\mathbf{D}[i, i])^\alpha$. Given a vector $\mathbf{x} \in \mathbb{R}^N$ and set $S \subseteq [N]$, $\mathbf{x}_S$ denotes the vector $\mathbf{x}$ where all entries out of S are masked to 0. For $\mathbf{x} \in \mathbb{R}^N$, $\mathrm{supp}(\mathbf{x}) \subseteq [N]$ denotes the support (i.e. the set of non-zero positions) of $\mathbf{x}$.

We use $x \pm h$ to refer to either the interval $[x - h, x + h]$ or a point in this interval, whichever is clear from context. Similarly, if S is an interval $[a, b]$ then $S \pm h$ is the interval $[a - h, b + h]$.

When stating algorithms, we use superscript notation to denote query access. That is $\mathcal{A}^{(\mathbf{x})}(\mathbf{z})$ takes input $\mathbf{z}$ and has query access to $\mathbf{x}$.

We use the notation $f(n) \lesssim g(n)$ to mean that there is some constant C so that, for sufficiently large $n \geq n_0$, $f(n) \leq Cg(n)$.

The notation $\mathcal{J}_\square$ means $\mathcal{J}_j$ for all indices j.

2.2 Orthogonal Polynomials

For the remainder of this paper, we consider polynomials $p_0(X), p_1(X), \ldots$ that form a normalized orthogonal polynomial family with respect to some compactly supported measure $w(X)$. By suitably scaling and translating X, we can ensure that the orthogonality is on $[-1, 1]$.[6] In particular $\deg(p_i) = i$ and for any $i, j \geq 0$,

$$\int_{-1}^{1} p_i(X) p_j(X) w(X) dX = \delta_{i,j}, \tag{1}$$

where $\delta_{i,j} = 1$ if $i = j$ and 0 otherwise.

[6] See footnote 4.

Then for given N evaluation points $\lambda_0, \dots \lambda_{N-1}$, define the orthogonal polynomial transform $\mathbf{P}_N$ as follows. For any $0 \le i, j < N$, we have

$$\mathbf{P}_N[i,j] = p_j(\lambda_i).$$

In other words, the rows of $\mathbf{P}_N$ are indexed by the evaluation points and the columns are indexed by the polynomials.

For the rest of the paper, assume $\lambda_0 \le \lambda_1 \le \cdots \le \lambda_{N-1}$ are the roots of $p_N(X)$. Then it is well-known (see e.g. [35]) that

- The roots lie in the support of the measure (i.e. $\lambda_i \in [-1,1]$) and are distinct (i.e. $\lambda_0 < \lambda_1 < \cdots < \lambda_{N-1}$).
- There exists Gaussian quadrature weights $w_\ell = \frac{1}{\sum_{j=0}^{N-1} p_j(\lambda_\ell)^2}, i = 0, \dots, N-1$ such that for any polynomial $f(X)$ of degree at most $2N-1$,

$$\int_{-1}^{1} f(X) w(X) dX = \sum_{\ell=0}^{N-1} f(\lambda_\ell) \cdot w_\ell. \tag{2}$$

We are now ready to define the orthogonal matrix corresponding to $\mathbf{P}_N$ that we deal with in this paper:

▶ **Definition 2.1.** *Let $p_0(X), \dots, p_{N-1}(X), \dots$ be an orthogonal polynomial family, $\lambda_0, \dots, \lambda_{N-1}$ be the roots of $p_N(X)$, and $w_0, \dots, w_{N-1}$ be the Gaussian quadrature weights. Define $\mathbf{D}_\mathbf{w}$ to be the diagonal matrix with $w_0, \dots, w_{N-1}$ on its diagonal, and*

$$\mathbf{F}_N = \mathbf{D}_\mathbf{w}^{\frac{1}{2}} \mathbf{P}_N.$$

Note that by (1) and (2),

$$\mathbf{F}_N^T \mathbf{F}_N = \mathbf{P}_N^T \mathbf{D}_\mathbf{w} \mathbf{P}_N = \mathbf{I}_N,$$

so $\mathbf{F}_N$ is an orthogonal matrix. In particular,

$$\mathbf{P}_N^T \mathbf{D}_\mathbf{w} \mathbf{P}_N[i,j] = \sum_{k=0}^{N-1} p_i(\lambda_k) w_k p_j(\lambda_k) = \int_{-1}^{1} p_i(X) p_j(X) w(X)\, dX = \delta_{i,j}.$$

Note that since $\mathbf{F}_N$ is orthogonal, by definition we have

$$\mathbf{F}_N^{-1} = \mathbf{F}_N^T.$$

2.2.1 Jacobi Polynomials and Special Cases

In this section we define Jacobi Polynomials, our main object of interest, and point out a few special cases. We note that families of named orthogonal polynomials $\{p_i(X)\}$ are sometimes defined through different means, hence are normalized differently up to constants. The corresponding discrete orthogonal polynomial transform (e.g. Discrete Legendre Transform) frequently refers to multiplication by $\mathbf{P}$ instead of $\mathbf{F}$. In these cases, the transform satisfies $\mathbf{P}_N^T \mathbf{D}_\mathbf{w} \mathbf{P}_N = \mathbf{D}$ for a diagonal matrix $\mathbf{D}$ corresponding to the normalization. The transform $\mathbf{F}_N = \mathbf{D}_\mathbf{w}^{\frac{1}{2}} \mathbf{P} \mathbf{D}^{-\frac{1}{2}}$ we consider (note that this matrix is indeed orthogonal) is thus equivalent up to diagonal multiplication.

Jacobi polynomials

Jacobi polynomials are indexed by two parameters $\alpha, \beta > -1$ and these are polynomials $\left\{P_j^{(\alpha,\beta)}\right\}_{j\geq 0}$ that are orthogonal with respect to the measure

$$w^{(\alpha,\beta)}(X) = (1-X)^\alpha \cdot (1+X)^\beta$$

in the range $[-1,1]$. This definition is not normalized, in the sense that we have $\mathbf{P}_N^T \mathbf{D}_\mathbf{w} \mathbf{P}_N = \mathbf{D}$, where

$$\mathbf{D}[j,j] = \frac{2^{\alpha+\beta+1}}{2j+\alpha+\beta+1} \cdot \frac{\Gamma(j+\alpha+1)\Gamma(j+\beta+1)}{\Gamma(j+1)\Gamma(j+\alpha+\beta+1)}$$

(see [35, Pg. 68, (4.3.3)]).

We record three well-known special cases: Chebyshev polynomials (of the first kind) are special case of $\alpha = \beta = -\frac{1}{2}$ and Legendre polynomials are the special case of $\alpha = \beta = 0$ (up to potentially a multiplicative factor that could depend on the degree j). Another notable special case of Jacobi polynomials are the *Gegenbauer* or *ultraspherical polynomials* ($\alpha = \beta$). Our results hold for all Jacobi polynomials with $\alpha, \beta \geq -\frac{1}{2}$, which include essentially all named special cases of Jacobi polynomials used in practice.

Chebyshev polynomials of the 1st kind

The Chebyshev polynomials of the 1st kind are orthogonal with respect to the weight measure $w(X) = (1-X^2)^{-\frac{1}{2}}$.

The normalized transform $\mathbf{F}_N$ has the closed form

$$\mathbf{F}_N[i,j] = \begin{cases} \sqrt{\frac{1}{N}} & j = 0 \\ \sqrt{\frac{2}{N}} \cdot \cos\left[\frac{\pi}{N} j \left(i+\frac{1}{2}\right)\right] & j = 1, \ldots, N-1. \end{cases}$$

This is a variant of the Discrete Cosine Transform (DCT-III, or the inverse DCT). It is well-known that the DCT-III can be "embedded" into a DFT of twice the dimension, and we work out some of the details of how to use the sparse FFT to compute a sparse DCT in the full version of the paper.

Legendre polynomials

Legendre polynomials are orthogonal with respect to the uniform measure i.e. $w(X) = 1$ and play a critical role in multipole expansions of potential functions (whether electrical or gravitational) in spherical coordinates. They are also important for solving Laplace's equation in spherical coordinates.

2.2.2 Roots of Orthogonal Polynomials

Since $\lambda_i \in [-1,1]$ for all i, there is a unique $\theta_i \in [0,\pi]$ such that $\lambda_i = \cos\theta_i$. Our reduction holds for orthogonal polynomials that have roots that are "well-separated" in this angle space:

▶ **Definition 2.2.** *Let $0 < C_0 < C_1$. A family of orthogonal polynomials $p_0(X), p_1(X), \ldots$ is* (C_0, C_1, γ_0)-dense *if for all large enough d, the following holds.*

Let $\lambda_0, \ldots, \lambda_{d-1}$ be the roots of p_d, and $\theta_i = \arccos \lambda_i$. Then for any $i \in [d]$, for any $\gamma \geq \gamma_0/d$:

$$C_0 \gamma d \leq \left| \{\theta_0, \ldots, \theta_{d-1}\} \cap \left[\theta_i - \frac{\gamma}{2}, \theta_i + \frac{\gamma}{2}\right] \right| \leq C_1 \gamma d.$$

It turns out that any family of Jacobi polynomials has the required property: their roots are spaced out such that θ_ℓ is close to $\ell\pi/N$.

2.3 Sparse Recovery Problem

We will consider approximately k-sparse vectors $\hat{\mathbf{v}} = \hat{\mathbf{x}} + \hat{\mathbf{w}}$, where $\hat{\mathbf{x}}$ is k-sparse and $\|\hat{\mathbf{w}}\|_2$ is sufficiently small. We will require that $\hat{\mathbf{x}}$ has a "spread out" support, defined as follows.

▶ **Definition 2.3.** *Let $k \in [N]$ and $0 \leq \sigma < 1$. We say that a vector $\mathbf{x} \in \mathbb{R}^N$ is* (k, σ)-sparsely separated *if there are k non-zero locations in $\mathbf{x}$ and any two non-zero locations are more than σN indices apart.*

It is not hard to see that a vector $\mathbf{x}$ with random support of size k is, with constant probability, $\left(k, \Omega\left(\frac{1}{k^2}\right)\right)$-sparsely separated.

In our reduction, we will reduce the k-sparse recovery problem to the special case of $k = 1$. Next, we define some notation for the 1-sparse case.

▶ **Definition 2.4.** *We say that the matrix $\mathbf{F}_N$ has an $(N, \epsilon, \delta, \mu)$ one-sparse recovery algorithm with query complexity $Q(N, \epsilon, \delta, \mu)$ and time complexity $T(N, \epsilon, \delta, \mu)$ if there exists an algorithm $\mathcal{A}$ with the properties below:*

For all $\mathbf{y}$ so that $\hat{\mathbf{y}} = \mathbf{F}_N \mathbf{y}$ can be decomposed as

$$\hat{\mathbf{y}} = \tilde{\mathbf{y}} + \mathbf{w},$$

where $\tilde{\mathbf{y}} = v \cdot \mathbf{e}_h$ is 1-sparse and

$$\|\mathbf{w}\|_2 \leq \epsilon \, |v| \,,$$

we have:

1. *$\mathcal{A}$ makes at most $Q(N, \epsilon, \delta, \mu)$ queries into $\mathbf{y} = \mathbf{F}_N^{-1}(v \cdot \mathbf{e}_h + \mathbf{w})$.*
2. *With probability at least $1 - \mu$, $\mathcal{A}$ outputs $\tilde{v} \cdot \mathbf{e}_h$ with $|v - \tilde{v}| \leq \delta \, |v|$ in time $T(N, \epsilon, \delta, \mu)$.*

Pre-processing time

Our algorithm requires some pre-processing of $\mathbf{F}_N$. Our pre-processing step involves computing the roots $\lambda_1, \ldots, \lambda_N$ of p_N and storing them in an appropriate data structure, and additionally forming and storing some matrices that we will use in our algorithm. Finding the roots and creating the data structure can be done in time $\mathrm{poly}(N)$, and the rest of the pre-processing step also takes time $\mathrm{poly}(N)$. We note that this is an up-front cost that needs to be only paid once.

Precision

We note that we need to make certain assumptions on size of the entries in $\hat{\mathbf{v}}$ since otherwise we would not even be able to read coefficients that are either too large or too small and need $\Omega(\log N)$ bits to represent. Towards this end we will make the standard assumption that $\|\hat{\mathbf{v}}\|_2 = 1$. In particular, this allows us to ignore any coefficients that are smaller than say $\frac{1}{N}$

since their contribution to $\|\hat{\mathbf{v}}\|_2$ is at most $\frac{1}{\sqrt{N}}$, which will be too small for our purposes.[7] In particular, this implies that we only have to deal with numbers that need $O(\log N)$ bits and as is standard in the RAM model, basic arithmetic operations on such numbers can be done in $O(1)$ time. We will implicitly assume this for the rest of the paper (except in the proof of one lemma, where we will explicitly make use of this assumption).

3 Results

In this section we state our main results. These results follow from more detailed versions which are stated in the full version of the paper.

We start off with our main result for Jacobi polynomials. We state an informal version here, and refer the reader to the full version of the paper for the formal result.

▶ **Theorem 3.1** (General Sparse Recovery for Jacobi Polynomial Transform, Informal). *Fix arbitrary parameters $\alpha, \beta \geq -\frac{1}{2}$ for Jacobi polynomials and let $\mathbf{J}_N^{(\alpha,\beta)}$ be the $N \times N$ orthogonal matrix that arises from it as in Definition 2.1. Then there is an algorithm* Recover *that does the following. Let $\mathbf{v} = \mathbf{x} + \mathbf{w}$ where $\hat{\mathbf{x}} = \mathbf{J}_N^{(\alpha,\beta)}\mathbf{x}$ is $(k, C_1/k^2)$-sparsely separated, and suppose that $\|\hat{\mathbf{w}}\|_2 \lesssim \delta \min_{h \in \mathrm{supp}(\hat{\mathbf{x}})} |\hat{\mathbf{x}}[h]|$. Then with probability at least 0.99,* Recover *outputs $\hat{\mathbf{z}}$ such that*

$$\|\hat{\mathbf{x}} - \hat{\mathbf{z}}\|_2 \lesssim \delta \|\hat{\mathbf{x}}\|_2,$$

with $\mathrm{poly}\left(\frac{k \log N}{\delta}\right)$ *queries and running time* $\mathrm{poly}\left(\frac{k \log N}{\delta}\right)$.

▶ Remark 3.2. The requirement on the noise term might be bad if one entry of $\hat{\mathbf{x}}$ is extremely small compared to the rest. However in this case we can decrease k and add the very small entries of $\hat{\mathbf{x}}$ to the noise term $\hat{\mathbf{w}}$ resulting in a potentially better guarantee. We note that our algorithm iteratively finds the large components of $\hat{\mathbf{x}}$ and in fact has a mechanism for stopping early when all of the "large-enough" entries have been found.

▶ Remark 3.3. The $(k, O(1/k^2))$-sparsely separated requirement is chosen to reflect the separation of a random k-sparse vector (c.f. comment below Definition 2.3). Smaller amounts of sparse separation are acceptable, which translate accordingly into the query and time complexity. The full dependence is in the complete result in the full version of the paper.

To prove the above result, we first reduce the k-sparse recovery problem to 1-sparse recovery problem, in the presence of a small amount of noise. Next, we present an informal statement of our reduction.

▶ **Theorem 3.4** (Main Reduction, Informal). *Let $p_1, \ldots, p_N$ be a (C_0, C_1, γ_0)-dense orthogonal polynomial family, and let $\mathbf{F}_N$ be the $N \times N$ orthogonal matrix that arises from it as in Definition 2.1. Suppose that $|\mathbf{F}_N^{-1}[i,j]| \lesssim 1/\sqrt{N}$ for all $i, j \in [N]$. Suppose that for some sufficiently small $\delta > 0$, $\mathbf{F}_N$ has a $\left(N, O(\delta), \delta, O(C_0/k^2)\right)$ one-sparse recovery algorithm with query complexity Q and running time T.*

Then there is an algorithm Recover *that does the following. Let $\mathbf{v} = \mathbf{x} + \mathbf{w}$ where $\hat{\mathbf{x}} = \mathbf{F}_N\mathbf{x}$ is $(k, C_1/k^2)$-sparsely separated, and suppose that $\|\hat{\mathbf{w}}\|_2 \lesssim \delta \min_{h \in \mathrm{supp}(\hat{\mathbf{x}})} |\hat{\mathbf{x}}[h]|$. Then with probability at least 0.99,* Recover *outputs $\hat{\mathbf{z}}$ so that*

$$\|\hat{\mathbf{x}} - \hat{\mathbf{z}}\|_2 \lesssim \delta \|\hat{\mathbf{x}}\|_2,$$

with $\mathrm{poly}(k/\delta C_0)Q$ *queries and running time* $\mathrm{poly}(k/\delta C_0)T$.

[7] More generally, we can ignore smaller coefficients as long as they are polynomial sized in N.

The final algorithmic piece missing from the result above is the algorithm for 1-sparse recovery. We provide this missing piece for Jacobi polynomials:

▶ **Theorem 3.5** (1-Sparse Recovery for Jacobi Transform, Informal)**.** *There exists a universal constant C such that the following holds. Consider the Jacobi transform for any fixed parameters $\alpha, \beta \geq -\frac{1}{2}$. There exists an $(N, \epsilon, C \cdot \epsilon, \gamma)$ 1-sparse recovery algorithm for the Jacobi transform that makes* $\text{poly}\left(\log\left(\frac{N}{\gamma}\right) \cdot \frac{1}{\epsilon}\right)$ *queries and takes time* $\text{poly}\left(\log\left(\frac{N}{\gamma}\right) \cdot \frac{1}{\epsilon}\right)$.

4 Open Questions

To conclude, we list a few questions left open by our work.

1. First, it is natural to try and improve our k-sparse recovery algorithm to work for arbitrary k-sparse support, rather than "well-separated" supports. One natural way to do this is to address the fourth (open) challenge in Section 1.3 for a general class of OPs.
2. Second, we could hope to handle a more general class of noise $\hat{\mathbf{w}}$ than we currently do. One could hope to handle *any* vector $\mathbf{v}$, with an error guarantee that degrades smoothly with the ℓ_2 norm of the "tail" of $\mathbf{v}$. There is a long list of work on "de-noising" the contribution of the "head" to the "tail" in the sFFT literature that could potentially be useful here [23–27].
3. Third, we would like to extend our results to hold for OPs defined over infinite intervals (e.g. Hermite and Laguerre polynomials).
4. Fourth, we would like to solve the sparse recovery for $\mathbf{F}^T$ (where $\mathbf{F}$ is as in Definition 2.1): i.e. given query access to $\mathbf{x}$ figure out a good k-sparse approximation to $\mathbf{F}^T\mathbf{x}$ (recall that $\mathbf{F}^{-1} = \mathbf{F}^T$). (Note that this problem can be equivalently stated as follows: given query access to $\mathbf{Fy}$, compute a good k-sparse approximation to $\mathbf{y}$.) Currently our results do not solve this problem since we cannot show that the existence of a filter $\mathbf{b}$ such that $\mathbf{F}\mathbf{D}_\mathbf{b}\mathbf{F}^T$ is row-sparse. Note that this is not an issue for DFT since it is symmetric.
5. Finally, we would like to reduce the exponent on k in our final runtime. In particular, for the case of random k-sparse support, the dependence on k in the runtime for Jacobi transform is k^8. We note that we have not tried too hard to optimize the constants though we believe even getting a quadratic dependence on k with our framework would be challenging. We would like to stress that the majority of the work in the sFFT literature has been to make the dependence on k be linear and for such results, it seems very unlikely that a generic reduction from k-sparse recovery to 1-sparse recovery would work. In other words, using the knowledge about the 1-sparse recovery algorithm for DFT seems necessary to get a overall k-sparse FFT with running time $k\text{poly}(\log n)$.

References

1 https://en.wikipedia.org/wiki/Approximation_theory#Chebyshev_approximation.
2 https://en.wikipedia.org/wiki/Jacobi_polynomials#Differential_equation.
3 http://www.chebfun.org.
4 Jaroslaw Blasiok, Patrick Lopatto, Kyle Luh, Jake Marcinek, and Shravas Rao. An improved lower bound for sparse reconstruction from subsampled hadamard matrices. In David Zuckerman, editor, *60th IEEE Annual Symposium on Foundations of Computer Science, FOCS 2019, Baltimore, Maryland, USA, November 9-12, 2019*, pages 1564–1567. IEEE Computer Society, 2019. doi:10.1109/FOCS.2019.00091.
5 Jean Bourgain. *An Improved Estimate in the Restricted Isometry Problem*, pages 65–70. Springer International Publishing, Cham, 2014. doi:10.1007/978-3-319-09477-9_5.

6 James Bremer, Qiyuan Pang, and Haizhao Yang. Fast Algorithms for the Multi-dimensional Jacobi Polynomial Transform. *arXiv e-prints*, January 2019. arXiv:1901.07275.

7 James Bremer and Haizhao Yang. Fast algorithms for Jacobi expansions via nonoscillatory phase functions. *arXiv e-prints*, March 2018. arXiv:1803.03889.

8 Volkan Cevher, Michael Kapralov, Jonathan Scarlett, and Amir Zandieh. An adaptive sublinear-time block sparse fourier transform. In Hamed Hatami, Pierre McKenzie, and Valerie King, editors, *Proceedings of the 49th Annual ACM SIGACT Symposium on Theory of Computing, STOC 2017, Montreal, QC, Canada, June 19-23, 2017*, pages 702–715. ACM, 2017. doi:10.1145/3055399.3055462.

9 Xue Chen, Daniel M. Kane, Eric Price, and Zhao Song. Fourier-sparse interpolation without a frequency gap. In Irit Dinur, editor, *IEEE 57th Annual Symposium on Foundations of Computer Science, FOCS 2016, 9-11 October 2016, Hyatt Regency, New Brunswick, New Jersey, USA*, pages 741–750. IEEE Computer Society, 2016. doi:10.1109/FOCS.2016.84.

10 Bosu Choi, Mark Iwen, and Felix Krahmer. Sparse Harmonic Transforms: A New Class of Sublinear-time Algorithms for Learning Functions of Many Variables. *arXiv 1808.04932*, 2018. arXiv:1808.04932.

11 Bosu Choi, Mark Iwen, and Toni Volkmer. Sparse harmonic transforms ii: Best s-term approximation guarantees for bounded orthonormal product bases in sublinear-time. *arXiv preprint*, 2019. arXiv:1909.09564.

12 John D. Cook. Orthogonal polynomials and gaussian quadrature. URL: https://www.johndcook.com/OrthogonalPolynomials.pdf.

13 Tri Dao, Christopher M De Sa, and Christopher Ré. Gaussian quadrature for kernel features. In I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, editors, *Advances in Neural Information Processing Systems 30*, pages 6107–6117. Curran Associates, Inc., 2017. URL: http://papers.nips.cc/paper/7191-gaussian-quadrature-for-kernel-features.pdf.

14 Christopher De Sa, Albert Cu, Rohan Puttagunta, Christopher Ré, and Atri Rudra. A two-pronged progress in structured dense matrix vector multiplication. In *Proceedings of the Twenty-Ninth Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 1060–1079. SIAM, 2018.

15 J. R. Driscoll, D. M. Healy, Jr., and D. N. Rockmore. Fast discrete polynomial transforms with applications to data analysis for distance transitive graphs. *SIAM J. Comput.*, 26(4):1066–1099, August 1997. doi:10.1137/S0097539792240121.

16 Simon Foucart and Holger Rauhut. *A Mathematical Introduction to Compressive Sensing*. Springer Science & Business Media, August 2013.

17 A. C. Gilbert, S. Guha, P. Indyk, S. Muthukrishnan, and M. Strauss. Near-optimal sparse fourier representations via sampling. In *Proceedings of the Thiry-fourth Annual ACM Symposium on Theory of Computing*, STOC '02, pages 152–161, New York, NY, USA, 2002. ACM. doi:10.1145/509907.509933.

18 Anna C Gilbert, Piotr Indyk, Mark Iwen, and Ludwig Schmidt. Recent Developments in the Sparse Fourier Transform. *IEEE Signal Processing Magazine*, 2014. doi:10.1109/MSP.2014.2329131.

19 Haitham Hassanieh, Piotr Indyk, Dina Katabi, and Eric Price. Nearly optimal sparse fourier transform. In *Proceedings of the forty-fourth annual ACM symposium on Theory of computing*, pages 563–578. ACM, 2012.

20 Ishay Haviv and Oded Regev. The restricted isometry property of subsampled fourier matrices. In *Geometric aspects of functional analysis*, pages 163–179. Springer, 2017.

21 Xianfeng Hu, Mark Iwen, and Hyejin Kim. Rapidly computing sparse legendre expansions via sparse fourier transforms. *Numerical Algorithms*, 74(4), 2017.

22 Y. Hua and T. K. Sarkar. Matrix pencil method for estimating parameters of exponentially damped/undamped sinusoids in noise. *IEEE Transactions on Acoustics, Speech, and Signal Processing*, 38(5):814–824, May 1990. doi:10.1109/29.56027.

23 Piotr Indyk and Michael Kapralov. Sample-optimal fourier sampling in any constant dimension. In *2014 IEEE 55th Annual Symposium on Foundations of Computer Science*, pages 514–523. IEEE, 2014.

24 Piotr Indyk, Michael Kapralov, and Eric Price. (nearly) sample-optimal sparse fourier transform. In *Proceedings of the twenty-fifth annual ACM-SIAM symposium on Discrete algorithms*, pages 480–499. Society for Industrial and Applied Mathematics, 2014.

25 Michael Kapralov. Sparse fourier transform in any constant dimension with nearly-optimal sample complexity in sublinear time. In Daniel Wichs and Yishay Mansour, editors, *Proceedings of the 48th Annual ACM SIGACT Symposium on Theory of Computing, STOC 2016, Cambridge, MA, USA, June 18-21, 2016*, pages 264–277. ACM, 2016. doi:10.1145/2897518.2897650.

26 Michael Kapralov. Sample efficient estimation and recovery in sparse FFT via isolation on average. In Chris Umans, editor, *58th IEEE Annual Symposium on Foundations of Computer Science, FOCS 2017, Berkeley, CA, USA, October 15-17, 2017*, pages 651–662. IEEE Computer Society, 2017. doi:10.1109/FOCS.2017.66.

27 Michael Kapralov, Ameya Velingker, and Amir Zandieh. Dimension-independent sparse fourier transform. In *Proceedings of the Thirtieth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2019, San Diego, California, USA, January 6-9, 2019*, pages 2709–2728, 2019. doi:10.1137/1.9781611975482.168.

28 Cameron Musco. Chebyshev polynomials in tcs and algorithm design. URL: http://www.cameronmusco.com/personal_site/pdfs/retreatTalk.pdf.

29 Thomas Peter and Gerlind Plonka. A generalized prony method for reconstruction of sparse sums of eigenfunctions of linear operators. *Inverse Problems*, 29(2):025001, January 2013. doi:10.1088/0266-5611/29/2/025001.

30 Daniel Potts and Manfred Tasche. Parameter estimation for exponential sums by approximate prony method. *Signal Processing*, 90(5):1631–1642, 2010. Special Section on Statistical Signal and Array Processing. doi:10.1016/j.sigpro.2009.11.012.

31 Daniel Potts and Manfred Tasche. Reconstruction of sparse legendre and gegenbauer expansions. *BIT Numerical Mathematics*, 56(3):1019–1043, 2016.

32 Holger Rauhut and Rachel Ward. Sparse legendre expansions via ℓ_1-minimization. *J. Approx. Theory*, 164(5):517–533, May 2012. doi:10.1016/j.jat.2012.01.008.

33 T.J. Rivlin. *An Introduction to the Approximation of Functions*. Blaisdell book in numerical analysis and computer science. Dover Publications, 1981. URL: https://books.google.com/books?id=wtS2xm8ggA4C.

34 Mark Rudelson and Roman Vershynin. On sparse reconstruction from fourier and gaussian measurements. *Communications on Pure and Applied Mathematics*, 61(8):1025–1045, 2008. doi:10.1002/cpa.20227.

35 G. Szegö. *Orthogonal Polynomials*. Number v. 23 in American Mathematical Society colloquium publications. American Mathematical Society, 1975. URL: https://books.google.com/books?id=3hcW8HBh7gsC.

36 William J. Tango. The circle polynomials of zernike and their application in optics. *Applied physics*, 13(4):327–332, August 1977. doi:10.1007/BF00882606.

37 Anna Thomas, Albert Gu, Tri Dao, Atri Rudra, and Christopher Ré. Learning compressed transforms with low displacement rank. In *Advances in neural information processing systems*, pages 9052–9060, 2018.

38 Aaron Voelker, Ivana Kajić, and Chris Eliasmith. Legendre memory units: Continuous-time representation in recurrent neural networks. In *Advances in Neural Information Processing Systems*, pages 15544–15553, 2019.

39 Kun-Hsing Yu, Ce Zhang, Gerald J Berry, Russ B Altman, Christopher Ré, Daniel L Rubin, and Michael Snyder. Predicting non-small cell lung cancer prognosis by fully automated microscopic pathology image features. *Nature Communications*, 7, 2016.