We would like to thank the editor and reviewers for their time and valuable comments concerning our manuscript entitled "Fast Algorithms for the Multi-dimensional Jacobi Polynomial Transform". We feel the revised manuscript are vastly improved based on the comments of the reviewers. To ease the reviewers, we mark the changes in the manuscript with blue color text. Below, we list our answer to each of the reviewers' questions.

Reviewer 1: This article developed a fast algorithm for computing multi-dimensional Jacobi polynomial transforms. The paper's chief contribution is that it extended the basic idea of the method in [19] from one dimension to multi-dimensions and from $(a, b) \subset (-1/2, 1/2)$ to $(a, b) \subset (-1, 1)$. The paper is interesting and well written. The mathematical results are correct and the numerical results appear to be plausible and consistent. This paper deserves to be published in Applied and Computational Harmonic Analysis. However, the authors should address and clarify the following comments and questions before the paper can be published.

1. **Theoretical complexity.** For the Chebyshev-Jacobi and even Jacobi-Jacobi transform, several algorithms with provable quasi-linear complexity are known as follows [1,2,3].

   [1] Alex Townsend, Marcus Webb and Sheehan Olver, Fast polynomial trans- forms based on Toeplitz and Hankel matrices, Mathematics of Computation, 87 (2018), pp. 1913-1934.

   [2] Richard Mikael Slevinsky, On the use of Hahn's asymptotic formula and stabilized recurrence for a fast, simple and stable Chebyshev-Jacobi transform, IMA Journal of Numerical Analysis, 38(2018), pp. 102-124.

   [3] Jie Shen, Yingwei Wang, Jianlin Xia, Fast structured Jacobi-Jacobi trans- forms, Mathematics of Computation, 88(2019), pp. 1743-1772.

   While the authors demonstrate good scaling performance in this article, say quasilinear complexity, by numerical examples, they do not have proofs of this behavior in the general setting (only observe it).

   Basically, the algorithmic complexity of the proposed algorithm heavily depends on the (numerical) rank r in Eq.(43) or Eq.(44). It is important to have the asymptotic analysis for r as $n \to \infty$. At least a conjecture like $r = O(logn)$ or sth else should be given.

   Response: *We would like to thank the reviewers for these important references. We wanted to discuss them as in our paper for the 1D case. We have added them in the introduction and discussed them in the revision on Page 2.*

   It is difficult to rigorously prove the theoretical upper bound of r. The key to the proof is to show the existence of a smooth function $G^{(a,b)}(k, t)$ such that $P_k^{(a,b)}(t) = G(k, t)^{(a,b)} e^{ikt}$, which is of great theoretical interest but not trivial. We hope the reviewer could understand the challenge of such a theoretical proof, since there is no available asymptotic expansions as in other cases used in [1-3]. A proof of such a result, or related problem, deserves a stanalone paper as future work.

   Though we cannot provide theoretial justification in this paper, as shown by our new numerical results and those in [19] compared with Slevinsky's algorithm in [2], which is of $O(N \log^2 N/ \log(\log N))$ (here N denotes the number of grid points per dimension or the problem size in 1D), we conjecture that

   $$r = O(\frac{\log N}{\log(\log N)}). \tag{1}$$

   We have added this conjecture on Page 11.

Our algorithm in 1D is more efficient than the one in [2] as shown in [19] and probably also more efficient than the one in [3] when N is not extremely large, which is usually the case in spectral methods. Considering that the extension of the algorithms in [2] and [3] has not been studied yet, and our multidimensional algorithm is straight forward and efficient, we believe that the report of our new algorithm can benefit the community of scientific computing.

2. Besides, it seems that the fast algorithms in both [19] and this paper are based on the low-rank property of the matrices. Why the algorithm in [19] only works for the cases with $(a, b) \subset (-1/2, 1/2)$ while the one in this paper works for a lager range $(a, b) \subset (-1, 1)$? This point is a great contribution of this paper and deserves better explanation.

   Response: *The randomized SVD algorithm is a more robust and optimal algorithm for low-rank factorization. The Chebyshev based algorithm in [19] is suitable for matrices discretized from smooth functions such that Chebyshev interpolation can provide good low-rank approximation. When the matrix contains oscilation or sharp changes, though it is still the discretization of a continuous function and the matrix is low-rank, the Chebyshev interpolation might not be able to give a good accuracy. In this case, the randomized SVD algorithm still works since the matrix is still low-rank. When a and b are not in the range of $(-1/2, 1/2)$, the polynomial transformation matrix is no longer purelly oscillatory and contains some areas with sharp changes. Hence, the SVD-based algorithm in this paper is better than the Chebysheve interpolation based method in [19].*

3. **Numerical results.**

   (a) **Numerical ranks.** It is shown in the bottom left of Figure 1 that for given tolerance, the RS method provides a more compact matrix compression competitive to that of the CHEB method. However, in both of the two cases, the numerical ranks grow similarly like $r = O(log(N))$. How about other values of (a, b)? Try more numerical tests and observe the different behaviors of numerical ranks for various pairs of (a, b), even for the cases with (a, b) outside $(-1, 1)$.

   Response: *Thanks for advice. We have added two tables (Table 1 on Page 16 and Table 2 on Page 17 in Section 4) consisting of numerical ranks and other information of the comparison between the RS method and the CHEB method. All methods fail when a and b are outside the $(-1, 1)$ and hence we do not provide results in this case.*

   (b) **Numerical stability.** No discussion is presented of the stability of any of these fast transformations, which presumably may be quite poorly conditioned. It would be nice to check if the fast transforms are actually accurate numerical inverses. Something like

   $$\|Backwardtransform(Forwardtransform(v)) - v\|_2/\|v\|_2 \tag{2}$$

   for some random vector v will do.

   Response: *Thanks for your advice. In the case of "uniform" transform, the transformation matrix is orthonormal after weighting and hence the condition number is 1. Hence, the forward and inerse transforms are well-conditioned. Our method is based on an approximation to these orthonormal matrices and hence the resulting algorithms are also well conditioned. We are not aware of any ill-conditioning issue as raised by the reviewer. May the reviewer explain more on this issue if our understanding is not correct? Thanks.*

   In the case of "nonuniform" transform, we do not aim at a fast inverse, which is left for future work.

4. **Applications.** The paper is weakly motivated. A reader needs an application in mind, where the authors advocate using the proposed algorithm.

   Response: *Thanks for the suggestion. We have added a paragraph on Page 1 to motivate the applications of Jacobi polynomial transforms.*

5. **Minor issues**

   (a) The notation of points $\{x_i\}, i = 1, ..., N, \subset (-1, 1)$ appears four times in the first paragraph, which is redundant. The reviewer suggests that it should be denoted as I for short.

   Response: *Thanks. We have used X for short.*

   (b) When you introduce the Eqs.(3) and (4), it would be better to give the range for the variables t and ν clearly. For example, $t \in [0, \pi]$ and ν is a positive integer.

   Response: *Thanks. Changed.*

   (c) What are the boundary/initial conditions for the third order ODE (22)?

   Response: *Thanks. to be updated.*

   (d) The reference [22] should have a year = 2018.

   Response: *Thanks. Corrected.*

Reviewer 2: The article describes an algorithm to compute the Jacobi coefficients of a function given samples at Gauss-Jacobi points as well as the observation that 2D and 3D Jacobi transforms are Kronecker products of the univariate Jacobi transform. I have high regard for the authors and their research contributions in computational orthogonal polynomials. Unfortunately, this paper is not reflecting on their talents.

1. **NOVELTY.** I am struggling to understand the significant novelty in the manuscript. Since Section 2 recaps existing material and sections 3.2 and 3.3 are doing Kronecker products of the 1D transform, I believe the only place to find significant novelty is in Section 3.1. In Section 3.1, the authors present a variant of the algorithm from an unpublished manuscript by Bremer and Yang in 2018 for the 1D Jacobi transform with an unquantified computational improvement and a side-remark about increasing the range of the Jacobi parameter (a,b) (section 4.1.2 is not precise). It feels incremental to me because nothing concrete is stated or observed. The literature is crowded in this area with the work of Daniel Potts, the work of Jianlin Xia, the work of Nick Hale, the work of Mikael Slevinsky, the related work of Akil Narayan, the alternative approaches of Arieh Iserles, and many others. In such a crowded space, one would like to see a novelty statement that is clearly stated and demonstrated.

   Response: *Thanks. The results in Section 4.1.2 (Figure 2) does not mean a bug in our code. Figure 2 shows that our variant (denoted by "RS" in Figure 2) still works well when $a = b = 0.8$ and $a = b = 0.9$, but the original algorithm in [19](denoted by "CHEB" in Figure 2) does not(its accuracy go from 8-digits to 1-digit quickly). This numerical result is a support for that our variant can work for a larger regime. As mentioned in Section 3.1 of our manuscript, the main contribution of our variant is that we replace the original Hadamard product in [19] with a Hadamard product of a low-rank matrix and a DFT matrix and then apply a randomized SVD to the new Hadamard product to obtain a low-rank factorization with a nearly optimal rank. In 1D transform, the method in our variant results in a approximation of lower computational* *complexity. For example, to approximate a 1D forward uniform Jacobi transform by applying a low-rank factorization, the original method in [19] might have to compute r_0 NUFFTs. But after applying our variant to obtain a low-rank factorization with lower rank(compared to other decomposition, SVD usually gives a low-rank factorization with smallest rank up to a fixed accuracy), to complete the same task, only $r_1(< r_0)$ inverse DFTs have to be computed. Notes that applying a DFT is usually cheaper than applying a corresponding NUFFT. Although more time might be needed to obtain a low-rank factorization using randomized SVD in our variant in 1D case(compared to the original method in [19]), but the reduction of the facorization rank would actually result in both the factorization and the application of the multi-dimensional transform with tensor-product structure, especially 3D transform(see Figure 1 for an example). And as a observation from numerical experiments (Figure 2, Section 4.1.2), our variant works for a larger regime of parameters a and b.*

2. Is it practical? A reader needs to be convinced that this is faster than the direct algorithm for practical transform sizes. Section 4 is a little unconvincing because (1) The 2D transform in Figure 3 is only shown for $N <= 1024$ (a regime where one can do $kron(J, J)vec(X) = JXJ^T$ using the naive algorithm), and (2) On my laptop I can construct the discrete Jacobi transform matrix for N = 1024 and compute the 2D transform in a naive way in 0.09 seconds. Assuming timings are measured in milliseconds (the manuscript does not give units for time in any of the figures), Figure 3 suggests that the fast transform needs 2^15 milliseconds = 32 seconds. I am a little confused because the authors say that they have 28 processes but they don't mention how or if they are using them.

   Response: *We need to add some tests for direct summation. We need to use BLAS 3 level coding for our algorithm. Try larger N.*

   Timings in all figures are measured in seconds. We have mentioned this in figure captions.

   Although we run our codes on a workstation with 28 processors, we actually use only one processor to run a single numerical experiment. We have clarified this at the beginning of Section 4.

3. What is the theoretical complexity? Throughout the manuscript there are complexity statements that involve the numerical rank of the matrix $A_n^{(a,b)}$ in (42). How does r grow with n (the paper suggests $r = O(log(N))$, but is this observed not proved)? It would be nice if the paper made the theoretical complexity precise by bounding r. In Section 4, I think we learn that the rank of the matrix $A_n^{(a,b)}$ is about 1000 for $N = (2^6)^2 = 4096$, which is barely low-rank structure in my mind. Does that mean the constants in this algorithm are computationally prohibitive? I am confused by Figure 1, bottom-left, it seems to suggest that the rank is growing linearly.

   Response: *We conjecture that the theoretical complexity is $O(r^d n^d \log n)$, where $r = O(\frac{\log n}{\log\log n})$, n is the grid size per dimension, and d is the dimension. In [19], we have shown that our algorithm in 1D is always faster than Slevinsky's algorithm in [2]. The problem sizes tested in [19] were from 10 to 10,000,000. The theoretical scaling of Slevinsky's algorithm is $O(N\log^2 N/\log(\log N))$ where N is the problem size in 1D. Hence, it was conjectured in [19] that*

   $$r = O(\frac{\log N}{\log(\log N)}). \tag{3}$$

   We have added Table 1 to demonstrated this conjecture of r numerically. Figure 1 and 2 in the first submission have been removed.

We would like to remark that for multivariate problems, the prefactor independent of the problem size N usually grows exponentially in the dimension. For example, multivariate NUFFT, the multivariate Chebyshev polynomial transforms in Chebfun. This drawback is common in our method as well as other methods.

It is difficult to rigorously prove the theoretical upper bound of r. The key to the proof is to show the existence of a smooth function $G^{(a,b)}(k,t)$ such that $P_k^{(a,b)}(t) = G(k,t)^{(a,b)}e^{ikt}$, which is of great theoretical interest but not trivial. We hope the reviewer could understand the challenge of such a theoretical proof, since there is no available asymptotic expansions. A proof of such a result, or related problem, deserves a stanalone paper as future work.

4. What are the numerical results showing us? I find the numerical results highly confusing. Why is the N^4 trend line plotted on the left column of Figure 3?

   Response: *Since the complexity of the direct summation for 2D transforms is $O(N^4)$, where N denotes the number of grid points per dimension, we want to use the N^4 trend line in Figure 3 to show that our 2D algorithm is much faster than the direct summation.*

5. Figure 2, bottom-row, makes it look like you have a bug in your CHEB algorithm for $a = b = 0.8$ and $a = b = 0.9$ (why did it go from 8-digits to 1-digit so quickly?). Why can one not slightly modify the CHEB algorithm to make it work in the $(-1, 1)/(-1/2, 1/2)$ regime?

   Response: *There is no bug in our code. The poor performance of the CHEB algorithm is one of the main motivation for the RS algorithm adopted in this paper. In theory, there is no slight modification to make the CHEB algorithm work unless the rank parameter grows quickly. We have explained why the CHEB algorithm fails in the response to the first reviewer in the answer to Question 2. Please refer to the answer in the second page of this letter.*

   We have repaced Figure 2 with Table 1 and 2 in the revision to show more numerical results about the comprision of the RS algorithm applied in this paper and the CHEB algorithm in [19].

6. Why is everything for such small N?

   Response: *In the first submission, the range of N is 2^5 to 2^{10}, which is usually the problem size in applications because the purpose of spectral methods is to reduce the problem size instead of using spacial discretization that requires larger problem sizes. Besides, we considered 2D and 3D transforms and hence the matrix size in the transform is N^2 by N^2 or N^3 by N^3. We thought N from 2^5 to 2^{10} should be large enough. Per the request of the reviewer, we present comparision of the RS algorithm in this paper and the CHEB algorithm in [19] with larger problem sizes ranging from 2^{11} to 2^{16}.*

7. The authors say that $r = ceil(2 * log2(N))$, but then say that they plot r from (52) in Figure 3, but it isn't $ceil(2 * log2(N))$. The times also need units. Also, I am assuming that "N" in the figures of Section 4 is "n" in Section 3.

   Response: *The rank parameter $r = ceil(2 * log2(N))$ is the upper bound of the rank for each dimension of a low rank factorization. In Figure 3 in the first submission, the total rank shown is the square of the rank parameter, i.e., r^2, which is larger than r. To avoid this confusing result, in this revision, we use Table 1 to show the scaling of r in N and the numerical results agree with our conjecture.*

   We have added the units of times as second and replaced "N" with "n". Thanks for your comments.

8. What about the Jacobi-to-Chebyshev transform? There are other approaches for computing the Jacobi coefficients, which are fast and make the inverse problem easier. For example, one could first evaluate a function at Chebyshev points, compute Chebyshev coefficients (using DCT), and then convert to Jacobi coefficients. In this direction, Slevinsky has a paper on writing the Chebyshev-toJacobi transform using a sum of diagonally scaled discrete sine and cosine transforms (Section 1.1 of (Slevinsky 2017) is a good review of the literature). There's also a paper by Townsend, Webb, and Olver that shows that the Chebyshev-to-Jacobi transform has a T-dot-H structure.

   Response: *We have compared our FFT approach with the approaches by Slevinsky and Townsend et al. in the one dimensional case in [19]. Our FFT approach is more efficient than these methods based on conversion for all problem sizes. Hence, we expect that in multivariate transformations, our approach could be more attractive. Furthermore, our FFT approach is basically a few applications of FFT, which might be conceptially simpler than the T-dot-H structure, which requires some expertise to implement the T and H computation using FFT.*

9. **Minor Remark 1.** The authors mention several times that the inverse for a nonuniform Jacobi transform requires solving a highly illconditioned linear system. While strictly correct, the authors may want to be aware of the relations in Potts, Steidl, and Tasche's paper from 1998 (see page 1578). In particular, that paper shows them that WJ is the inverse of J in the uniform case, and generalizes this observation to other nodes from interpolative quadrature rules.

   Response: *Thanks for mentioning the observation in related papers. We have cited and discussed this paper on Page 2. James, do you know this WJ inversion?.*

10. **Minor Remark 2.** In Section 3.2, the authors say that the 2D nonuniform transform can be written as a Kronecker product. While this is true for the transform in (5.3), the fully nonuniform Jacobi transform involves points $(x1, y1), ..., (xN, yN)$ without tensor-product structure. Here, the authors have restricted themselves to nonuniform tensor product grids. It might be worth explicitly stating this restriction.

    Response: *Thanks. We have explicitly stated the restriction in Section 3.2 and 3.3 in our new manuscript. We are not aware of any applications for the "fully" nonuniform transforms and hence we didn't study them, the solution of which could be obtained via interpolation.*

11. **TYPOS.**Here are a few typos that I spotted while reading:

    - p.2, line -3: "referred as" should read "referred to as". Same error on p.13.

    - p.7, just before (28): "barcyentric" should read "barycentric". Same error on p.2, line -5 and p.8 just after (34).

    - p.11, line 11: "to demonstrate the superior of the new method" should read "to demonstrate the superiority of the new method"

    - p.13, just before (57): "we consider the the tensor" should read "we consider the tensor"

    - Section 4 uses "N" for the size of transforms, while the rest of the manuscript uses "n".

    - p18, conclusion: This is the first time that you tell the reader that the range of interest for (a,b) is $-1 < a, b < 1$.

    Response: *Thanks. Corrected. We have added that the meaning of "N" in Section 4 is as the same as that of "n" in the rest of the manuscript. And added in the beginning of the paper that range of interest for (a,b) is $-1 < a, b < 1$.*

Reviewer 3: Thanks for your reading, though we haven't received your elaborate comments.