Tighter Fourier Transform Complexity Tradeoffs

Nir Ailon
Department of Computer Science
Technion Israel Institute of Technology
Haifa, Israel
nailon@cs.technion.ac.il

February 12, 2015

Abstract

The Fourier Transform is one of the most important linear transformations used in science and engineering. Cooley and Tukey’s Fast Fourier Transform (FFT) from 1964 is a method for computing this transformation in time $O(n \log n)$. From a lower bound perspective, Morgenstern’s result from 1974 provides an $\Omega(n \log n)$ lower bound for the unnormalized Fourier Transform (of determinant $n^{n/2}$), assuming the linear computational model using numbers of at most constant modulus. Ailon shows in 2013 an $\Omega(n \log n)$ for computing the normalized Fourier Transform (of determinant 1) assuming only unitary operations on two coordinates are allowed at each step, and no extra memory is allowed. In 2014, Ailon then improved the result to show that, essentially, if an algorithm speeds up the FFT by a factor of $b(n) \geq 1$, then it must rely on computing, as an intermediate “bottleneck” step, a linear mapping $M$ of the input with condition number $\Omega(b(n))$.

We improve [Ailon 2014] in two ways. Our secondary result shows that the bottleneck is more severe than that presented in [Ailon 2014]. Our primary result shows that a factor $b$ speedup implies existence of a $b$-ill conditioned bottleneck occurring at $\Omega(n)$ different steps, each causing information from independent (orthogonal) components of the input to either overflow or underflow.

1 Introduction

The (discrete) normalized Fourier transform is a complex linear mapping sending an input $x \in \mathbb{C}^n$ to $y = Fx \in \mathbb{C}^n$, where $F$ is an $n \times n$ unitary matrix defined by

$$F(k, \ell) = n^{-1/2} e^{-i2\pi k\ell/n}. \quad (1.1)$$

The Fast Fourier Transform (FFT) of Cooley and Tukey [7] is a method for computing the Fourier transform of a vector $x \in \mathbb{C}^n$ in time $O(n \log n)$ using a so called linear algorithm. A linear algorithm, as defined in [12], is a sequence $\mathcal{F}_0, \mathcal{F}_1, \ldots$, where each $\mathcal{F}_i$ is a set of affine functions, for each $i \geq 0$, $\mathcal{F}_{i+1} = \mathcal{F}_i \cup \{\lambda_i f + \mu_i g\}$ for some $\lambda_i, \mu_i \in \mathbb{C}$ and $f, g \in \mathcal{F}_i$, and $\mathcal{F}_0$ contains (projections onto) the input variables as well as constants.

As for lower bounds, it is trivial that computing the Fourier Transform requires a linear number of steps. Papadimitriou, for example, derives in [13] an $\Omega(n \log n)$ lower bound for Fourier transforms in finite fields using a notion of an information flow network. It is not clear how to extend...
that result to the Complex field. There have also been attempts [15] to reduce the constants hiding in the upper bound of $O(n \log n)$, while also separately counting the number of additions versus the number of multiplications (by constants). In 1973, Morgenstern proved that if the modulus of the $\lambda_i$’s and $\mu_i$’s is bounded by 1 then the number of steps required for computing the unnormalized Fourier transform, defined by $n^{1/2}F$ in the linear algorithm model is at least $\frac{1}{2}n \log_2 n$. He used a potential function related to matrix determinant, which makes the technique inapplicable for deriving lower bounds for the (normalized) $F$. Morgenstern’s result also happens to imply that the transformation $\sqrt{n} \text{Id}$ ($\sqrt{n}$ times the identity) has the same complexity as the Fourier transform, which is not a satisfying conclusion. We also note that the unnormalized Fourier transform, as studied by Morgenstern cannot be simply implemented on a computer of fixed word length, because that transformation increases norms by $\sqrt{n}$, and hence $\Omega(\log n)$ bit words are necessary.

Ailon [1] studied the complexity of the (normalized) Fourier transform in a computational model allowing only orthogonal transformations acting on (and replacing in memory) two intermediates at each step. He showed that at least $\Omega(n \log n)$ steps were required. The proof was done by defining a potential function on the matrices $M^{(t)}$ defined by composing the first $t$ gates. The potential function is simply the sum of Shannon entropy of the probability distributions defined by the squared modulus of elements in the matrix rows. (Due to orthogonality, each row, in fact, thus defines a probability distribution.) That result had two shortcomings: (i) The algorithm was assumed not to be allowed to use extra memory in addition to the space used to hold the input. In other words, the computation was done in place. (ii) The result was sensitive to the normalization of $F$, and was not useful in deriving any lower bound for $\gamma F$ for $\gamma \not\in \{0, \pm 1\}$.

In [2], Ailon took another step forward by showing a lower bound for computing any scaling of the Fourier transform in a stronger model of computation which we call uniformly well conditioned. At each step, the algorithm can perform a nonsingular linear transformation on at most two intermediates, as long as the matrix $M^{(t)}$ defining the composition of the first $t$ steps must have condition number at most $\kappa$, for all $i$. We remind the reader that condition number is defined as $\|M^{(t)}\| \cdot \|(M^{(t)})^{-1}\|$, where $\| \cdot \|$ is spectral norm. Equivalently, it is defined as $\sigma_1(M^{(t)})/\sigma_n(M^{(t)})$, where $\sigma_1(A) \geq \cdots \geq \sigma_n(A)$ are singular values of a matrix $A$. Otherwise stated, the result implies that if an algorithm computes the Fourier transform in time $(n \log n)/b$ for some $b > 1$, then some $M^{(t)}$ must have condition number at least $\Omega(b)$. This means that the computation output relies on an ill conditioned intermediate step. It is not difficult to see from the proof of the main result of [2] that, in fact most of the algorithm steps have condition number at least $\Omega(b)$. Hence, the bottleneck is ubiquitous. (We will return to the point of bottleneck ubiquity shortly.) Ill conditionedness, as claimed in [2], leads to numerical instability and compromise in robustness against noise. In this work we make a quantification of these claims. Roughly speaking, we show that a speed up of factor $b$ implies that we either have to represent scalars with $\Omega(\log b)$ bits (implying increased computation complexity in bit operations), or we’d have to sacrifice numerical errors $\Omega(b)$ times larger than those of standard FFT, for a given fixed word size architecture.

1.1 Main Contribution

Our first result (Theorem 5.1) is that, a speedup by a factor of $b$ requires not only $b$ ill-conditionedness in the usual sense, but requires a stronger notion of ill conditionedness which we define below. The theorem generalizes the main result in [2], while also providing a simpler proof.

Our second and main result (Theorem 5.3) states that, roughly speaking, in an architecture of fixed word size, speeding up FFT by a factor of $b$ implies the existence of $\Omega(n)$ orthogonal
directions in input space that are numerically unstable in that they cause overflow or underflow at \( \Omega(n) \) different time steps. The exact definition of overflow and underflow will be defined below. Avoiding these numerical instabilities could be technically achieved by allowing word size of length \( \Omega(\log b) \) at the offending memory word locations at particular time steps, but this incurs a bit operation complexity of \( \Omega(n \log b) \) because the number of such steps is \( \Omega(n) \).

## 2 Fourier Transform Types

In theory and practice of engineering and computer science, there are two main types of Fourier transforms considered. The \( n \) dimensional Discrete Fourier Transform (DFT), as defined in (1.1), is a complex unitary mapping defined by the characters of the Abelian group \( \mathbb{Z}/n\mathbb{Z} \). The Walsh-Hadamard transform is a real orthogonal mapping defined by the characters of the \( n \) dimensional binary hypercube \( (\mathbb{Z}/2\mathbb{Z})^n \). More precisely, for \( n \) an integer power of 2, the \( (i, j) \)'th matrix element is defined as \( \frac{1}{\sqrt{n}}(-1)^{\langle i-1,j-1 \rangle} \), where \( \langle \cdot, \cdot \rangle \) is dot-product, and \( [p] \) denotes (here only) the bit representation of the integer \( p \in \{0, \ldots, n-1\} \) as a vector of \( \log n \) bits. Similarly to FFT for DFT, there is an \( O(n \log n) \) time algorithm for computing the Walsh-Hadamard transform of an input \( x \).

In this work we consider computation over the reals, and must hence view the \( n \)-DFT as an orthogonal mapping in \( \mathbb{R}^{2n} \) in a standard way. (Each complex input coordinate is viewed as two real coordinates, one representing the real part and the other the imaginary part. Each complex matrix element of DFT is, accordingly, viewed as a two-by-two real matrix.) Throughout, we will assume \( n \) is an integer power of 2 and will use \( F \) to denote either the \( n \)-Walsh-Hadamard transform or the real representation of the \( n/2 \)-DFT transform. In order for our results to work for the DFT case, however, we will have to make a slight modification to the potential function \( \Phi \) defined in [2]. (This was not an issue there). We will present the modified function \( \Phi^C \) below and briefly explain where it is necessary, but will omit the technical details of working with \( \Phi^C \), which are not much different than for \( \Phi \). For simplicity, the reader is invited to assume that \( F \) denotes the Walsh-Hadamard transform, and FFT refers to the fast Walsh-Hadamard transform.

### 2.1 Paper Structure

We start by defining the computational model in Section 3. In Section 4 we introduce further notation and recall (and generalize) the matrix quasi-entropy function which was first defined in [2]. In Section 5 we present our results. The proofs, together with useful lemmas, are deferred to Section 6. One particular Lemma 6.4, a key to proving Theorem 5.3, is interesting in its own right.

## 3 Computational Model and Notation

We remind the reader of the computational model discussed in [1, 2], which is a special case of the linear computational model. The machine state represents a vector in \( \mathbb{R}^\ell \) for some \( \ell \geq n \), where it initially equals the input \( x \in \mathbb{R}^n \) (with possible padding by zeroes, in case \( \ell > n \)). Each step (gate) is either a rotation or a constant. A rotation applies a 2-by-2 rotation mapping on a pair of machine state coordinates (rewriting the result of the mapping to the two coordinates). We remind the reader that a 2-by-2 rotation mapping is written in matrix form as \( \begin{pmatrix} \cos \theta & \sin \theta \\ -\sin \theta & \cos \theta \end{pmatrix} \) for some
real (angle) \( \theta \). A constant gate multiplies a single machine state coordinate (rewriting the result) by a nonzero constant. In case the constant equals \(-1\), we call it a reflection gate.

In case \( \ell = n \) we say that we are in the in-place model. Any nonsingular linear mapping over \( \mathbb{R}^n \) can be decomposed into a sequence of rotation and constant gates in the in-place model, and hence our model is, in a sense, universal. FFT works in the in-place model, using rotations (and possibly reflections) only. A restricted method for dealing with \( \ell > n \) was developed in [2], and can be applied here too in a certain sense (see Section 7 for a discussion). We focus in this work on the in-place model only.

Since both rotations and constants apply a linear transformation on the machine state, their composition is a linear transformation. If \( \mathcal{A}_n \) is an in-place algorithm for computing a linear mapping over \( \mathbb{R}^n \), it is convenient to write it as \( \mathcal{A}_n = (M^{(0)} = \text{Id}, M^{(1)}, \ldots, M^{(m)}) \) where \( m \) is the number of steps (gates), \( M^{(t)} \in \mathbb{R}^{n \times n} \) is the mapping that satisfies that for input \( x \in \mathbb{R}^n \) (the initial machine state), \( M^{(t)}x \) is the machine state after \( t \) steps. (Id is the identity matrix). The matrix \( M^{(m)} \) is the target transformation, which will typically be \( F \) in our setting. For any \( t \in [m] \), if the \( t \)th gate is a rotation, then \( M^{(t)} \) deflects from \( M^{(t-1)} \) in at most two rows, and if the \( t \)th gate is a constant, then \( M^{(t)} \) deflects from \( M^{(t-1)} \) in at most one row.

Both the \( n \)-Walsh-Hadamard and the real representation of the (\( n/2 \))-DFT can be implemented in-place using \( O(n \log n) \) rotations.

### 3.1 Numerical Architecture

The in-place model implicitly assumes representation of a vector in \( \mathbb{R}^n \) in memory using \( n \) words. A typical computer word represents a coordinate (with respect to some fixed orthogonal basis) in the range \([-1, 1]\) to within some accuracy \( \varepsilon \). The accuracy level \( \varepsilon \) is \( \Theta(1) \). For sake of simplicity, it should be thought of as \( 2^{-31} \) or \( 2^{-63} \) in modern computers of 32 or 64 bit words, respectively.

To explain the difficulties in speeding up FFT on computers of fixed precision words in the in-place model, we need to understand whether (and in what sense) standard FFT is at all suitable on such machines. First, we must restrict the domain of inputs. Clearly this domain cannot be \( \mathbb{R}^n \), because computer words can only represent coordinates in the range \([-1, 1]\), by our convention. We could restrict the input to come from the cube \([-1, 1]^n\). Initializing the machine with such input is easy, but the output (as well as intermediate steps) of the standard FFT map the cube to orthogonal transformations thereof, which might exceed the cube. The extreme case is an orthogonal transformation that translates the unit vector \((1/\sqrt{n}, \ldots, 1/\sqrt{n})\) to \((0, \ldots, 1)\). Under such transformation, an input of \((1, \ldots, 1)\) from the cube is mapped to \((0, \ldots, 0, \sqrt{n})\), overflowing in the last coordinate.

It makes more sense to consider input from an \( n \)-ball of radius \( \Theta(\sqrt{n}) \), which we denote \( \mathcal{B}(\Theta(\sqrt{n})) \). An \( n \)-ball is invariant to orthogonal transformations. Additionally, a coordinate of a random input from \( \mathcal{B}(\Theta(\sqrt{n})) \) is of typical magnitude \( \Theta(1) \), and hence typically does not overflow. The problem is that, using well known principles from high dimensional geometry, it is very likely for some coordinate to be of magnitude \( \Theta(\sqrt{\log n}) \). Hence, paradoxically, it is likely that some coordinate would overflow. Perhaps the reason that this issue is hardly every considered in literature on Fourier transform complexity (including the author’s prior work) is because a number of magnitude \( \Theta(\sqrt{\log n}) \) can be accommodated using \( \Theta(\log \log n) \) bit words. It is unlikely to encounter an application with \( n = \Omega \left( 2^{32} \right) \) (let alone \( \Omega \left( 2^{64} \right) \)), and hence this is really a non-issue for applications. Nevertheless, it’s an issue for complexity theoretical purposes. So is the true
running time of standard FFT

\[ O(n \log n) \times (\text{time to multiply two } \Theta(\log \log n) \text{ bit words}) , \]

owing to a tiny minority of coordinates that overflow? The answer is no, because on expectation the number of bits required to encode a coordinate \( x(i) \) of a random vector \( x \) from \( B(\Theta(\sqrt{n})) \) to within accuracy \( \varepsilon = \Theta(1) \), is \( \Theta(1) \)\(^1\) In fact, using state-of-the-art integer multiplication technology \([8, 9]\), the expected time required to multiply such a random coordinate with a number of \( \Theta(1) \) bits is \( \Theta(1) \) when taking logical bit operations into account. However, we prefer to avoid dealing with bit operations in this work, and leave it to future work discussion (see Section 7). Instead, we shall take an information theoretical approach, and think of a word in memory, at a particular time step of the algorithm, as a channel that is used for encoding a random coordinate. If the coordinate is drawn from a distribution that requires \( O(1) \) bits on expectation to encode to within the fixed accuracy of \( \varepsilon \), then no overflow occurs. If \( \omega(1) \) bits are necessary, then we define this as overflow. \(^2\)

By this definition, standard FFT for input drawn uniformly from \( B(\Theta(\sqrt{n})) \) does not overflow at all, because any coordinate of the machine state at any step is tightly concentrated (in absolute values) around \( \Theta(1) \).

It will be easier to replace the uniform distribution from the ball with the multivariate Gaussian \( \mathcal{N}(0, \text{Id}(\Theta(n))) \), which is a good approximation of the former for large \( n \). With this assumption, any coordinate of the standard FFT machine state at any step follows the law \( \mathcal{N}(0, \Theta(1)) \). By simple integration against the Gaussian measure, one can verify that the expected number of bits required to encode such a random variable (to within fixed accuracy \( \varepsilon \)) is \( \Theta(1) \), hence no overflow occurs. This input assumption together with the no-overflow guarantee will serve as our benchmark.

Two notes regarding the numerical architecture:

1. Our definition of overflow is counterintuitive, because we are used to thinking about overflow as an offending machine state at a particular step of the algorithm execution for a particular input, while our definition is stochastic. The reason we use this definition is from the combination of (a) our desire to work with a spherically symmetric input and (b) avoiding measuring complexity in the granularity of logical bit operations, stemming from the varying word length typically arising even in the standard FFT benchmark. \(^3\)

2. We are completely ignoring the fact that the machine state after \( t \) steps on input \( x \) in any computer is not only a result of quantizing the vector \( M^{(t)}x \). Rather, errors are accumulated from the effects of quantizing at earlier steps. Taking accumulated errors into account should affect the numerical accuracy of both the benchmark and of any speed-up, but quantifying this effect seems extremely difficult. Again, we take an information theoretical approach by saying that the machine state after \( t \) steps contains at most the information in the quantization of

---

\(^1\)By “encoding” here we simply mean the base-2 representation of the integer \( \lfloor x(i)/\varepsilon \rfloor \).

\(^2\)By encoding of a random number \( X \) here we simply mean the integer representation of its quantization \( \lfloor X/\varepsilon \rfloor \).

\(^3\)It is possible to somewhat practically justify the stochastic definition of overflow by thinking of running FFT on a large number \( L \) of iid inputs, which can be thought of as being stacked as columns of an input matrix \( X \). The larger \( L \) is, the more concentrated the total number of bits required to encode each row of the matrix around \( \Theta(L) \) will be. For \( L \) polynomial in \( n \), the probability of requiring more than \( \Theta(L) \) bits per row becomes exponentially small at any step of standard FFT. We could have defined this as our benchmark, but we do not think that anyone would want to compute over a tightly compressed representation of matrix rows into arrays of words. Hence we’ve decided to settle for the stochastic definition of overflow.
the coordinates of $M^{(t)}x$, and whatever information lost (due to this quantization) cannot be later recovered because the machine state encodes everything that is known about the input (and output) at any given step.

Our chosen approach (and compromises) enable us to obtain what we believe are important insights into inherent tradeoffs of information version computation speed.

4 The Matrix Quasi-Entropy Function

The set $\{1, \ldots, q\}$ is denoted by $[q]$. By $\mathbb{R}^{a \times b}$ we formally denote matrices of $a$ rows and $b$ columns. Matrix transpose is denoted by $(\cdot)^T$. We use $(\cdot)^{-T}$ as shorthand for $((\cdot)^{-1})^T = ((\cdot)^T)^{-1}$. If $A \in \mathbb{R}^{a \times b}$ is a matrix and $I$ is a subset of $[b]$, then (borrowing from Matlab syntax) $A(:, I)$ is the submatrix obtained by stacking the columns corresponding to the indices in $I$ side by side and $A(I, :)$ is the submatrix obtained by stacking the rows corresponding to the indices in $I$ one on top of the other. We shall also write, for $i \in [b]$, $A(:, i)$ and $A(i, :)$ as shorthands for $A(:, \{i\})$ and $A(\{i\}, :)$, respectively. All logarithms are taken in base 2.

We slightly abuse notation and extend the definition of the quasi-entropy function $\Phi(M)$ defined on nonsingular matrices $M$ from [2], as follows. Given two matrix arguments $A, B \in \mathbb{R}^{a \times b}$ for some $a, b \geq 1$, $\Phi(A, B)$ is defined as

$$
\sum_{i=1}^{a} \sum_{j=1}^{b} -A(i, j)B(i, j) \log |A(i, j)B(i, j)| .
$$

This extends naturally to vectors, namely for $u, v \in \mathbb{R}^{a}$, $\Phi(u, v)$ is as above by viewing $\mathbb{R}^{a}$ as $\mathbb{R}^{a \times 1}$. If $A, B \in \mathbb{R}^{a \times b}$ and $a, b$ are even, then we define the complex quasi-entropy function $\Phi^C(A, B)$ to be:

$$
\sum_{i=1}^{a} \sum_{j=1}^{b/2} -(A(i, 2j - 1)B(i, 2j - 1) + A(i, 2j)B(i, 2j)) \log |A(i, 2j - 1)B(i, 2j - 1) + A(i, 2j)B(i, 2j)| .
$$

The function $\Phi^C$ can be used for proving our results for the real representation of the complex DFT, which we omit from this manuscript for simplicity. The reason we need this modification to $\Phi$ for DFT is explained in the proof of Lemma 6.4, needed by Theorem 5.3 below. Elsewhere, we will work (for convenience and brevity) only with $\Phi$. Abusing notation, and following [2], we define for any nonsingular matrix $M$:

$$
\Phi(M) := \Phi(M, M^{-T}) , \Phi^C(M) := \Phi^C(M, M^{-T}) .
$$

It is easy to see that $\Phi(F) = n \log n$ for the Walsh-Hadamard transform, because all matrix elements are $\pm 1/\sqrt{n}$. If $F$ is a real representation of the $(n/2)$-DFT, then it is easy to see that $\Phi^C(F) = n \log (n/2)$, because all matrix elements of the (complex representation of the) $(n/2)$-DFT are complex unit roots divided by $\sqrt{n/2}$.

It will be also useful to consider a generalization of the potential of a nonsingular matrix $M$, by allowing linear operators acting on the rows of $M$ and $M^{-T}$, respectively. More precisely, we will let $\Phi_{P, Q}(M)$ be shorthand for

$$
\Phi(MP, M^{-T}Q) ,
$$
where \( P, Q \in \mathbb{R}^{n \times n} \) are some mappings. (We will only be working with projection matrices \( P, Q \) here). Similarly, \( \Phi_{P,Q}^c(M, M^{-T}) := \Phi^c(MP, M^{-T}Q) \).

Finally, for any matrix \( A \in \mathbb{R}^{n \times n} \), let \( \sigma_1(A), \ldots, \sigma_n(A) \) denote its singular values, where we use the convention \( \sigma_1(A) \geq \cdots \geq \sigma_n(A) \). If \( A \) is nonsingular, then the condition number \( \kappa(A) \) is defined by \( \sigma_1(A)/\sigma_n(A) \). For any matrix \( A \), we let \( \|A\| \) denote its spectral norm and \( \|A\|_F \) its Frobenius norm. If \( x \) is a vector, hence, \( \|x\| = \|x\|_2 = \|x\|_F \). Let \( B \) denote the Euclidean unit ball in \( \mathbb{R}^n \).

## 5 Results

### 5.1 FFT Speedup \( \Rightarrow \) Bottleneck

The result for this section should interpreted as follows: If an in-place algorithm \( \mathcal{A}_n = (M^{(0)} = \text{Id}, \ldots, M^{(m)} = F) \) speeds up FFT by a factor of \( b \geq 1 \), then there exists an intermediate step \( M(t) \) that is ill conditioned (in a generalized sense, to be explained).

**Theorem 5.1.** Fix \( n \), and let \( \mathcal{A}_n = \{\text{Id} = M^{(0)}, \ldots, M^{(m)} = F\} \) be an in-place algorithm computing some linear function in \( \mathbb{R}^n \) and let \( P, Q \in \mathbb{R}^{n \times n} \) be two matrices. For any \( t \in [m] \), let \( \{i_t, j_t\} \) denote the set of at most two indices that are affected by the \( t \)th gate (if the \( t \)th gate is a constant gate, then \( i_t = j_t \), otherwise it’s a rotation acting on indices \( i_t, j_t \)). Then for any \( R \in [(n/2)] \) there exists \( t \in [m] \) such that

\[
\sqrt{\|(M^{(t)})^T(P)(I_t, :)\|_F^2} \|(M^{(t)})^{-T}Q(I_t, :)\|_F^2 \geq \frac{R(\Phi_{P,Q}^c(M^{(m)} - \Phi_{P,Q}(\text{Id}))}{m \log 2R},
\]

where \( I_t = \bigcup_{t=1}^{R-1} \{i_t, j_t\} \). Additionally, if \( R = 1 \) then the \( t \)th gate can be assumed to be a rotation.

In particular, if \( M^{(m)} = F \) and \( m = (n \log n)/b \) for some \( b \geq 1 \) (“\( \mathcal{A}_n \) speeds up FFT by a factor of \( b \)”)) and \( P = Q = \text{Id} \), then

\[
\sqrt{\|(M^{(t)})^T(I_t, :)\|_F^2} \|(M^{(t)})^{-T}(I_t, :)\|_F^2 \geq \frac{Rb}{\log 2R}.
\]

For the main result in this paper in the next section, we will only need the case \( R = 1 \) of the theorem. It is worthwhile, however, to state the following corollary for general \( R \), which is a simple consequence of the fact that for any matrix \( X \) and integer \( q \), \( \sum_{i=1}^q \sigma_i^2(X) \geq \|X(I, :)\|_F^2 \) whenever \( |I| \leq q \).

**Corollary 5.2.** If \( M^{(m)} = F \) and \( m = (n \log n)/b \) then

\[
\sqrt{\left( \sum_{i=1}^{2R} \sigma_i^2(M^{(t)}) \right)} \left( \sum_{i=1}^{2R} \sigma_{n+1-i}^2(M^{(t)}) \right) \geq \frac{Rb}{\log 2R}.
\]

This repeats the main result in [2], for the case \( R = 1 \) (our proof is also simper). The bound (5.3) is stronger in the sense that it imposes a stronger restriction on the spectrum of \( M^{(t)} \) for larger values of \( R \). The interesting case is, say, \( R = 2^{b/2} \). With this choice, (5.3) becomes:

\[
\sqrt{\left( \sum_{i=1}^{2^{b/2+1}} \sigma_i^2(M^{(t)}) \right)} \left( \sum_{i=1}^{2^{b/2+1}} \sigma_{n+1-i}^2(M^{(t)}) \right) \geq 1.9 \times 2^{b/2}.
\]

7
One extreme case of (5.4) is when \( \sigma_1(M^{(t)}) = \Omega(2^{b/2}) \) and \( \sigma_2(M^{(t)}) = \cdots = \sigma_n(M^{(t)}) = 1 \). In this case, the condition number is exponentially higher than the bound achieved in [2]. The other extreme is the case \( \sigma_1(M^{(t)}) = \cdots = \sigma_{2^{b/2}+1}(M^{(t)}) = \alpha \) and \( \sigma_{n-2^{b/2}+2}(M^{(t)}) = \cdots = \sigma_n(M^{(t)}) = 1 \), for some global \( \alpha > 1 \). In other words, \( \alpha \)-ill-conditionedness is exhibited by \( 2^{b/2} \) disjoint pairs of eigenvalues of \( M^{(t)} \). We do not continue the discussion of this extended notion of ill condition, because we only need the \( R = 1 \) case for our main result in what follows.

### 5.2 Implications of Theorem 5.1 on Numerical Accuracy

We discuss the implication of Theorem 5.1 in case \( R = 1, P = Q = \mathrm{Id} \). The theorem implies that an algorithm with \( m = (n \log n)/b \) must exhibit an intermediate matrix \( M^{(t)} \) and a pair of indices \( i_t, j_t \) such that the \( t \)'th gate is a rotation acting on \( i_t, j_t \) and additionally:

\[
\sqrt{(\|M^{(t)}(i_t, :)\|^2 + \|M^{(t)}(j_t, :)\|^2)} \sqrt{(\|M^{(t)} - T(i_t, :)\|^2 + \|M^{(t)} - T(j_t, :)\|^2)} \geq \sqrt{b} .
\]

Hence, either

(i) \( \sqrt{\|M^{(t)}(i_t, :)\|^2 + \|M^{(t)}(j_t, :)\|^2} \geq \sqrt{b} \) or

(ii) \( \sqrt{\|M^{(t)} - T(i_t, :)\|^2 + \|M^{(t)} - T(j_t, :)\|^2} \geq \sqrt{b} \).

**Case (i).** We can assume wlog that

\[
\|M^{(t)}(i_t, :)\|^2 \geq b/2 .
\]

Then, letting

\[
x_{\text{over}}^T = M^{(t)}(i_t, :) / \|M^{(t)}(i_t, :)\| \in \mathbb{R}^n
\]

(\( x_{\text{over}} \) is the normalized \( i_t \)'th row of \( M^{(t)} \), transposed), we get that the \( i_t \)'th coordinate of \( M^{(t)} x_{\text{over}} \) is at least \( \sqrt{b} \). But this implies that, for input \( x = x_{\text{over}} \), the \( t \)'th step acts on a pair of numbers \( (M^{(t)} x_{\text{over}})(i_t), (M^{(t)} x_{\text{over}})(j_t) \), at least one of which requires representation using \( O(\log b) \) in the numerical architecture described in Section 3. Now, for input \( x \sim \mathcal{N}(0, \Theta(n)) \), the \( i_t \)'th coordinate just before the application of the \( t \)'th gate equals \( \|M^{(t)}(i_t, :)\| x^T x_{\text{over}} \), and is hence distributed

\[
\mathcal{N}(0, \Theta(\|M^{(t)}(i_t, :)\|^2)) .
\]

Using (5.5), this distribution is \( \mathcal{N}(0, \Omega(b)) \). If \( b = \omega(1) \), then by our definition we reach overflow.

Note also that it is possible as a preprocessing step to replace \( x \) with \( x - (x^T x_{\text{over}}) x_{\text{over}} \) (eliminating the overflow component), and then to reintroduce the offending component by adding \( (x^T x_{\text{over}}) F x_{\text{over}} \) as a postprocessing step. In the next section, however, we shall show that, in fact, there must be \( \Omega(n) \) pairwise orthonormal directions (in input space) that cause such an overflow at \( \Omega(n) \) different time steps, so such a simple “hack” cannot not work.

**Case (ii).** This scenario, as the reader guesses, should be called *underflow*. In case (ii), wlog

\[
\|M^{(t)} - T(i_t, :)\|^2 \geq b/2 .
\]

Now define \( x_{\text{under}}^T = (M^{(t)} - T(i_t, :) / \|M^{(t)} - T(i_t, :)\| \in \mathbb{R}^n \), and consider the orthonormal basis \( u_1, \ldots, u_n \in \mathbb{R}^n \) so that \( u_1 = x_{\text{under}} \). For any \( t' \in [m] \) (and in particular for \( t' = t \)):

\[
g_1 := x_{\text{under}}^T x = (x_{\text{under}}^T (M^{(t')} - T)) . (M^{(t')} x) .
\]
Now notice that the $i_{th}$ coordinate of $(x_{\text{over}}^T(M^{(t)})^{-1})$ has magnitude at least $\sqrt{b/2}$ by (5.6) and the construction of $x_{\text{over}}$. Also notice that for all $i \neq i_t$, the row $M^{(t)}(i,:)$ is orthogonal to $x_{\text{over}}$, by matrix inverse definition. This means that coordinate $i \neq i_t$ of $M^{(t)}x$ contains no information about $g_1$. All the information in $g_1$ is hence contained in $(M^{(t)}x)(i_t)$. More precisely, $g_1$ is given by

$$g_1 = ((M^{(t)})^{-T}x_{\text{under}}(i_t) \times (M^{(t)}x)(i_t)) - e,$$

where $e$ is a random variable independent of $g_1$. But $|(M^{(t)})^{-T}x_{\text{under}}(i_t)| \geq \sqrt{b/2}$, and $(M^{(t)}x)(i_t)$ is known only up to an additive error of $\varepsilon$, due to our assumptions on quantization in the numerical architecture. This means that $g_1$ can only be known up to an additive error of at least $\varepsilon \sqrt{b/2}$, for any value of $e$. It is important to note that this uncertainty cannot be “recovered” later by the algorithm, because at any step the machine state contains all the information about the input (aside from the input distribution prior). In other words, any information forgotten at any step cannot be later recalled.

Figure 1: The uncertainty interval in $g_1$ as a function of $e$, given the representation of $M^{(t)}x(i_t)$ in a computer word. The random variable $e$ contains only information from components of the input that are orthogonal to $x_{\text{under}}$.

The uncertainty interval of size $\geq \varepsilon \sqrt{b/2}$ is not very severe, in the sense that it affects just one out of $n$ input components. Indeed, at step 0 the input vector coordinates $x(1), \ldots, x(n)$ are represented in individual words, each of which gives rise to an uncertainty interval of width $\varepsilon$. This means that merely storing the input in memory in the standard coordinate system implies knowing its location up to an uncertainty $n$-cube with side $\varepsilon$, and of diameter $\varepsilon \sqrt{n}$. An uncertainty interval of size $\varepsilon \sqrt{b/2} = O(\varepsilon \sqrt{\log n})$ in a single direction is therefore hardly a deterrent for someone who wants to asymptotically beat FFT. The next section tells us, however, that the situation is much more severe, in the sense that underflow occurs in $\Omega(n)$ pairwise orthogonal input directions.

5.3 Ill Conditioned Botlenecks in $\Omega(n)$ Orthogonal Directions

Theorem 5.3. Fix $n$, and let $A_n = \{ \text{Id} = M^{(0)}, \ldots, M^{(m)} = F \}$ be an in-place algorithm computing $F$ in time $m = (n \log n)/b$ for some $b \geq 1$. Then one of the following (i)-(ii) must hold:

(i) (Severe Overflow) There exists an orthonormal system $v_1, \ldots, v_{n'} \in \mathbb{R}^n$, integers $t_1, \ldots, t_{n'} \in [m]$ and $i_1, \ldots, i_{n'} \in [n]$ with $n' = \Omega(n)$ such that for all $j \in [n']$,

$$M^{(t_j)}(i_j,:)P_j = \alpha_j v_j \quad \text{with} \quad \alpha_j = \Omega(\sqrt{b}), \quad (5.7)$$

4To be precise, we must acknowledge the prior distribution on $x$ which also provides information about its whereabouts.
where $P_j$ is projection onto the space orthogonal to $v_1, \ldots, v_{j-1}$.

(ii) (Severe Underflow) There exists an orthonormal system $u_1, \ldots, u_{n'} \in \mathbb{R}^n$, integers $t_1, \ldots, t_{n'} \in [m]$ and $i_1, \ldots, i_{n'} \in [n]$ with $n' = \Omega(n)$ such that for all $j \in [n']$,

$$(M^{(t_j)})^{-T}(i_j,:)Q_j = \gamma_j u_j \quad \text{with } \gamma_j = \Omega(\sqrt{b}) \ ,$$

(5.8)

where $Q_j$ is projection onto the space orthogonal to $u_1, \ldots, u_{j-1}$.

In both cases of severe overflow and severe underflow, the gates at time $t_1, \ldots, t_{n'}$ are all rotations, for all $j \in [n']$ the index $i_j$ is one of the two indices affected by the corresponding rotation. Additionally, the set $\{t_1, \ldots, t_{n'}\}$ is of size at least $n'/2$.

The proof is deferred to Section 6.2. We discuss its implications, continuing the discussion following Theorem 5.1 in Section 5.2.

In the severe overflow case, Theorem 5.3 tells us that there exists an orthonormal collection $v_1, \ldots, v_{n'}$ (with $n' = \Omega(n)$) in the input space, such that each $v_i$ behaves like $x_{\text{overflow}}$ (from Section 5.2). This means that, if the speedup factor $b$ is $\omega(1)$, we have overflow caused by a linear number of independent in put components, occurring at $\Omega(n)$ different time steps (by the last sentence in the theorem). In the extreme case of speedup $b = \Theta(\log n)$ (linear number of gates), this means that in a constant fraction of time steps overflow occurs.

The severe underflow is more interesting to analyze. An intuitive analysis is as follows: For each orthogonal direction $v_j$, for $j \in [n']$, we lose $\Theta(\log b)$ bits of information about $x^T v_j$ just before applying the $t_j$'th gate because, roughly speaking, we are trying to squeeze the value $x^T v_j/\gamma_j \sim N(0, \Theta(1/\gamma_j^2))$ in the $i_j$'th word while not storing any information about $x^T v_j$ in any other word, and $\gamma_j \geq \sqrt{b/2}$. This lost information cannot be recovered later.

Although this explanation can be made rigorous using information theoretical tools, we prefer to offer a geometric interpretation. The theorem tells us that there exists an orthonormal collection $u_1, \ldots, u_{n'}$ in the input space that is bad in the following sense. For each $j \in [n']$, redefine $g_j = u_j^T x$ to be the input component in direction $u_j$. Again, the variables $g_1, \ldots, g_{n'}$ are iid $N(0, \Theta(1))$. The first element in the series, $u_1$, can be analyzed as $x_{\text{under}}$ (Section 5.2), whereby it was argued that before the $t_1$'th step, the component $g_1 = u_1^T x$ can only be known to within an interval of width $\Omega(\gamma_1 \varepsilon)$, independently of information from components orthogonal to $u_1$. We remind the reader that by this we mean that the width of the interval is independent, but the location of the interval depends smoothly (in fact, linearly) on information from orthogonal components of $x$ (see Figure 1).

As for $u_2, \ldots, u_{n'}$: For each $j \in [n']$, let $z_j := (M^{(t_j)})^{-T}(i_j,:)$. Therefore $u_1 = z_1/\|z_1\|$ and by (5.8), for $j > 1$ we can write

$$z_j = \gamma_j u_j + h_j$$

where $h_j \in \text{span}\{u_1, \ldots, u_{j-1}\}$. Treating $z_j/\|z_j\|$ as $x_{\text{under}}$ in Section 5.2 we conclude that the component $(z_j/\|z_j\|)^T x$ of $x$ can only be known to within an interval of size $\Omega(\varepsilon z_j)$, given any value of the projection of input $x$ onto the space orthogonal to $z$.

We extend the list of vectors $z_1, \ldots, z_{n'}$, orthonormal vectors $u_1, \ldots, u_{n'}$, numbers $\gamma_1, \ldots, \gamma_{n'}$ and projections $Q_1, \ldots, Q_{n'}$ to size $n$ as follows. Having defined $z_j, u_j, Q_j, \gamma_j$ for some $j \geq n'$, we inductively define $Q_{j+1}$ as projection onto the space orthogonal to span$\{z_1, \ldots, z_j\} = \text{span}\{u_1, \ldots, u_j\}$ and $z_{j+1}$ to be a standard basis vector such that $\|Q_{j+1} z_{j+1}\|^2 \geq 1 - j/n$. (Such a vector exists because there must exist an index $i_0 \in [n]$ such that $\sum_{j'=1}^{j} u_j(i_0)^2 \leq j/n$, by orthonormality of the collection $u_1, \ldots, u_j$; Now set $z_{j+1}$ to have a unique 1 at coordinate $i_0$ and 0 at all other

10
coordinates.) We let \( u_{j+1} \) be \( Q_{j+1}z_{j+1}/\|Q_{j+1}z_{j+1}\| \), that is, a normalized vector pointing to the component of \( z_{j+1} \) that is orthogonal to \( \text{span}\{z_1, \ldots, z_j\} = \text{span}\{u_1, \ldots, u_j\} \). The number \( \gamma_{j+1} \) is defined as \( \|Q_{j+1}z_{j+1}\| \). By construction, \( \gamma_{j+1} \geq \sqrt{1 - j/n} \).

The above extends the partial construction arising from the severe underflow to a full basis, with the following property:

**Proposition 5.4.** For any \( j \in \mathbb{N} \), even given exact knowledge of the exact projection \( \bar{x} \) of \( x \) onto the space orthogonal to \( z_j \), the quantity \( x^T(z_j/\|z_j\|) \) upon termination of the algorithm can only be known to within an interval of the form \([s, s + \varepsilon\|z_j\|]\) where \( s \) depends smoothly (in fact, linearly) on \( \bar{x} \).

The proposition is simply a repetition of the analysis done for \( x_{\text{under}} \) in Section 5.2 for \( j \leq n' \). For \( j > n' \) it is a simple consequence of the fact that upon initialization of the algorithm with input \( x \), each coordinate of \( x \) (and in particular \( x^Tz_j \)) is stored in a single machine word, while all other machine words store information independent of \( x^Tz_j \). Hence the uncertainty of width \( \varepsilon\|z_j\| = \varepsilon \).

What do we know about \( x \) upon termination of the algorithm? As stated earlier, any information that was lost during execution, cannot be later recovered. Let \( \mathcal{I} \) denote the set of possible inputs, given the information the we are left with upon termination. Consider the projection \( Q_2 \) onto the space orthogonal to \( u_1 = z_1/\|z_1\| \), as a function defined over \( \mathcal{I} \). Let \( \mathcal{I}_2 = Q_2 \mathcal{I} \) denote its image. The preimage of any point \( w \in \mathcal{I}_2 \) must contain a line segment of length at least \( \varepsilon \gamma_1 \) parallel to \( u_1 \), due to the uncertainty in \( x^Tu_1 \). Hence the volume of \( \mathcal{I} \) is at least \( \varepsilon \gamma_1 \) times the \((n-1)\)-volume of \( \mathcal{I}_2 \). Continuing inductively, we lower bound the \((n-j)\)-volume of \( \mathcal{I}_j := Q_j \mathcal{I} = Q_j \mathcal{I}_{j-1} \) for \( j > 2 \). Consider the projection \( Q_j \) as a function operating on \( \mathcal{I}_{j-1} \), and any point \( w \) in the image \( \mathcal{I}_j \). By definition of \( Q_j \), there exists \( \hat{w} \in \mathcal{I} \) such that \( Q_j \hat{w} = w \). By proposition 5.3, the intersection of the line \( \mathcal{L} = \{\hat{w} + \eta z_j : \eta \in \mathbb{R}\} \) with \( \mathcal{I} \) must contain a segment \( \Delta \) of size \( \varepsilon\|z_j\| \). The projection \( Q_j \Delta \) of this segment is contained in the line \( Q_j \mathcal{L} = \{w + \eta u_j : \eta \in \mathbb{R}\} \). The size of the segment is \( \varepsilon\|Q_j z_j\| = \varepsilon \gamma_j \). This means that the \((n-j)\)-volume of \( \mathcal{I}_{j+1} \) is at least \( \varepsilon \gamma_j \) times the \((n-j)\)-volume of \( \mathcal{I}_{j+1} = Q_{j+1} \mathcal{I}_j \).

Concluding, we get that the volume of \( \mathcal{I} \) is at least \( \prod_{j=1}^n \gamma_j \). From the construction immediately preceding Proposition 5.4 we get (using the fact that \( n' = \Omega(n) \)):

\[
\log \frac{\text{vol}(\mathcal{I})}{\varepsilon^n} \geq n' \log \sqrt{b/2} + \sum_{j=n'+1}^n \log \sqrt{1 - \frac{j-1}{n}} = \Omega(n \log b) . \tag{5.9}
\]

The bound (5.9) tells us that the volume of uncertainty in the input (and hence, the output) of a \( b \)-speedup of FFT in the in-place model is at least \( b^{\Omega(n)} \) times the volume of uncertainty incurred simply by storing the input in memory.

### 6 Proofs

We start with some useful lemmas.

**Lemma 6.1.** Let \( x, y \in \mathbb{R}^a \) for some integer \( a \), with \( \|x\|_2 = \|y\|_2 = 1 \). Then

\[
- \log a \leq \Phi(x, y) \leq \log a . \tag{6.1}
\]

---

5 We need to be precise about measurability, but this is a simple technical point from the fact that the interval endpoint depends smoothly on the projection, as claimed in Proposition 5.4.
The proof is a simple done by a simple analysis of the function $\Phi(x, y)$ under the stated constraints using, say, Lagrange multipliers.

**Lemma 6.2.** Let $A, B \in \mathbb{R}^{a \times n}$. Let $U \in \mathbb{R}^{a \times a}$ be orthogonal. Then

$$|\Phi(A, B) - \Phi(UA, UB)| \leq \|A\|_F \|B\|_F \log a .$$  

(6.2)

**Proof.** Let $r_i = \|A(:, i)\|_2, s_i = \|B(:, i)\|_2$. Then

$$\Phi(A, B) = \sum_{i=1}^n \Phi(A(:, i), B(:, i)) ,$$

$$\Phi(UA, UB) = \sum_{i=1}^n \Phi(UA(:, i), UB(:, i)) ,$$

and by the triangle inequality:

$$|\Phi(A, B) - \Phi(UA, UB)| \leq \sum_{i=1}^n |\Phi(A(:, i), B(:, i)) - \Phi(UA(:, i), UB(:, i))| .$$  

(6.3)

Fix $i \in [n]$ and let $x, y \in \mathbb{R}^a$ denote $\frac{A(:, i)}{r_i}, \frac{B(:, i)}{s_i}$, respectively (note that $\|x\|_2 = \|y\|_2 = 1$).

$$\Phi(A(:, i), B(:, i)) - \Phi(UA(:, i), UB(:, i))
= - \sum_{j=1}^a rs \cdot x(j) y(j) \log |rs \cdot x(j) y(j)| + \sum_{j=1}^a rs \cdot (Ux)(j)(Uy)(j) \log |rs \cdot (Ux)(j)(Uy)(j)|$$

$$= (rs \log(rs)) \left( - \sum_{j=1}^a x(j) y(j) + \sum_{j=1}^a (Ux)(j)(Uy)(j) \right) - rs \sum_{j=1}^a x(j) y(j) \log |x(j) y(j)|$$

$$+ rs \sum_{j=1}^a (Ux)(j)(Uy)(j) \log |(Ux)(j)(Uy)(j)| .$$

By orthogonality of $U$, we have that $\sum_{j=1}^a x(j) y(j) = \sum_{j=1}^a (Ux)(j)(Uy)(j)$. Also for the same reason we have $\|Ux\|_2 = \|Uy\|_2 = 1$. Using Lemma 6.1 we conclude

$$|\Phi(A(:, i), B(:, i)) - \Phi(UA(:, i), UB(:, i))| \leq 2rs \cdot \log a .$$  

(6.5)

Summing up over $i$ and applying Cauchy-Schwarz we conclude the result. \hfill \Box

**Lemma 6.3.** Let $A, B \in \mathbb{R}^{a \times n}$, and let $D \in \mathbb{R}^{a \times a}$ be some nonsingular matrix. Then

$$|\Phi(A, B) - \Phi(DA, D^{-T}B)| \leq (\|A\|_F \|B\|_F + \|DA\|_F \|D^{-T}B\|_F) \log a$$  

(6.6)

**Proof.** Let $U, V \in \mathbb{R}^a$ be orthogonal and $\Sigma \in \mathbb{R}^{a}$ diagonal (and nonsingular) so that $D = U\Sigma V$. (Such a composition exists by standard SVD theory.)

$$|\Phi(A, B) - \Phi(DA, D^{-T}B)| \leq |\Phi(A, B) - \Phi(VA, VB)|$$

$$+ |\Phi(VA, VB) - \Phi(\Sigma VA, \Sigma^{-1}VB)|$$

$$+ |\Phi(\Sigma VA, \Sigma^{-1}VB) - \Phi(U\Sigma VA, U\Sigma^{-1}VB)|$$

$$\leq \|A\|_F \|B\|_F \log a + \|\Sigma VA\|_F \|\Sigma^{-1}VB\|_F \log a$$

$$= (\|A\|_F \|B\|_F + \|DA\|_F \|D^{-T}B\|_F) \log a ,$$  

(6.7)
as required. (We used Lemma 6.2 twice in the second inequality, and the orthogonality of $U$ for the last derivation. The reason the middle term in the RHS of the first inequality is null is by properties of $\Phi$ that are trivial to check.)

The following is the most important technical lemma in this work. Roughly speaking, it tells us that application of operators that are close to $\Id$ to the rows of $F$ and $F^{-T}$ does not increase the corresponding potential by much. Similarly, assuming that $P, Q$ are PSD with spectral norm at most 1, applying these transformations to the rows of $\Id$ does not increase the corresponding potential by much.

**Lemma 6.4.** Let $P, Q \in \mathbb{R}^{n \times n}$ be two matrices. Let $\hat{P} = \Id - P, \hat{Q} = \Id - Q$. Then

$$\Phi(FP, F^{-T}Q) = \Phi_{P, Q}(F) \geq \left( n \log n - (\text{tr } \hat{P} + \text{tr } \hat{Q}) \log n - O\left( (\|\hat{P}\|_F^2 + \|\hat{Q}\|_F^2) \log n \right) \right).$$

If, additionally, $P$ and $Q$ are positive semi-definite contractions (the spectral norm of $P, Q$ is at most 1), then

$$\Phi(\Id P, \Id Q) = \Phi_{P, Q}(\Id) = \Phi(P, Q) \leq \text{tr } \hat{P} + \text{tr } \hat{Q} + O\left( (\|\hat{P}\|_F^2 + \|\hat{Q}\|_F^2) \log n \right).$$

The proof is deferred to Section 6.3. The bounds (6.8)-(6.9) use the smoothness of the matrices $F$ and $\Id$ (that is, almost all matrix elements have exactly the same magnitude). This is the reason we needed to modify $\Phi$ and work with $\Phi^C$ for the complex case: If $F$ were the real representation of the $n/2$-DFT matrix, then it is not smooth in this sense. It does hold though that for any $i \in [n]$ and $j \in [n/2]$: $F(i, 2j - 1)^2 + F(i, 2j)^2 = 2/n$, in other words, the matrix is smooth only in the sense that all pairs of adjacent elements have the same norm (viewed as vectors in $\mathbb{R}^2$).

### 6.1 Proof of Theorem 5.1

We directly prove the less general (5.3). The more general bound (5.1) is shown similarly, but with more notation. Fix $R \in [(n/2)]$. Let $m'$ be the smallest integer divisible by $R$ satisfying $m' \geq m$. If $m' > m$, then “pad” the algorithm $\mathcal{A}_n$ by defining $M^{(m+1)} \ldots M^{(m')} = M^{(m)} = F$. By the triangle inequality,

$$\left| \Phi(M^{(m')}) - \Phi(M^{(0)}) \right| \leq \sum_{j=1}^{m'/R} \left| \Phi(M^{(jR)}) - \Phi(M^{((j-1)R)}) \right|.$$  \hspace{1cm} (6.10)

Now note that for each $j \in m'/R$, the matrix $M^{(jR^*)}$ is obtained from $M^{((j-1)R^*)}$ by applying a nonsingular operation acting on the left, affecting at most $2R$ rows. Denote the set of indices of the corresponding set of affected rows by $I_j$. (If the cardinality of $I_j$ is less than $2R$, then pad it with an arbitrary set of indices.) Using Lemma 6.3, this implies that for all $j \in [m'/R]$,

$$\left| \Phi(M^{(jR)}) - \Phi(M^{((j-1)R)}) \right| \leq \left( \|M^{(jR)}(I_j, :)\|_F \|M^{(jR)}(I_j, :)\|_F^{-T}(I_j, :)\|_F + \|M^{((j-1)R)}(I_j, :)\|_F \|M^{((j-1)R^*)}(I_j, :)\|_F \|M^{((j-1)R^*)}(I_j, :)\|_F^{-T}(I_j, :)\|_F \right) \log 2R.$$  \hspace{1cm} (6.11)
Combining (6.11) with (6.10), we get
\[
\left| \Phi(M^{(L)}) - \Phi(M^{(0)}) \right| \leq 2 \left( \sum_{j=0}^{m'/R} \| M^{(jR)}(I_j,: \cdot ) \|_F \| (M^{(jR)})^{-T}(I_j,: \cdot ) \|_F \right) \log 2R.
\]

For any matrix \( A \in \mathbb{R}^{n \times n} \) and any subset \( I \subseteq [n] \), we have \( \| A(I,: \cdot ) \|_F^2 \leq \sum_{i=1}^{|I|} \sigma_i^2(A) \) (this can be seen e.g. using the SVD theorem). Therefore,
\[
\left| \Phi(M^{(m')}) - \Phi(M^{(0)}) \right| \leq 2 \left( \sum_{j=0}^{m'/R} \sum_{i=1}^{2R} \sigma_i^2(M^{(jR)}) \sum_{i'=1}^{2R} \sigma_{i'}^2(M^{(jR)})^{-T} \right) \log 2R
\]
\[
= 2 \left( \sum_{j=0}^{m'/R} \sum_{i=1}^{2R} \sigma_i^2(M^{(jR)}) \sum_{i'=1}^{2R} \sigma_{n-i'+1}^2(M^{(jR)}) \right) \log 2R.
\]

But \( \Phi(M^{(m')}) = \Phi(F) = n \log n \) and \( \Phi(M^{(0)}) = 0 \), hence, there must exists \( j \in [m'/R] \) with
\[
\sqrt{2Rn \log n / m'} \geq \frac{2Rn \log n}{m' \log 2R} \geq \frac{2Rn \log n}{2((n \log n)/b + R) \log 2R}
\]
\[
\geq \frac{2Rn \log n}{2((n \log n)/b) \log 2R} = \frac{Rb}{\log 2R}.
\]

6.2 Proof of Theorem 5.3

Let \( V = \{v_1, \ldots, v_k\} \subseteq \mathbb{R}^n \) be an orthonormal set satisfying the properties described in case (i) of the theorem. Similarly, let \( U = \{u_1, \ldots, u_{\ell}\} \subseteq \mathbb{R}^n \) be an orthonormal set satisfying the properties described in case (ii) of the theorem. We show that as long as \( k + \ell \) is at most \( O(n) \), then we can extend one of the two sets by one element.

Let \( P \) denote the projection onto \( (\text{span}V)^{\perp} \) and \( Q \) the projection onto \( (\text{span}U)^{\perp} \). Using Lemma 6.1, we have that
\[
\Phi_{P,Q}(F) - \Phi_{P,Q}(\text{Id}) \geq n \log n - (\text{tr} \hat{P} + \text{tr} \hat{Q})(1 + \log n) - C(\| \hat{P} \|_F^2 + \| \hat{Q} \|_F^2) \log n,
\]
for some global \( C > 0 \), where \( \hat{P} = \text{Id} - P, \hat{Q} = \text{Id} - Q \) (the orthogonal projections). By known properties of projection matrices, \( \text{tr} \hat{P} = \| \hat{P} \|_F^2 = k, \text{tr} \hat{Q} = \| \hat{Q} \|_F^2 = \ell \). Therefore,
\[
\Phi_{P,Q}(F) - \Phi_{P,Q}(\text{Id}) \geq n \log n - C'(k + \ell) \log n,
\]
for some global \( C' > 0 \). This implies, using Theorem 5.1 (with \( R = 1 \)) that for some \( t \in \{0, \ldots, m\} \) and \( i \in [n] \):
\[
\| M^{(t)}P(i,:) \| \cdot \| (M^{(t)})^{-T}Q(i,:) \| \geq n \log n - C'(k + \ell) \log n
\]
\[
= \frac{m n \log n - C'(k + \ell) \log n}{n \log n}.
\]
Hence, as long as \( k + \ell \leq n/(2C') \):

\[
\| M^{(t)} P(i,:) \| \cdot \| (M^{(t)})^{-T} Q(i,:) \| \geq \frac{b}{2}.
\] (6.12)

This implies that either \( \| M^{(t)} P_{\perp}(i,:) \| \geq \sqrt{\frac{b}{2}} / 2 \) or \( \| (M^{(t)})^{-T} Q_{\perp}(i,:) \| \geq \sqrt{\frac{b}{2}} / 2 \). In the former case we can extend the set \( V \) by adding \( v_{k+1} = M^{(t)} P(i,:) / \| M^{(t)} P(i,:) \| \), which is orthogonal to \( v_1, \ldots, v_k \) by construction. In the latter case we can extend the set \( U \) by adding \( u_{k+1} = (M^{(t)})^{-T} Q(i,:) / \| (M^{(t)})^{-T} Q(i,:) \| \) which is again, orthogonal to \( u_1, \ldots, u_k \) by construction.

This process of augmenting \( V \) and \( U \) can continue until \( k + \ell \geq n/2C' \), which implies that either \( k \geq n/4C' \) (establishing extreme overflow of the theorem) or \( \ell \geq n/4C' \) (establishing extreme underflow).

Assume \( k \geq n/4C' \) and let \( n' = k \). For \( j \in [n'] \) let \( t_j \) denote the time step of the overflow corresponding to direction \( v_j \in \mathbb{R}^n \) and let \( i_j \) denote the coordinate at which the overflow occurs. It is clear from the construction that \( i_j \) is one of the at most two coordinates affected by the \( t_j \)th step. We show that there exist no \( 1 \leq j < j' \leq n' \) such that \( (t_j, i_j) = (t_{j'}, i_{j'}) \). Indeed, note that \( M^{(t_j)} P_{\perp}(i_j,:) \) must be null by construction, contradicting the fact that \( \| M^{(t_j)} P_{\perp}(i_j,:) \| = \| M^{(t_{j'})} P_{\perp}(i_{j'},:) \| = \Omega(\sqrt{b}) \). The conclusion is that for any \( t \in [m] \) there can be at most two indices \( j, j' \in [n'] \) such that \( t_j = t_{j'} = t \), and therefore the cardinality of the set \( \{t_1, \ldots, t_{n'}\} \) is at least \( n/2 \). A similar argument is done for the extreme underflow case, concluding the proof.

### 6.3 Proof of Lemma 6.4

**Proof.** We start by proving (6.8). For brevity, we denote \( F(i,j) \) by \( f_{i,j} \in \{1/\sqrt{n}, -1/\sqrt{n}\} \), \((FP)(i,j) = \epsilon_{i,j}, (FQ)(i,j) = \delta_{i,j} \). Therefore, \((FP)(i,j) = f_{i,j} - \epsilon_{i,j} \) and \((FQ)(i,j) = f_{i,j} - \delta_{i,j} \).

Let \( \|\epsilon\|_F = \sqrt{\sum_{i=1}^{n} \sum_{j=1}^{n} \epsilon_{i,j}^2} \) and \( \|\delta\|_F = \sqrt{\sum_{i=1}^{n} \sum_{j=1}^{n} \delta_{i,j}^2} \). By orthogonality of \( F \), we have that

\[
\|\epsilon\|_F = \|\hat{P}\|_F =: \alpha \quad (6.13)
\]

\[
\|\delta\|_F = \|\hat{Q}\|_F =: \beta \quad (6.14)
\]

\[
\sum_{i=1}^{n} \sum_{j=1}^{n} f_{i,j} \epsilon_{i,j} = \text{tr} F^T \hat{P} = \text{tr} \hat{P} \quad (6.15)
\]

\[
\sum_{i=1}^{n} \sum_{j=1}^{n} f_{i,j} \delta_{i,j} = \text{tr} F^T \hat{Q} = \text{tr} \hat{Q} . \quad (6.16)
\]

Let \( I_1, I_2, I_3, I_4 \subseteq [n] \times [n] \) be defined as

\[
I_1 := \{(i,j) : \epsilon_{i,j}^2 < 1/(2n) \text{ and } \delta_{i,j}^2 < 1/(2n)\}
\]

\[
I_2 := \{(i,j) : \epsilon_{i,j}^2 < 1/(2n) \text{ and } \delta_{i,j}^2 \geq 1/(2n)\}
\]

\[
I_3 := \{(i,j) : \epsilon_{i,j}^2 \geq 1/(2n) \text{ and } \delta_{i,j}^2 < 1/(2n)\}
\]

\[
I_4 := \{(i,j) : \epsilon_{i,j}^2 \geq 1/(2n) \text{ and } \delta_{i,j}^2 \geq 1/(2n)\} .
\]
Now,

$$\Phi(FP, FQ) = - \sum_{i=1}^{n} \sum_{j=1}^{n} (f_{i,j} - \epsilon_{i,j})(f_{i,j} - \delta_{i,j}) \log |(f_{i,j} - \epsilon_{i,j})(f_{i,j} - \delta_{i,j})|$$ \hspace{1cm} (6.17)

$$= \Phi_1 + \Phi_2 + \Phi_3 + \Phi_4,$$ \hspace{1cm} (6.18)

where \(\forall h = 1, 2, 3, 4: \Phi_h = - \sum_{(i,j) \in I_h} (f_{i,j} - \epsilon_{i,j})(f_{i,j} - \delta_{i,j}) \log |(f_{i,j} - \epsilon_{i,j})(f_{i,j} - \delta_{i,j})|.$$ 

We start by bounding \(\Phi_4\). For any \((i, j) \in I_4\),

$$|f_{i,j} - \epsilon_{i,j}| \leq 3|\epsilon_{i,j}| \hspace{1cm} (6.19)$$

$$|f_{i,j} - \delta_{i,j}| \leq 3|\delta_{i,j}|. \hspace{1cm} (6.20)$$

Write \(I_{4.1} \cup I_{4.2}\), where \(I_{4.1} = \{(i, j) : 9|\epsilon_{i,j}\delta_{i,j}| \leq 1/e\}\) and \(I_{4.2} = I_4 \setminus I_{4.1}\). Accordingly,

$$\Phi_{4.1} := - \sum_{(i,j) \in I_{4.1}} |(f_{i,j} - \epsilon_{i,j})(f_{i,j} - \delta_{i,j})| \log |(f_{i,j} - \epsilon_{i,j})(f_{i,j} - \delta_{i,j})|$$

$$\Phi_{4.2} := - \sum_{(i,j) \in I_{4.2}} |(f_{i,j} - \epsilon_{i,j})(f_{i,j} - \delta_{i,j})| \log |(f_{i,j} - \epsilon_{i,j})(f_{i,j} - \delta_{i,j})|.$$ 

Using (6.19)-(6.20) and the monotonicity (increasing) of \(-x \log x\) in the range \(x \in [0, 1/e]\), we conclude

$$|\Phi_{4.1}| \leq - \sum_{(i,j) \in I_4} 9|\epsilon_{i,j}\delta_{i,j}| \log 9|\epsilon_{i,j}\delta_{i,j}|$$

$$= - \sum_{(i,j) \in I_4} 9||\epsilon||||\delta|| \frac{|\epsilon_{i,j}\delta_{i,j}|}{||\epsilon||||\delta||} \log 9||\epsilon||||\delta||$$

$$= - \sum_{(i,j) \in I_4} 9||\epsilon||||\delta|| \frac{|\epsilon_{i,j}\delta_{i,j}|}{||\epsilon||||\delta||} \log 9||\epsilon||||\delta|| - \sum_{(i,j) \in I_4} 9||\epsilon||||\delta|| \frac{|\epsilon_{i,j}\delta_{i,j}|}{||\epsilon||||\delta||} \log \frac{|\epsilon_{i,j}\delta_{i,j}|}{||\epsilon||||\delta||}$$

$$\leq 9\alpha\beta \log 9\alpha\beta + 18\alpha\beta \log n$$

$$\leq 27\alpha\beta \log n + 9\alpha\beta \log 9.$$ 

where the second inequality used (6.13), Lemma 6.1 and Cauchy-Schwarz. To bound \(|\Phi_{4.2}|\), note that by Cauchy-Schwarz \(\sum_{(i,j) \in I_{4.2}} |\epsilon_{i,j}\delta_{i,j}| \leq \alpha\beta\), and hence \(|I_{4.2}| \leq 9\alpha\beta \leq 27\alpha\beta\). This implies that \(|\Phi_{4.2}| \leq 27\alpha\beta\). Combining, we conclude

$$|\Phi_4| \leq |\Phi_{4.1}| + |\Phi_{4.2}| \leq 27\alpha\beta \log n + 63\alpha\beta \leq (\alpha^2 + \beta^2)(63 + 27 \log n). \hspace{1cm} (6.21)$$

We now bound \(|\Phi_3|\). For all \((i, j) \in I_3\), (6.19) holds. Again we need to consider two cases, by defining \(I_{3.1} := \{(i, j) \in I_3 : 3|\epsilon_{i,j}| \leq 1/e\}\) and \(I_{3.2} = I_3 \setminus I_{3.1}\) and, as above, \(\Phi_{3.1}\) and \(\Phi_{3.2}\) in an
obvious way. Then,
\[
|\Phi_{3,1}| \leq -3 \sum_{(i,j) \in I_{3,1}} |\epsilon_{i,j}||f_{i,j} - \delta_{i,j}| \log(3|\epsilon_{i,j}||f_{i,j} - \delta_{i,j}|)
\]
\[
= -3 \sum_{(i,j) \in I_{3,1}} |\epsilon_{i,j}||f_{i,j} - \delta_{i,j}| \log 3|\epsilon_{i,j}||f_{i,j}|-3 \sum_{(i,j) \in I_{3,1}} |\epsilon_{i,j}||f_{i,j} - \delta_{i,j}| \log |1 - \delta_{i,j}/f_{i,j}|
\]
\[
\leq 3 \sum_{(i,j) \in I_{3,1}} |\epsilon_{i,j}||f_{i,j} - \delta_{i,j}| \log n + 3 \sum_{(i,j) \in I_{3,1}} |\epsilon_{i,j}||f_{i,j} - \delta_{i,j}||\delta_{i,j}/f_{i,j}|
\]
\[
\leq 3 \sum_{(i,j) \in I_{3,1}} |\epsilon_{i,j}| \frac{1}{2\sqrt{n}} \log n + 3 \sum_{(i,j) \in I_{3,1}} |\epsilon_{i,j}|/(2\sqrt{2n}) \tag{6.22}
\]

By Cauchy-Schwarz (or, alternatively, the $\ell_1/\ell_2$ ratio bound inequality),
\[
\sum_{(i,j) \in I_{3,1}} |\epsilon_{i,j}| \leq \sqrt{|I_{3,1}|} \sqrt{\sum_{(i,j) \in I_{3,1}} \epsilon_{i,j}^2} \leq \sqrt{|I_{3}|}\|\epsilon\| = \sqrt{|I_3|}\alpha .
\]

But by [6.13] and definition of $I_3$: $|I_3| \leq 2\alpha^2 n$. Combining with (6.22), we get
\[
|\Phi_{3,1}| \leq 3\alpha^2 \log n + 3\alpha^2 . \tag{6.23}
\]

To bound $|\Phi_{3,2}|$, note that by (6.13) and by definition of $I_{3,2}$, $|I_{3,2}| \leq 9\alpha^2 \alpha^2 \leq 81\alpha^2$. But clearly $|\Phi_{3,2}| \leq |I_{3,2}|$, hence $|\Phi_{3,2}| \leq 81\alpha^2$. Combining with (6.23), we conclude
\[
|\Phi_3| \leq 3\alpha^2 \log n + 84\alpha^2 . \tag{6.24}
\]

By symmetry, we also have:
\[
|\Phi_2| \leq 3\beta^2 \log n + 84\beta^2 . \tag{6.25}
\]

We now turn to approximate $\Phi_1$.
\[
\Phi_1 = -\sum_{(i,j) \in I_1} f_{i,j}^2 \log f_{i,j}^2 - \sum_{(i,j) \in I_1} f_{i,j}^2 \log((1 - \epsilon_{i,j}/f_{i,j})(1 - \delta_{i,j}/f_{i,j}))
\]
\[
+ \sum_{(i,j) \in I_1} f_{i,j}(\epsilon_{i,j} + \delta_{i,j}) \log((f_{i,j} - \epsilon_{i,j})(f_{i,j} - \delta_{i,j}))
\]
\[
- \sum_{(i,j) \in I_1} \epsilon_{i,j} \delta_{i,j} \log((f_{i,j} - \epsilon_{i,j})(f_{i,j} - \delta_{i,j}))
\]

Subtracting $n \log n = -\sum_{i=1}^n \sum_{j=1}^n f_{i,j}^2 \log f_{i,j}^2$ from both sides gives:
\[
\Phi_1 - n \log n = \sum_{(i,j) \notin I_1} f_{i,j}^2 \log f_{i,j}^2 - A + B - C .
\]
By (6.13), the number of pairs \((i, j) \in [n] \times [n]\) for which \(\epsilon_{i, j}^2 \geq 1/(2n)\) is at most \(2\alpha^2 n\). Similarly, the number of pairs \((i, j)\) for which \(\delta_{i, j}^2 \geq 1/(2n)\) is at most \(2\beta^2 n\). Hence,

\[
\left|([n] \times [n]) \setminus I_1\right| \leq 2(\alpha^2 + \beta^2)n . \tag{6.26}
\]

Therefore, \(|D| \leq 2(\alpha^2 + \beta^2)n(1/n) \log n = 2(\alpha^2 + \beta^2) \log n\).

For \(A\), we just need to notice that:

\[
A \leq 0 . \tag{6.27}
\]

We now bound \(B\).

\[
B = \sum_{(i,j) \in I_1} f_{i,j}(\epsilon_{i,j} + \delta_{i,j}) \log f_{i,j}^2 + \sum_{(i,j) \in I_1} f_{i,j}(\epsilon_{i,j} + \delta_{i,j}) \log (1 - \epsilon_{i,j}/f_{i,j})(1 - \delta_{i,j}/f_{i,j})
\]

\[
= - \sum_{(i,j) \in I_1} f_{i,j}(\epsilon_{i,j} + \delta_{i,j}) \log n + \sum_{(i,j) \in I_1} f_{i,j}(\epsilon_{i,j} + \delta_{i,j}) \log (1 - \epsilon_{i,j}/f_{i,j})(1 - \delta_{i,j}/f_{i,j})
\]

Adding \(\sum_{i=1}^{n} \sum_{j=1}^{n} f_{i,j}(\epsilon_{i,j} + \delta_{i,j}) \log n = (\text{tr } \hat{P} + \text{tr } \hat{Q}) \log n \) (see (6.15) and (6.16)) to both sides, taking absolute value on both sides, using the triangle inequality and the estimate \(\log(1 - \alpha) \leq |\alpha|\) for all \(|\alpha| \leq 1/\sqrt{2}\):

\[
\left|B + (\text{tr } \hat{P} + \text{tr } \hat{Q}) \log n\right| \leq \sum_{(i,j) \notin I_1} \left|f_{i,j}(|\epsilon_{i,j}| + |\delta_{i,j}|) \log n + \sum_{(i,j) \in I_1} \left(\epsilon_{i,j}^2 + \delta_{i,j}^2 + 2|\epsilon_{i,j}||\delta_{i,j}|\right)\right|
\]

\[
\leq \sum_{(i,j) \notin I_1} \frac{1}{\sqrt{n}}(|\epsilon_{i,j}| + |\delta_{i,j}|) \log n + 3(\alpha^2 + \beta^2)
\]

\[
\leq 2(\alpha^2 + \beta^2) \log n + 3(\alpha^2 + \beta^2) .
\]

Where the second inequality used Cauchy-Schwarz, and the third used Cauchy-Schwarz to obtain \(\sum_{(i,j) \notin I_1} |\epsilon_{i,j}| \leq \sqrt{n^2 - |I_1|\alpha}\) together with the estimate \((n^2 - |I_1|) \leq 2n\alpha^2\) (from the definition of \(I_1\) and \(\alpha\)), and a similar step for bounding \(\sum_{(i,j) \notin I_1} |\delta_{i,j}|\).

To bound \(|C|\), note that \(|\log((f_{i,j} - \epsilon_{i,j})(f_{i,j} - \delta_{i,j}))| \leq 4\log n\) for all \((i, j) \in I_1\). Hence, using Cauchy-Schwarz,

\[
|C| \leq 4 \sum_{(i,j) \in I_1} |\epsilon_{i,j}||\delta_{i,j}| \log n \leq 4\alpha\beta \log n \leq 4(\alpha^2 + \beta^2) \log n .
\]

Combining our upper bound for \(A\) and estimates for \(B\), \(|C|\) and \(|D|\), we conclude:

\[
\Phi_1 \geq n \log n - (\text{tr } \hat{P} + \text{tr } \hat{Q}) \log n - (\alpha^2 + \beta^2)(6 + 8 \log n) . \tag{6.28}
\]

Finally, by combining (6.21), (6.24), (6.25), (6.28) and (6.18), we conclude

\[
\Phi(FP, FQ) \geq n \log n - (\text{tr } \hat{P} + \text{tr } \hat{Q}) \log n - (\alpha^2 + \beta^2)(147 + 30 \log n) .
\]

This concludes the proof of (6.8).

We now prove (6.9), whence we assume that \(P, Q\) are PSD contractions (as are \(\hat{P}, \hat{Q}\)). We decompose \(\Phi_{P,Q}(\text{Id, Id})\) as two sums, as follows:

\[
\Phi_{\text{diag}} = - \sum_{i=1}^{n} P(i, i)Q(i, i) \log |P(i, i)Q(i, i)|
\]

\[
\Phi_{\text{off}} = - \sum_{i \neq j} P(i, j)Q(i, j) \log |P(i, j)Q(i, j)| .
\]

18
We start by bounding $\Phi_{\text{diag}}$. Define $\eta$ as:

$$
\eta := \sum_{i=1}^{n} P(i, i)Q(i, i) = \sum_{i=1}^{n} (1 - \hat{P}(i, i))(1 - \hat{Q}(i, i)) .
$$

Notice that $\eta \geq n - \text{tr} \hat{P} - \text{tr} \hat{Q}$, and that $0 \leq P(i, i)Q(i, i) \leq 1$ for all $i \in [n]$ by the contraction property. Using standard tools (e.g. Lagrange multipliers), it can be shown that the function $-\sum y_i \log y_i$ under the constraints $0 \leq y_i \leq 1$, $\sum y_i = \eta$ obtains its maximum when $\forall i \in [n] : y_i = 1/\eta$, at which case its value is $-\eta \log(\eta/n)$. Hence,

$$
\Phi_{\text{diag}} \leq -\eta \log(\eta/n) \leq -(n - \text{tr} \hat{P} - \text{tr} \hat{Q}) \log(1 - \text{tr} \hat{P}/n - \text{tr} \hat{Q}/n)
$$

$$
\leq -n \log(1 - \text{tr} \hat{P}/n - \text{tr} \hat{Q}/n)
$$

$$
\leq n(\text{tr} \hat{P}/n + \text{tr} \hat{Q}/n + (\text{tr} \hat{P} + \text{tr} \hat{Q})^2/n^2)
$$

$$
= \text{tr} \hat{P} + \text{tr} \hat{Q} + (\text{tr} \hat{P} + \text{tr} \hat{Q})^2/n
$$

$$
\leq \text{tr} \hat{P} + \text{tr} \hat{Q} + \|\hat{P}\|_F^2 + \|\hat{Q}\|_F^2 .
$$

(we used twice the assumption that $n - \text{tr} \hat{P} - \text{tr} \hat{Q} \geq n/e$, for otherwise $\|\hat{P}\|_F^2 + \|\hat{Q}\|_F^2 = \Omega(n)$ and (6.9) is trivial.)

We now turn to bound $\Phi_{\text{off}}$. Let $\mu_P = \sum_{i\neq j} P(i, j)^2$, $\mu_Q = \sum_{i\neq j} Q(i, j)^2$. If $\mu_P\mu_Q \leq 1$ then without loss of generality $\mu_P \leq 1$, implying $\|\hat{P}\|_F = \Omega(n)$, and therefore (6.9) is trivial. Hence we assume $\mu_P\mu_Q \geq 1$. Using Lemma 6.1 (define $x$ to be the $n(n-1)$-dimensional vector with $x_{ij} = P(i, j)/\sqrt{\mu_P}$ for $i \neq j$, and similarly define $y$ using $Q$):

$$
\Phi_{\text{off}} \leq \sqrt{\mu_P\mu_Q} \log n(n-1) - \sum_{i\neq j} P(i, j)Q(i, j) \log \sqrt{\mu_P\mu_Q}
$$

$$
\leq 2\sqrt{\mu_P\mu_Q} \log n \leq (\mu_P + \mu_Q) \log n ,
$$

where we used the AMGM inequality in the last step. But now notice that $\mu_P \leq \|\hat{P}\|_F^2$, $\mu_Q \leq \|\hat{Q}\|_F^2$. Combining this with our bound of $\Phi_{\text{diag}}$ completes the proof.

## 7 Future Work

Taking into account bit operation complexity, and using state-of-the-art integer multiplication algorithms [8, 9] it can be quite easily shown that both severe overflow and severe underflow could be resolved by allowing flexible word size, accommodating either large numbers (in the overflow case) or increased accuracy (in the underflow case). In fact, allowing $O(\log b)$-bit words at the time steps at which overflow (or underflow) occur, of which there are $\Omega(n)$ many by Theorem 5.3, suffices. Hence, this work does not rule out the possibility of (in the extreme case of $b = \Theta(\log n)$) a Fourier transform algorithm in the in-place model using a linear number of gates, in bit operation complexity of $\tilde{\Omega}(n \log \log n)$, where $\tilde{O}()$ here hides log log log $n$ factors arising from fast integer multiplication algorithms. We conjecture that such an algorithm does not actually exist, and leave this as the main open problem.

Another problem that was left out in this work is going beyond the in-place model. In the more general model, the algorithm works in space $\mathbb{R}^\ell$ for $\ell > n$, where the $(\ell - n)$ extra coordinates
can be assumed to be initialized with 0, and the first $n$ are initialized with the input $x \in \mathbb{R}^n$. The final matrix $M^{(m)}$ of Fourier transform algorithm $A_n = \{\text{Id} = M^{(0)}, \ldots, M^{(m)}\}$ contains $F$ as a sub matrix, so that the output $Fx$ can simply be extracted from a subset of $n$ coordinates of $M^{(m)}x$, which can be assumed to be the first. The matrix $M^{(m)}$ (and its inverse-traspose) therefore contains $(\ell - n)$ extra rows. The submatrix defined by the extra rows (namely, the last $\ell - n$) and the first $n$ columns were referred to in [2] as the “garbage” part of the computation. To obtain an $\Omega(n \log n)$ computational lower bound in the model assumed there, it was necessary to show that $\Phi_{P,P}(M^{(m)}) P = \Omega(n \log n)$, where $P \in \mathbb{R}^{\ell \times \ell}$ is projection onto the space spanned by the first $n$ standard basis vectors. To that end, it was shown that such a potential lower bound held as long as spectral norm of the “garbage” submatrices was properly upper bounded. That result, in fact, can be deduced as a simple outcome of Lemma 6.4 that was developed here. What’s more interesting is how to generalize Theorem 5.3 to the non in-place model, and more importantly how to analyze the numerical accuracy implications of overflow and underflow to the non in-place model. Such a generalization is not trivial and is an immediate open problem following this work.

The closing question of Section 5 is another avenue of possible research stemming from this work, and seems (to the author) as an achievable goal in the quest to solve the half decade old problem of understanding the complexity of Fourier transform. In view of this work and of the preceding [1, 2], it seems that the correct way to understand the hardness of this important transformation is by way of trading off time versus accuracy, for a given machine architecture.

Another interesting possible avenue is to study the complexity of Fourier transform on input $x$ for which some prior knowledge is known. The best example is when $Fx$ is assumed sparse, for which much interesting work on the upper bound side has been recently done by Indyk et al. (see [10] and references therein).

Many algorithms use the Fourier transform as a subroutine. In certain cases (fast polynomial multiplication, fast integer multiplication [8, 9], fast Johnson-Lindenstrauss transform for dimensionality reduction [3, 4, 5, 11] and the related restricted isometry property (RIP) matrix construction [14, 6, 11]) the Fourier transform subroutine is the algorithm’s bottleneck. Can we use the techniques developed here to derive lower bounds (or rather, time-accuracy tradeoffs) for those algorithms as well? Moreover, we can ask how the implications of speeding up the Fourier transform subroutine (as derived in this work) affect the numerical outcome of these algorithms, assuming they insist on using Fourier transform as a black box.

### References

1. Nir Ailon. A lower bound for fourier transform computation in a linear model over 2x2 unitary gates using matrix entropy. *Chicago J. of Theo. Comp. Sci.*, 2013.

2. Nir Ailon. An $n \log n$ lower bound for fourier transform computation in the well conditioned model. *arXiv:1403.1907*, 2014.

3. Nir Ailon and Bernard Chazelle. The fast johnson–lindenstrauss transform and approximate nearest neighbors. *SIAM J. Comput.*, 39(1):302–322, 2009.

---

6In [2], the model simply assumed that all matrices $M^{(t)}$ for $t=1 \ldots m$ have bounded condition number. Quantifying the effect of ill condition on numerical stability, overflow and underflow, was not done there.

7The function $\Phi_{P,Q}(M)$ was not defined in [2], and was only implicitly used.
[4] Nir Ailon and Edo Liberty. Fast dimension reduction using rademacher series on dual BCH codes. *Discrete & Computational Geometry*, 42(4):615–630, 2009.

[5] Nir Ailon and Edo Liberty. An almost optimal unrestricted fast johnson-lindenstrauss transform. *ACM Transactions on Algorithms*, 9(3):21, 2013.

[6] Nir Ailon and Holger Rauhut. Fast and rip-optimal transforms. *Discrete and Comp. Geom. (DCG)*, 2014 (in press).

[7] J. W Cooley and J. W Tukey. An algorithm for the machine computation of complex Fourier series. *J. of American Math. Soc.*, pages 297–301, 1964.

[8] Anindya De, Piyush P. Kurur, Chandan Saha, and Ramprasad Saptharishi. Fast integer multiplication using modular arithmetic. *SIAM J. on Comp.*, 42, 2013.

[9] Martin Fürer. Faster integer multiplication. In *Proceedings of the Thirty-ninth Annual ACM Symposium on Theory of Computing*, STOC ’07, pages 57–66, 2007.

[10] 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, SODA 2014, Portland, Oregon, USA, January 5-7, 2014*, pages 480–499, 2014.

[11] Felix Krahmer and Rachel Ward. New and improved johnson-lindenstrauss embeddings via the restricted isometry property. *SIAM J. Math. Analysis*, 43(3):1269–1281, 2011.

[12] Jacques Morgenstern. Note on a lower bound on the linear complexity of the fast Fourier transform. *J. ACM*, 20(2):305–306, April 1973.

[13] Christos H. Papadimitriou. Optimality of the fast Fourier transform. *J. ACM*, 26(1):95–102, January 1979.

[14] Mark Rudelson and Roman Vershynin. Sampling from large matrices: An approach through geometric functional analysis. *J. ACM*, 54(4), 2007.

[15] S. Winograd. On computing the discrete Fourier transform. *Proc. Nat. Assoc. Sci.*, 73(4):1005–1006, 1976.