Inverse Nonlinear Fourier Transform: Applications in QSP

B10901042 Wen Perng
B10901017 Po-chieh Liu

Department of Electrical Engineering, National Taiwan University, Taipei, Taiwan

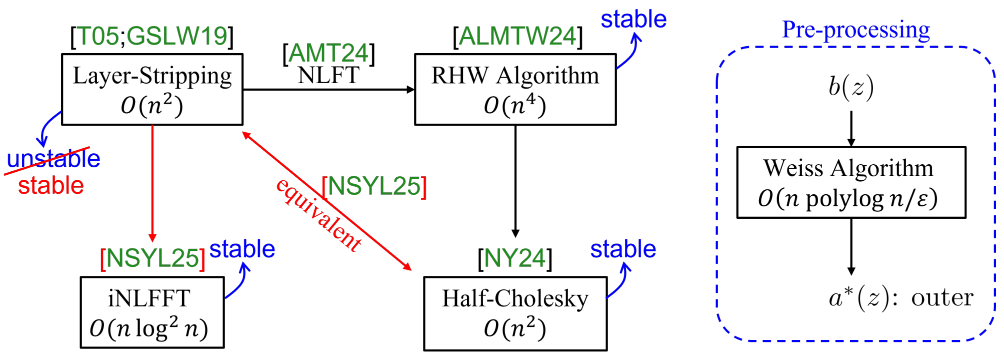

FIGURE 1: A road map to the development of the application of inverse nonlinear Fourier transform in quantum signal processing.

Abstract – Quantum signal processing is a powerful framework encompassing many quantum algorithms, enabling polynomial transformations of the singular values of matrix block-encodings. Finding the phase factors in QSP protocols is a critical step in implementing these transformations. Traditional phase-finding methods, such as the layer-stripping method, while intuitive, face challenges with numerical instability. In recent years, the introduction of the relationship between nonlinear Fourier transform and quantum signal processing has provided new perspectives and tools to address this task. This paper reviews the recent development of quantum signal processing phase-finding methods based on nonlinear Fourier transform, including the Riemann–Hilbert–Weiss algorithm, the Half-Cholesky method, and the state-of-the-art inverse nonlinear fast Fourier transform. The mathematical principles, computational complexities, and numerical stability of these algorithms are detailed, with particular emphasis on the crucial role of the *outerness* condition of the complementary polynomial $a^*(z)$ in ensuring stability. Finally, this paper summarizes current research progress and looks ahead to potential future research directions, aiming to promote the broader application of the nonlinear Fourier transform perspective in the field of quantum signal processing and quantum computing.

13 June 2025

1. INTRODUCTION

The quantum signal processing (QSP) protocol provides a systematic procedure to apply an arbitrary function on a block-encoding. Its usage is ubiquitous, being used in Hamiltonian simulation, linear system solvers, and much more. Thus, it is lauded as the “grand unification of quantum algorithms [MRTC21].”

The QSP protocol is succinctly summarized as below: given the signal unitary

$$W = \begin{bmatrix} x & \mathrm{i}\sqrt{1-x^2} \\ \mathrm{i}\sqrt{1-x^2} & x \end{bmatrix} = \mathrm{e}^{\mathrm{i}(\arccos x)X} \tag{1}$$

and the control unitary

$$R(\phi) = \mathrm{e}^{\mathrm{i}\phi Z}, \tag{2}$$

where X and Z are Pauli matrices, find the sequence of phases $\Phi = (\phi_0, \phi_1, \ldots, \phi_n)$ such that we have

$$\begin{bmatrix} P(x) & \mathrm{i}Q(x)\sqrt{1-x^2} \\ * & * \end{bmatrix} = \mathrm{e}^{\mathrm{i}\phi_0 Z} W(x) \mathrm{e}^{\mathrm{i}\phi_1 Z} W(x) \cdots W(x) \mathrm{e}^{\mathrm{i}\phi_n Z} =: V_\Phi(x). \tag{3}$$

The degree n polynomial $f(x) = \mathrm{Im}\{P(x)\}$ with definite parity $n \mod 2$ is the target polynomial chosen depending on the task desired. Both the polynomials P and Q are termed the *complementary polynomials* to f. The choice to Φ follows suit from the choice of P and Q. How exactly are the phases determined?

See Figure 1, since the introduction of QSP in the seminal work by Gilyén et al. [GSLW19], plenty methods have been proposed in solving the sequence of phases given the function $P(x)$. The earliest of which is the *layer-stripping method*. By coming up with a complementary polynomial Q, one iteratively reduces both the degrees to P and Q to "strip away" the phases from ϕ_n down to ϕ_0. The phase factors are computed sequentially, with a time complexity of $O(n^2)$. One of the first explicit construction to the complementary polynomials is given by Haah [H19]. However, in the same work, it is also shown that the layer-stripping method is unstable: requiring $O(n \log(n/\varepsilon))$ bits of precision to ensure $\left\|\hat{f} - f\right\|_\infty < \varepsilon$. A numerically stable algorithm should only require $O(\mathrm{polylog}(n/\varepsilon))$ bit of precision, such that the error will not blow-up exponentially.

Later methods took an entirely different approach to finding the phase sequence. Some works focus on using polynomial decomposition [Y22] to figure out the phase factors. Works by Dong, Lin, Ni, and Wang [DLNW24a; WDL22; DLNW24b] turn the phase finding problem into an optimization problem. To state it explicitly, in these work, they often worked with a symmetric phase scheme to cope with the extra degrees of freedom from the restriction of the parity of f. Their methods then re-parameterize the target function and the constructed function via their Chebyshev coefficients. A loss function was constructed as the ℓ^2-difference in the Chebyshev coefficients with the symmetric phase as the variable. Then, by directly apply the Newton's method [DLNW24b] or a fixed-point iteration [DLNW24a] over the loss function, it is expected that the loss function converge to a minimum, successfully retrieving the phase. However, it was also shown that the energy landscape to this minimization problem is very complicated [WDL22], with many local minima that hinder the algorithm from success. Henceforth, starting the iterative optimization with a suitable initial condition is fairly important. Luckily, it was shown that by initializing at the origin (all-zero phase), the algorithm converges to a *global* minimum, with this solution termed the *maximal solution*. These methods have a complexity of $O(n^2 \log \varepsilon^{-1})$ and are stable. However, these methods fail in the extreme cases of close-to-coherent regime: when $\|f\|_\infty = 1$, the QSP protocol is fully-coherent; when $\|f\|_\infty = 1-\eta \lesssim 1$, the protocol is close-to-coherent; whereas when $\|f\|_\infty \ll 1$, the protocol is incoherent.

Next, in 2024, the groundbreaking work by Alexis et al. [AMT24] linked the QSP protocol with the long developed field of nonlinear Fourier transform (NLFT) over the special unitary matrices SU(2). The NLFT transforms a sequence of complex numbers into an SU(2) matrix. In fact, the layer-stripping method was proposed long ago in 2005 in the PhD dissertation of Tsai [T05]. It was shown that the mathematical framework to QSP is a special case of NLFT, whereas the mathematical framework of GQSP is exactly the same as that of NLFT [L25]. To find the phase to the QSP (GQSP) protocol, one now translates into solving the inverse NLFT problem: given an SU(2) matrix that is promised to be generated from a sequence through NLFT, find the sequence of complex numbers. In the continued work by Alexis et al. [ALMTW24], they combined the Weiss algorithm with the Riemann–Hilbert algorithm to create the Riemann–Hilbert–Weiss (RHW) algorithm to solve the phase-finding problem. The Weiss algorithm is used to generate a *unique* complementary polynomial that satisfies the *outerness* condition, which proves to be very crucial in the stability of the algorithm later on. The Riemann–Hilbert algorithm is used to solve the Riemann–Hilbert factorization problem. This algorithm allows independent computation of all the phase factors, with a time complexity of $O(n^4)$. Later, the half-Cholesky (HC) method [NY24] was proposed to speed up the matrix decomposition task in the Riemann–Hilbert algorithm due to the *displacement structure* within, successfully bringing the time complexity down to $O(n^2)$.

Recently, a new method termed the *inverse nonlinear fast Fourier transform* (iNLFFT) was proposed by Ni, Sarkar, Ying, and Lin [NSYL25], successfully bringing down the time complexity to $O(n \log^2 n)$ through a divide-and-conquer scheme. More importantly, they pointed out the direct link between the HC method and layer-stripping, showing them to be equivalent. What's more, they shown that the outerness of the complementary polynomial is

Algorithm	Time Complexity	Coherence	Stability
FPI [DLNW24a]	$\tilde{O}(n^2 \log \varepsilon^{-1})$	$1-\eta \ll 1$	Y
Newton's [DLNW24b]	$O(n^2 \log \varepsilon^{-1})$	$1-\eta \ll 1$	Y
Layer-Stripping [T05; GSLW19]	$O(n^2)^*$	Full	N/Y
RHW [ALMTW24]	$O(n^4)^*$	Full	Y
HC [NY24]	$O(n^2)^*$	Full	Y
iNLFFT [NSYL25]	$O(n \log^2 n)^*$	Full	Y

TABLE 1: Comparison of phase-finding algorithms.

what determines the stability of the inverse problem. In fact, the instability that Haah showed [H19] was a worst case analysis. Finally, incorporating the Weiss algorithm, the total time complexity to iNLFFT is $O(n \operatorname{polylog}(n/\varepsilon))$. A short summary of the algorithms mentioned for the phase-finding problem of the QSP protocol is in Table 1. Note that the algorithms with * requires the Weiss algorithm, costing an additional $O(n \operatorname{polylog}(n/\varepsilon))$.

We have introduced the history and development to the intersection of QSP and NLFT in Section 1. Next, in Section 2, we introduce the essential definitions and properties of NLFT that will be used in this writing. Further, the explicit connection between NLFT and QSP is also given. In Section 3, we describe the details to the inverse NLFT algorithms, this includes the Weiss algorithm for finding an outer complementary polynomial, the RHW algorithm, and the iNLFFT algorithm. The latter is the state-of-the-art, while the RHW algorithm is useful in the stability analysis that is developed in Section 4. We conclude this exploration into the application of NLFT in QSP in Section 5.

2. NONLINEAR FOURIER TRANSFORM

Just like how the linear Fourier transformation is invented to deal with the heat diffusion equation, which is a linear partial differential equation, the nonlinear Fourier transform is invented to solve nonlinear partial differential equations. These earliest works deal with continuous NLFT over groups such as SU(2), which is a transform from a function defined over $\mathbb{R}$ to an SU(2) matrix. Other variants exist such as working over the group SU(1, 1): the difference in the metric results in a different mathematical structure of the transformation. In our work, however, we mainly work with the discrete variant of NLFT, transforming a sequence over $\mathbb{Z}$ into an SU(2) matrix.

In this section, let us introduce the definition to the nonlinear Fourier transform and some basic properties of it that will be used as the basis of inspiration for the inverse nonlinear Fourier transform discussed in Section 3.

Definition 2.1. *Some sets are crucial to our discussion: The unit circle is denoted by* $\mathbb{T} = \left\{ z \in \mathbb{C} \,\middle|\, |z| = 1 \right\}$, *the unit disc is denoted by* $\mathbb{D} = \left\{ z \in \mathbb{C} \,\middle|\, |z| < 1 \right\}$. *Furthermore, we have* $\mathbb{D}^* = \left\{ z \in \mathbb{C} \cup \{\infty\} \,\middle|\, |z| > 1 \right\}$, *and the closed unit disc* $\overline{D} = \mathbb{T} \cup \mathbb{D}$.

Definition 2.2 (Outer Function). *A function is outer if it has no zeros in* $\mathbb{D}$.

As mentioned before, outerness is a crucial property that determines the stability of our algorithms. For a function to be outer, there exists plenty different equivalent definitions, the current one provided is useful in bounding the error to the algorithms. As for actually constructing an outer function, Definition 3.1 introduces another definition, and it is used in Section 3.1.

Definition 2.3. *Denote the following conjugation*

$$a^*(z) = \overline{a(\overline{z}^{-1})}, \tag{4}$$

where $\bar{z}$ *is the usual complex conjugation. This conjugation preserves the analyticity of a function from the unit disc* $\mathbb{D}$ *to the region outside the disc* $\mathbb{D}^*$.

A prime example that demonstrates the effect of this conjugation is:

$$f(z) = cz^n \Rightarrow f^*(z) = \bar{c}z^{-n}. \tag{5}$$

Definition 2.4 (Nonlinear Fourier Transform). *Let* $F \in \ell^0(\mathbb{Z}, \mathbb{C})$ *be a compactly supported sequence on* $[m : n]$, *its* SU(2) *nonlinear Fourier transform on* $\mathbb{C} \cup \{\infty\}$ *is defined as the ordered product*

$$\mathcal{F}(z) = (a, b) = \begin{bmatrix} a(z) & b(z) \\ -b^*(z) & a^*(z) \end{bmatrix} := \overrightarrow{\prod_{k=m}^{n}} \frac{1}{\sqrt{1 + |F_k|^2}} \begin{bmatrix} 1 & F_k z^k \\ -\overline{F}_k z^{-k} & 1 \end{bmatrix}. \tag{6}$$

One often simply use the tuple (a, b) *in place of the whole NLFT* 2×2 *matrix.*

In literature, one often uses the notation $\widehat{F}$ or $\overbrace{[F_m, \dots, F_n]}$ to represent the NLFT of the sequence $F = (F_m, \dots, F_n)$. However The emphasis on "SU(2)" is due to the iteration always staying within the group SU(2) for $|z| = 1$, which also implies nonlinear Fourier transform over different group structures. However, throughout this script, we will only working on SU(2), hence the term will be neglected and we will only mention the transform by the abbreviation NLFT.

2.1. Relating QSP and NLFT

A more general framework of QSP is termed the general QSP, or GQSP for short. It generalizes the control unitaries to

$$R(\psi, \phi) = \begin{bmatrix} \cos\psi & \mathrm{e}^{\mathrm{i}\phi}\sin\psi \\ -\mathrm{e}^{-\mathrm{i}\phi}\sin\psi & \cos\psi \end{bmatrix}. \tag{7}$$

With the signal matrices modified to

$$W(z) = \begin{bmatrix} z & 0 \\ 0 & 1 \end{bmatrix}, \tag{8}$$

the GQSP protocol is the ordered product

$$R(\psi_0, \phi_0) \cdot \overrightarrow{\prod_{k=1}^{n}} W(z) R(\psi_k, \phi_k) = \begin{bmatrix} * & f(z) \\ * & * \end{bmatrix} \tag{9}$$

with the desired function f encoded in the top-right corner. The GQSP encompasses the QSP protocol as a special case [L25]. With the definition to NLFT and GQSP given, let us see how they are equivalent: by setting $F_k = \mathrm{e}^{\mathrm{i}\phi_k}\tan\psi_k$ and $m = 0$, we have

$$\begin{aligned}
\mathcal{F}(z) = \begin{bmatrix} * & b(z) \\ * & * \end{bmatrix} &= \overrightarrow{\prod_{k=m}^{n}} \frac{1}{\sqrt{1+|F_k|^2}} \begin{bmatrix} 1 & F_k z^k \\ -\overline{F}_k z^{-k} & 1 \end{bmatrix} \\
&= \overrightarrow{\prod_{k=m}^{n}} \frac{1}{\sqrt{1+|F_k|^2}} \begin{bmatrix} z^k & \\ & 1 \end{bmatrix} \begin{bmatrix} 1 & F_k \\ -\overline{F}_k & 1 \end{bmatrix} \begin{bmatrix} z^{-k} & \\ & 1 \end{bmatrix} \\
&= \begin{bmatrix} z^{m-1} & \\ & 1 \end{bmatrix} \left(\overrightarrow{\prod_{k=m}^{n}} \frac{1}{\sqrt{1+|F_k|^2}} \begin{bmatrix} z & \\ & 1 \end{bmatrix} \begin{bmatrix} 1 & F_k \\ -\overline{F}_k & 1 \end{bmatrix} \right) \begin{bmatrix} z^{-n} & \\ & 1 \end{bmatrix} \\
&= \begin{bmatrix} z^{m-1} & \\ & 1 \end{bmatrix} \left(\overrightarrow{\prod_{k=m}^{n}} W(z) R(\psi_k, \phi_k) \right) \begin{bmatrix} z^{-n} & \\ & 1 \end{bmatrix} \\
&= \begin{bmatrix} z^{m-1} & \\ & 1 \end{bmatrix} \left(W(z) R(\psi_m, \phi_m) \cdot \overrightarrow{\prod_{k=m+1}^{n}} W(z) R(\psi_k, \phi_k) \right) \begin{bmatrix} z^{-n} & \\ & 1 \end{bmatrix} \\
&= R(\psi_0, \phi_0) \cdot \overrightarrow{\prod_{k=1}^{n}} W(z) R(\psi_k, \phi_k) \cdot \begin{bmatrix} z^{-n} & \\ & 1 \end{bmatrix} = \begin{bmatrix} * & f(z) \\ * & * \end{bmatrix} \begin{bmatrix} z^{-n} & \\ & 1 \end{bmatrix}.
\end{aligned}$$

By identifying $b(z)$ with $f(z)$, we have shown that the GQSP protocol and NLFT is equivalent. On the other hand, by setting $F_k = \mathrm{i}\tan\psi_k$ and identifying $x = \cos\theta$, $z = \mathrm{e}^{\mathrm{i}2\theta}$, and $b(z) = \mathrm{Im}\{P(x)\}$, thus showing that QSP and NLFT are equivalent by the equality $HV_\Phi(x)H = \mathcal{F}(z)\mathrm{e}^{\mathrm{i}n\theta Z}$. The matrix H is the Hadamard matrix. Henceforth, when faced with a QSP or a GQSP protocol and we are asked to find its phase, we can swiftly transform the problem into solving the inverse NLFT.

2.2. Properties of NLFT

These first few basic properties, like Theorem 2.1 and Theorem 2.2, greatly simplify the notations used. Next, up to Lemma 2.2, the existence and uniqueness to an outer complementary polynomial is established, these are essential to the stability of our algorithms. And lastly, Theorem 2.5 and Theorem 2.6 set the scene for the upcoming RHW algorithm.

Theorem 2.1. *If the Fourier coefficients $F = (\dots, 0, F_m, F_{m+1}, \dots, F_n)$, then for $\mathcal{F}(z) = (a(z), b(z))$, we have a and b both being Laurent polynomials:*

$$a(z) = a_{m-n} z^{m-n} + a_{m-n+1} z^{m-n+1} + \cdots + a_0 \text{ and } b(z) = b_m z^m + b_{m+1} z^{m+1} + \cdots + b_n z^n. \tag{10}$$

Theorem 2.2 (Shifting Property). *Consider the sequence $F = \{F_k\}$ having the NLFT $\mathcal{F} = (a, b)$. Then for the sequence $G = \{G_k\}$ satisfying $G_k = F_{k-1}$, then its NLFT is*

$$\mathcal{G}(z) = \big(a(z), zb(z)\big). \tag{11}$$

The two theorems above can be proven by simple induction. These two theorems showed that, without loss of generality, we can set F to be fully supported on $[0 : n]$ from now on. Any other cases can be easily related by a shift. As a consequence, both $a^*(z)$ and $b(z)$ are degree-n polynomials in z.

Theorem 2.3. *For $\mathcal{F} = (a, b)$, the Laurent polynomials a and b satisfy*

$$a(z)a^*(z) + b(z)b^*(z) = 1 \tag{12}$$

for all $z \in \mathbb{C} \cup \{\infty\}$.

Proof. Since for all $z \in \mathbb{T}$, $\mathcal{F}$ is an SU(2) matrix, we have

$$a(z)a^*(z) + b(z)b^*(z) = \left|a(z)\right|^2 + \left|b(z)\right|^2 = 1.$$

By uniquely expanding the $a(z)a^*(z)+b(z)b^*(z)$ via the Fourier series over $\mathbb{T}$, we know it to be identical to 1. Hence, by analytic continuation, it must be identical to 1 over the whole $\mathbb{C} \cup \{\infty\}$. ■

Next, note that

$$a^*(0) = \prod_{k=0}^{n} \frac{1}{\sqrt{1 + |F_k|^2}} > 0, \tag{13}$$

we can finally state the admissible pairs of functions a and b such that it is the NLFT of a sequence.

Lemma 2.1 (NLFT Bijection, Lemma 2.2 [NSYL25]). *The NLFT is a bijection from $\ell^0(\mathbb{Z}, \mathbb{C})$ onto the space*

$$\mathbf{S} := \left\{ (a, b) \,\middle|\, a, b \text{ are Laurent polynomials, } aa^* + bb^* = 1, \text{ and } 0 < a^*(0) < \infty \right\}. \tag{14}$$

Definition 2.5. *For a parameter $\eta \in (0, 1)$, we define the decreasing family of subsets $\mathbf{S}_\eta \subseteq \mathbf{S}$ as*

$$\mathbf{S}_\eta := \left\{ (a, b) \in \mathbf{S} \,\middle|\, b \text{ is a polynomial and } 0 < \sup_{z\in\mathbb{T}} \left|b(z)\right| \le 1 - \eta \right\}. \tag{15}$$

As we have mentioned in the introduction, when $\sup\left|b(z)\right|$ approaches one, the problem becomes coherent and the algorithms become more and more unstable. It is further important to know the following:

Theorem 2.4. *If $(a, b) \in \mathbf{S}_\eta$ and $a^*(z)$ is outer, then it must be the case that $a^*(z)$ has no zeros over $\overline{\mathbb{D}}$.*

Proof. By the outerness assumption, $a^*(z)$ has no zeros in $\mathbb{D}$, what remains is to show it does not equal to 0 on $\mathbb{T}$. If this is not the case, then $|a|^2 + |b|^2 \le 1 - \eta$ for some $z \in \mathbb{T}$, which is a contradiction of that fact that (a, b) is unitary. Thus it is shown. ■

As a corollary: given b, there are a lot of different a's such that $(a, b) \in \mathbf{S}$; however, there should exists a *unique* a^* that is outer such that $(a, b) \in \mathbf{S}$. Finding this unique outer a^* requires the use of the Weiss algorithm, which is discussed in Section 3.1. When does such a^* exist?

Lemma 2.2 (Szegő Condition, Theorem 4 [ALMTW24]). *Consider a measurable function b on $\mathbb{T}$ with $\|b\|_\infty < 1$. If b satisfies the Szegő condition*

$$\int_{\mathbb{T}} \log\left(1 - \left|b(z)\right|^2\right) \mathrm{d}z > -\infty, \tag{16}$$

then there exists a unique measurable outer function a^* on $\mathbb{T}$ such that $(a, b) \in \mathbf{S}$.

This condition is natural, as it implies that the subset of $\mathbb{T}$ where $|b| = 1$ has a measure of 0, implying that a^* is almost surely not equal to 0 over $\mathbb{T}$, which in turn has severe consequences on the stability of the inverse NLFT algorithms.

After discussing the existence and uniqueness to the outer complementary polynomial $a^*(z)$, the upcoming theorems set the stage and motivation for the RHW algorithm.

Definition 2.6 (Riemann–Hilbert Factorization). *We can decompose the sequence $F = (F_0, F_1, \ldots, F_n)$ into two disjoint sequences $F_{<k} = (F_0, \ldots, F_{k-1}, 0, \ldots)$ and $F_{\geq k} = (0, \ldots, 0, F_k, \ldots, F_n)$. Then, the corresponding NLFT will have the factorization*

$$\underbrace{\begin{bmatrix} a(z) & b(z) \\ -b^*(z) & a^*(z) \end{bmatrix}}_{\mathcal{F}} = \underbrace{\begin{bmatrix} a_{<k}(z) & b_{<k}(z) \\ -b^*_{<k}(z) & a^*_{<k}(z) \end{bmatrix}}_{\mathcal{F}_{<k}} \underbrace{\begin{bmatrix} a_{\geq k}(z) & b_{\geq k}(z) \\ -b^*_{\geq k}(z) & a^*_{\geq k}(z) \end{bmatrix}}_{\mathcal{F}_{\geq k}}. \tag{17}$$

This is the Riemann–Hilbert factorization of $\mathcal{F}$ at the index k. The resulting polynomials are parameterized by

$$a^*_{\geq k}(z) = a_{k,0} + a_{k,1}z + \cdots + a_{k,n-k}z^{n-k}, \tag{18}$$

$$b_{\geq k}(z) = z^k \left(b_{k,0} + b_{k,1}z + \cdots + b_{k,n-k}z^{n-k} \right). \tag{19}$$

As one can see, $a^*(z)$ will be used more often since it is a polynomial while $a(z)$ is not. How does one then solve for the factorization given $\mathcal{F}$ and a particular k? The following theorem answers this question:

Theorem 2.5 (Riemann–Hilbert Factorization). *Let $P_{\mathbb{D}}$ be the projection of a Laurent polynomial onto its non-negative powers, and $P_{\mathbb{D}^*}$ the projection onto the non-positive powers. The Riemann–Hilbert factorization problem can be solved effectively by the following system of linear equations:*

$$\begin{bmatrix} 1 & P_{\mathbb{D}^*} \frac{b^*}{a^*} \\ -z^k P_{\mathbb{D}} z^{-k} \frac{b}{a} & 1 \end{bmatrix} \begin{bmatrix} a_{\geq k} \\ b_{\geq k} \end{bmatrix} \propto \begin{bmatrix} 1 \\ 0 \end{bmatrix}. \tag{20}$$

Proof. The theorem above can be proven by observing the degrees to the polynomials. Since

$$(a_{<k}, b_{<k}) = (a, b) \left(a_{\geq k}, b_{\geq k}\right)^{-1} = (a, b) \left(a^*_{\geq k}, -b_{\geq k}\right) = \left(aa^*_{\geq k} + bb^*_{\geq k}, -ab_{\geq k} + a_{\geq k}b\right),$$

we have

$$\begin{cases} b_{<k} = -ab_{\geq k} + a_{\geq k}b \\ a_{<k} = aa^*_{\geq k} + bb^*_{\geq k} \end{cases} \Rightarrow \begin{cases} b_{\geq k} = \dfrac{ba_{\geq k}}{a} - \dfrac{b_{<k}}{a}, \\ a^*_{\geq k} = \dfrac{a_{<k}}{a} - \dfrac{bb^*_{\geq k}}{a} \end{cases}$$

Firstly, since $b_{\geq k}$ is a polynomial with the minimum degree greater than or equal to k,

$$b_{\geq k} = z^k P_{\mathbb{D}} \left(z^{-k} b_{\geq k}\right) = z^k P_{\mathbb{D}} \left(z^{-k} \frac{ba_{\geq k}}{a} - z^{-k} \frac{b_{<k}}{a}\right).$$

The term $z^{-k} b_{<k}/a$ has a Laurent series expansion analytic in $\mathbb{D}^*$ with constant term being zero, hence it vanishes under $P_{\mathbb{D}}$:

$$b_{\geq k} = z^k P_{\mathbb{D}} \left(z^{-k} \frac{b}{a} a_{\geq k}\right).$$

Secondly, $a^*_{\geq k}$ is a polynomial in z, while $a_{<k}/a$ has a Laurent series expansion analytic in $\mathbb{D}^*$, only its constant term survives under the projector $P_{\mathbb{D}}$:

$$a^*_{\geq k} = P_{\mathbb{D}} \left(a^*_{\geq k}\right) = P_{\mathbb{D}} \left(\frac{a_{<k}}{a} - \frac{bb^*_{\geq k}}{a}\right) = \frac{a_{<k}(\infty)}{a(\infty)} - P_{\mathbb{D}} \left(\frac{b}{a} b^*_{\geq k}\right).$$

Then, by Equation 13, the constant term is equal to

$$\frac{a_{<k}(\infty)}{a(\infty)} = \prod_{m=k}^{n} \sqrt{1 + |F_m|^2} = \left(a_{\geq k}(\infty)\right)^{-1}.$$

Finally, by taking the conjugate to the equation, we have

$$a_{\geq k} = \frac{1}{a^*_{\geq k}(0)} - P_{\mathbb{D}^*} \left(\frac{b^*}{a^*} b_{\geq k}\right).$$

Thus, the theorem is proven, and the constant of proportionality is identified to be $a^*_{\geq k}(0)$. ■

Being able to solve for the Riemann–Hilbert factorization proves to be really useful as it allows for parallel computation of the Fourier coefficients by incorporating the theorem below.

Theorem 2.6 (Layer-Stripping). *The kth Fourier coefficient can be calculated as*

$$F_k = \left.\frac{b_{\geq k}(z)z^{-k}}{a^*_{\geq k}(z)}\right|_{z=0} = \frac{b_{k,0}}{a_{k,0}}. \tag{21}$$

Proof. Note that

$$\begin{aligned}\left(a_{\geq k}, b_{\geq k}\right) &= \frac{1}{\sqrt{1+|F_k|^2}}\left(1, F_k z^k\right)\left(a_{\geq k+1}, b_{\geq k+1}\right)\\ \Rightarrow \left(a_{\geq k+1}, b_{\geq k+1}\right) &\propto \left(1, -F_k z^k\right)\left(a_{\geq k}, b_{\geq k}\right) = \left(a_{\geq k} + F_k z^k b^*_{\geq k}, b_{\geq k} - F_k z^k a^*_{\geq k}\right).\end{aligned}$$

Further note that the minimum degree to $b_{\geq k+1}$ is $k+1$, hence $z^{-k}b_{\geq k+1}(z)\big|_{z=0} = 0$, then

$$\begin{aligned} b_{\geq k+1}(z) &\propto b_{\geq k}(z) - F_k z^k a^*_{\geq k}(z)\\ 0 = z^{-k}b_{\geq k+1}(z)\Big|_{z=0} &= z^{-k}b_{\geq k}(z) - F_k a^*_{\geq k}(z)\Big|_{z=0}\\ \Rightarrow F_k &= \left.\frac{b_{\geq k}(z)z^{-k}}{a^*_{\geq k}(z)}\right|_{z=0} = \frac{b_{k,0}}{a_{k,0}}.\end{aligned}$$

The parameters $b_{k,0}$ and $a_{k,0}$ follows from Definition 2.6. Thus it is proven. ■

2.3. Layer-Stripping Method

As a starter, let us see how Theorem 2.6 is applied to find the Fourier coefficients in the "layer-stripping" method.

Given that we have a and b, we can immediately apply Theorem 2.6 to obtain F_0 as

$$F_0 = \frac{b(0)}{a^*(0)} = \frac{b(0)}{\overline{a(\infty)}} = \frac{b_{0,0}}{a_{0,0}}.$$

Next, since

$$\begin{aligned}\left(a_{\geq k}, b_{\geq k}\right) &= \frac{1}{\sqrt{1+|F_k|^2}}\left(1, F_k z^k\right)\left(a_{\geq k+1}, b_{\geq k+1}\right),\\ \Rightarrow \left(a_{\geq k+1}, b_{\geq k+1}\right) &= \frac{1}{\sqrt{1+|F_k|^2}}\left(1, -F_k z^k\right)\left(a_{\geq k}, b_{\geq k}\right),\end{aligned}$$

we can obtain $a^*_{\geq 1}$ and $b_{\geq 1}$ simply by the recursive relations

$$a^*_{\geq k+1} = \frac{1}{\sqrt{1+|F_k|^2}}\left(a^*_{\geq k} + \overline{F}_k z^{-k} b_{\geq k}\right), \tag{22}$$

$$z^{-(k+1)}b_{\geq k+1} = \frac{1}{z\sqrt{1+|F_k|^2}}\left(-F_k a^*_{\geq k} + z^{-k}b_{\geq k}\right). \tag{23}$$

With $a^*_{\geq 1}$ and $b_{\geq 1}$, we can obtain F_1. Continue stripping each F_k by the same fashion, we can successfully obtain all Fourier coefficients. This is the layer-stripping method proposed way back in 2005 by Tsai [T05]. Note that, though being equivalent, the layer-stripping proposed by Gilyén et al. strips from F_n down to F_0, whereas the one we just described strips from F_0 up to F_n.

3. INVERSE NONLINEAR FOURIER TRANSFORM

In this section, we go over the algorithms for the inverse nonlinear Fourier transform. These includes the Weiss algorithm for constructing an outer complementary polynomial, the RHW algorithm for finding each Fourier coefficients in parallel with a time complexity of $O(n^4)$, the half-Cholesky method which speeds up the RHW algorithm to $O(n^2)$, and the iNLFFT method having a time complexity of $O(n\log^2 n)$. This last method links up the half-Cholesky method with the layer-stripping method and is the state-of-the-art.

3.1. Weiss Algorithm

The Weiss algorithm generates a unique outer complementary polynomial. It is the core to the stability of the algorithms. The following definition to the outerness is equivalent to that of Definition 2.2, but it is more useful in aiding our construction of an outer function.

Definition 3.1 (Outer Function). *A function $g \in L^\infty(\mathbb{T})$ is classified as outer if $\log|g| \in L^1(\mathbb{T})$, and can be expressed in the form $g = e^G$, where $G = \log|g| + \mathrm{i}\mathcal{H}(\log|g|)$. Here, $\mathcal{H}$ represents the Hilbert transform, an operator defined such that $\mathcal{H}(c) = 0$ for a constant c, $\mathcal{H}(z^n) = -\mathrm{i}z^n$, and $\mathcal{H}(z^{-n}) = \mathrm{i}z^{-n}$.*

Proof. Let us show the equivalence between Definition 2.2 and Definition 3.1.

Let $f \in H^p(\mathbb{D})$ be a Hardy function. Then f is analytic in $\mathbb{D}$, has no zeros, and satisfies $\log|f(e^{\mathrm{i}t})| \in L^1(\mathbb{T})$. Hence, it automatically satisfies Definition 2.2.

Since $\log|f(z)|$ is a harmonic function on the unit disk (as the real part of $\log f(z)$), it admits the Poisson integral representation:

$$\log|f(z)| = \frac{1}{2\pi}\int_0^{2\pi} \frac{1-|z|^2}{|e^{\mathrm{i}t}-z|^2}\log|f(e^{\mathrm{i}t})|\,\mathrm{d}t.$$

This formula expresses a harmonic function in the disk in terms of its boundary values using the Poisson kernel. It applies whenever the boundary function is integrable, which is satisfied here because $\log|f(e^{\mathrm{i}t})| \in L^1(\mathbb{T})$.

Furthermore, the harmonic conjugate $\Theta(z)$ of $\log|f(z)|$ is also harmonic in $\mathbb{D}$, and its boundary values are given by the Hilbert transform:

$$\Theta(e^{\mathrm{i}\theta}) = \mathcal{H}(\log|f|)(e^{\mathrm{i}\theta}) = \frac{1}{2\pi}\text{p.v.}\int_0^{2\pi}\cot\left(\frac{\theta-t}{2}\right)\log|f(e^{\mathrm{i}t})|\,\mathrm{d}t.$$

As a result, $\log f(z) = \log|f(z)| + \mathrm{i}\Theta(z)$ is analytic, and the boundary values of $\log f$ satisfy:

$$\log f(e^{\mathrm{i}\theta}) = \log|f(e^{\mathrm{i}\theta})| + \mathrm{i}\mathcal{H}(\log|f|)(e^{\mathrm{i}\theta}),$$

so

$$f(e^{\mathrm{i}\theta}) = \exp\left(\log|f(e^{\mathrm{i}\theta})| + \mathrm{i}\mathcal{H}(\log|f|)(e^{\mathrm{i}\theta})\right).$$

Letting $g = f|_{\mathbb{T}}$, this shows $g = e^G$ with $G = \log|g| + \mathrm{i}\mathcal{H}(\log|g|)$. Thus, g satisfies Definition 3.1.

Suppose $g = \exp(\log|g| + \mathrm{i}\mathcal{H}(\log|g|)) \in L^\infty(\mathbb{T})$. Define

$$f(z) = \exp\left(\frac{1}{2\pi}\int_0^{2\pi}\frac{e^{\mathrm{i}t}+z}{e^{\mathrm{i}t}-z}\log|g(e^{\mathrm{i}t})|\,\mathrm{d}t\right).$$

This defines an analytic function in the disk because the integrand is analytic in z for $|z| < 1$, and the exponential of the integral remains non-zero, so $f(z) \neq 0$ in $\mathbb{D}$.

The real part of the integrand

$$\mathrm{Re}\left\{\frac{e^{\mathrm{i}t}+z}{e^{\mathrm{i}t}-z}\right\} = \frac{1-|z|^2}{|e^{\mathrm{i}t}-z|^2}$$

is the Poisson kernel, which ensures that

$$\log|f(z)| = \frac{1}{2\pi}\int_0^{2\pi}\frac{1-|z|^2}{|e^{\mathrm{i}t}-z|^2}\log|g(e^{\mathrm{i}t})|\,\mathrm{d}t.$$

Thus, the real part of $\log f(z)$ is harmonic and equal to the Poisson integral of $\log|g|$, and hence

$$|f(e^{\mathrm{i}\theta})| = \exp\left(\log|g(e^{\mathrm{i}\theta})|\right) = |g(e^{\mathrm{i}\theta})|.$$

The imaginary part of $\log f(z)$ is the harmonic conjugate of $\log|f(z)|$, and its boundary value is given by the Hilbert transform of $\log|g|$. Thus,

$$\arg f(e^{\mathrm{i}\theta}) = \mathcal{H}(\log|g|)(e^{\mathrm{i}\theta}).$$

Combining magnitude and phase, we have:

$$f(e^{\mathrm{i}\theta}) = \exp(\log|g(e^{\mathrm{i}\theta})| + \mathrm{i}\mathcal{H}(\log|g|)(e^{\mathrm{i}\theta})) = g(e^{\mathrm{i}\theta}). \tag{24}$$

Therefore, $f \in H^p(\mathbb{D})$, is analytic, has no zeros, and agrees with g on the boundary. So f is outer by Definition 2.2. ∎

With this definition to the outer function, the Weiss algorithm can easily produce an outer complementary polynomial a^* given b by the following procedure:

(i) **Compute** $R(z)$: For a given pure imaginary Laurent polynomial $b(z)$ that satisfies $||b||_\infty \leq 1 - \eta < 1$, the function $R(z)$ is defined on the unit circle $\mathbb{T}$ as $R(z) := \log \sqrt{1 - |b(z)|^2}$. This $R(z)$ is real-valued and belongs to $L^1(\mathbb{T})$ due to the Szegő condition being satisfied.

(ii) **Compute the FFT of** $R(z)$: To obtain the Fourier coefficients of $R(z)$, we select a discretization parameter N, which is the smallest power of 2 satisfying a given condition (e.g., Equation. 53 or 101, [ALMTW24]). We then apply the Fast Fourier Transform (FFT) to evaluate $R(z)$ at the Nth roots of unity $\{z_l\}_{l=0}^{N-1}$. The rearranged results of this FFT provide the approximate Fourier coefficients $\hat{r}_j$ of $R(z)$, denoted as $\hat{R} = (\hat{r}_{-\lfloor N/2 \rfloor}, \ldots, \hat{r}_0, \ldots, \hat{r}_{N-1-\lfloor N/2 \rfloor})$. The value of N is of the order

$$N = \Omega\left(\frac{n}{\eta} \log \frac{n^2}{\eta^4 \varepsilon}\right), \tag{25}$$

where ε is the intended maximum error in the phase factors.

(iii) **Compute** $\hat{G}(z)$: The approximate function $\hat{G}(z)$ is constructed from these Fourier coefficients to realize the Hilbert transform. The following properties of the Hilbert transform provide the foundation for computing $\hat{G}(z)$:

- For positive frequencies: $z^n - \mathrm{i}\mathcal{H}(z^n) = 0$, for $n \in \mathbb{N}^+$. This means that positive frequency components should be discarded.
- For negative frequencies: $z^{-n} - \mathrm{i}\mathcal{H}(z^{-n}) = 2z^{-n}$, for $n \in \mathbb{N}^+$. This means that negative frequency components should be doubled.
- For constants: $c - \mathrm{i}\mathcal{H}(c) = c$, for $c \in \mathbb{C}$. This means that the constant (zero-frequency) component should remain unchanged.

Based on these properties, $\hat{G}(z) := \hat{R}(z) - \mathrm{i}\mathcal{H}(\hat{R}(z))$ is explicitly given by:

$$\hat{G}(z) = 2\sum_{l=1}^{N/2} \hat{r}_{-l} z^{-l} + \hat{r}_0.$$

(iv) **Compute** $\hat{a}(z) = e^{\hat{G}(z)}$

Notice that the FFT algorithm gives $O(N \log N) = O(n \,\mathrm{polylog}(n/\varepsilon))$ time complexity.

It should be noted that since in application, we are working with target functions f subject to $\|f\|_\infty = 1 - \eta$ where $\eta \in (0, 1)$, combined with the fact that a^* is outer, we have that a^* has no zeros over $\overline{\mathbb{D}}$.

3.2. Riemann–Hilbert–Weiss Algorithm

Next, we state the details to the RHW algorithm. Though in terms of time complexity, it is not the state-of-the-art, but its derivation closely mirrors parts of the derivation of iNLFFT. Further, the details to the derivation proves to be very useful in analyzing the stability of the algorithms.

From its name, one already knows that it incorporates the Weiss algorithm and the Riemann–Hilbert factorization. From the Weiss algorithm, we have obtained the series expansion to b/a, hence, we can obtain the elements to the matrix in Equation 20. Writing it out explicitly: let $c(z) := b(z)/a(z) = \sum_{k=-\infty}^{n} c_k z^k$, we can rewrite the system of equations as

$$\begin{bmatrix} \mathbb{1}_{n-k+1} & -T_k^\dagger \\ T_k & \mathbb{1}_{n-k+1} \end{bmatrix} \begin{bmatrix} \boldsymbol{b}_k \\ \mathrm{rev}(\boldsymbol{a}_k^*) \end{bmatrix} \propto \mathrm{rev}(\boldsymbol{e}_0), \tag{26}$$

with vectors $\boldsymbol{e}_0 = (1, 0, \ldots, 0)^\mathsf{T}$, $\boldsymbol{a}_k^* = (a_{k,0}^*, \ldots, a_{k,n-k}^*)^\mathsf{T}$, $\boldsymbol{b}_k = (b_{k,0}, \ldots, b_{k,n-k})^\mathsf{T}$, the operator $\mathrm{rev}(\cdot)$ reversing the order of the vector, and the Toeplitz matrix

$$T_k = \begin{bmatrix} c_n^* & & & \\ \vdots & \ddots & & \\ c_{k+1}^* & & \ddots & \\ c_k^* & c_{k+1}^* & \cdots & c_n^* \end{bmatrix}. \tag{27}$$

Since $F_k = b_{k,0}/a_{k,0}$, we only care about their ratio, and the proportionality constant can be set to 1. With the coefficients computed, we can now obtain each Fourier coefficient in parallel by the linear system above! Note that inverting a general matrix requires a time complexity of $O(n^3)$, the total complexity for computing all Fourier coefficients will then be $O(n^4)$. Costly!

3.3. Half-Cholesky Method

If we let go of the requirement of the computation being parallel, we can combine all n systems of linear equations into one. Let $T := T_0$, then we have

$$\underbrace{\left[\begin{array}{c|c} \mathbb{1}_{n+1} & -T^\dagger \\ \hline T & \mathbb{1}_{n+1} \end{array}\right]}_{A} \underbrace{\left[\begin{array}{c|cccc} & b_{n,0} & b_{n-1,0} & \cdots & b_{0,0} \\ \mathbb{1}_{n+1} & & b_{n-1,1} & & \vdots \\ & & & \ddots & b_{0,n} \\ \hline & a^*_{n,0} & a^*_{n-1,1} & \cdots & a^*_{0,n} \\ 0 & & a^*_{n-1,0} & & \vdots \\ & & & \ddots & a^*_{0,0} \end{array}\right]}_{S} = \underbrace{\left[\begin{array}{c|ccc} \mathbb{1}_{n+1} & & 0 & \\ \hline & 1 & & \\ T & \vdots & \ddots & \\ & * & \cdots & 1 \end{array}\right]}_{L}. \tag{28}$$

Note that L is a lower-triangular matrix with diagonals being all 1's and S is an upper-triangular matrix, this is the LU decomposition of $A = LS^{-1}$. Performing the LU decomposition once brings the time complexity down to $O(n^3)$.

But we can do much better! Note that the LU decomposition of A can be written as

$$\begin{bmatrix} \mathbb{1} & -T^\dagger \\ T & \mathbb{1} \end{bmatrix} = \begin{bmatrix} \mathbb{1} & \\ T & \mathbb{1} \end{bmatrix} \begin{bmatrix} \mathbb{1} & \\ & \mathbb{1} + TT^\dagger \end{bmatrix} \begin{bmatrix} \mathbb{1} & -T^\dagger \\ & \mathbb{1} \end{bmatrix}. \tag{29}$$

If we can obtain the Cholesky decomposition of $\mathbb{1} + TT^\dagger = LDL^\dagger$, then

$$A = \begin{bmatrix} \mathbb{1} & \\ T & L \end{bmatrix} \cdot \begin{bmatrix} \mathbb{1} & \\ & DL^\dagger \end{bmatrix} \begin{bmatrix} \mathbb{1} & -T^\dagger \\ & \mathbb{1} \end{bmatrix} \Rightarrow S = \begin{bmatrix} \mathbb{1} & T^\dagger (DL^\dagger)^{-1} \\ 0 & (DL^\dagger)^{-1} \end{bmatrix}. \tag{30}$$

The general time complexity required for Cholesky decomposition is, however, still $O(n^3)$. But notice that $\mathbb{1} + TT^\dagger$ is a Toeplitz matrix, and it can be fully specified by $2n$ elements. Hence, we should expect the Cholesky decomposition to be faster. And it indeed can be faster, but we require the notion of the displacement rank.

Definition 3.2. *Let the matrix Z be the lower-shifting matrix where $Z_{ij} = 1$ if $i = j+1$ and 0 otherwise. The displacement of a matrix K is defined to be*

$$\Delta K := K - ZKZ^\mathsf{T}. \tag{31}$$

If ΔK is a rank-r matrix, then we say that K has a displacement rank of r.

Lemma 3.1 ([AG89]). *If the displacement rank of an $n \times n$ matrix K is r, then the Cholesky decomposition of K has a time complexity of $O(n^2 r)$.*

Let us see this is indeed the case for $K = \mathbb{1} + TT^\dagger$. Let us denote

$$T = \begin{bmatrix} c^*_n & & & \\ \vdots & \ddots & & \\ c^*_1 & & \ddots & \\ c^*_0 & c^*_1 & \cdots & c^*_n \end{bmatrix} =: \begin{bmatrix} \boldsymbol{c}_0 & \boldsymbol{c}_1 & \dots & \boldsymbol{c}_n \end{bmatrix},$$

then

$$TT^\dagger = \boldsymbol{c}_0 \boldsymbol{c}_0^\dagger + \boldsymbol{c}_1 \boldsymbol{c}_1^\dagger + \cdots + \boldsymbol{c}_n \boldsymbol{c}_n^\dagger.$$

The displacement to K is calculated to be

$$\Delta K = \Delta(\mathbb{1}) + \sum_{k=0}^{n} \Delta\left(\boldsymbol{c}_k \boldsymbol{c}_k^\dagger\right) = \boldsymbol{e}_0 \boldsymbol{e}_0^\dagger + \sum_{k=0}^{n-1} \left(\boldsymbol{c}_k \boldsymbol{c}_k^\dagger - \boldsymbol{c}_{k+1} \boldsymbol{c}_{k+1}^\dagger\right) + \boldsymbol{c}_n \boldsymbol{c}_n^\dagger = \boldsymbol{e}_0 \boldsymbol{e}_0^\dagger + \boldsymbol{c}_0 \boldsymbol{c}_0^\dagger.$$

We have thus shown that the displacement rank to K is 2, and its Cholesky decomposition requires only $O(n^2)$ steps of computation. We will not go into the details of how the Cholesky decomposition is actually performed. However, let us hint at a detail that will shown to be reminiscent later on: if we define

$$G_0 = \begin{bmatrix} \boldsymbol{e}_0 & \boldsymbol{c}_0 \end{bmatrix}, \text{ then } \Delta K = G_0 G_0^\dagger.$$

We can obtain the Fourier coefficient F_0 by performing QR decomposition on G_0, say

$$G_0 = R_0 Q_0 = \begin{bmatrix} s_0 & 0 \\ \boldsymbol{u}_0 & \boldsymbol{v}_0 \end{bmatrix} Q_0.$$

Then, the upper-left element of G_0 will be $(1 + |F_0|^2)^{-1/2}$ and the upper-right element will be $F_0 \cdot (1 + |F_0|^2)^{-1/2}$. Furthermore, we can define $G_1 = [Z\boldsymbol{u}_0, \boldsymbol{v}_0]$, applying the QR decomposition on G_1 gives us F_1, and so on.

After successfully computing the Cholesky decomposition, notice that only the colored terms in Equation 28 is what we need. We have

$$\begin{bmatrix} a_{n,0} \\ \vdots \\ a_{0,0} \end{bmatrix} = D^{-1} \begin{bmatrix} 1 \\ \vdots \\ 1 \end{bmatrix} \text{ and } \begin{bmatrix} b_{n,0} \\ \vdots \\ b_{0,0} \end{bmatrix} = D^{-1} L^{-*} \boldsymbol{c}_0, \text{ with } \boldsymbol{c}_0 = \begin{bmatrix} c_n \\ \vdots \\ c_0 \end{bmatrix}, \tag{32}$$

as $\mathbb{1} + TT^\dagger \succeq 0$, and D is real. The Fourier coefficients is computed to be

$$\begin{bmatrix} F_n \\ \vdots \\ F_0 \end{bmatrix} = \begin{bmatrix} b_{n,0}/a_{n,0} \\ \vdots \\ b_{0,0}/a_{0,0} \end{bmatrix} = L^{-*} \boldsymbol{c}_0. \tag{33}$$

The whole task requires only a time complexity of $O(n^2)$.

3.4. Layer-Stripping Revisited

Let us revisit Section 2.3 and see how a simple modification to how we write the equations changes everything. Note that previously in Section 2.3 we introduced the recursive relations

$$a^*_{\geq k+1} = \frac{1}{\sqrt{1 + |F_k|^2}} \left(a^*_{\geq k} + \overline{F}_k z^{-k} b_{\geq k} \right), \tag{34}$$

$$z^{-(k+1)} b_{\geq k+1} = \frac{1}{z\sqrt{1 + |F_k|^2}} \left(-F_k a^*_{\geq k} + z^{-k} b_{\geq k} \right). \tag{35}$$

Through the parameterization $a^*_{\geq k} = \sum_{j=0}^{n-k} a_{k,j} z^j$ and $z^{-k} b_{\geq k} = \sum_{j=0}^{n-k} b_{k,j} z^j$, we have

$$\begin{bmatrix} a_{k+1,0} & 0 \\ a_{k+1,1} & b_{k+1,0} \\ \vdots & \vdots \\ a_{k+1,n-k-1} & b_{k+1,n-k-2} \\ 0 & b_{k+1,n-k-1} \end{bmatrix} = \underbrace{\begin{bmatrix} a_{k,0} & b_{k,0} \\ a_{k,1} & b_{k,1} \\ \vdots & \vdots \\ a_{k,n-k-1} & b_{k,n-k-1} \\ a_{k,n-k} & b_{k,n-k} \end{bmatrix}}_{G_k = [\boldsymbol{a}_k, \boldsymbol{b}_k]} \frac{1}{\sqrt{1 + |F_k|^2}} \begin{bmatrix} 1 & -F_k \\ \overline{F}_k & 1 \end{bmatrix} \tag{36}$$

The coefficient $F_k = b_{k,0}/a_{k,0}$ is obtained by the QR decomposition of G_k, requiring $O(n)$ complexity each. The total complexity will be $O(n^2)$. Notice how the notation of G_k here is similar to that introduced in the last subsection. In fact, they are indeed the same! In this section, F_k is obtained by the layer-stripping of (a, b), which is done via the QR decomposition of G_k written in terms of the series expansion to a^* and b. However, in the work by Ni et al. [NSYL25], they show through induction that for any $f(z)$, the sequence $(a(z)/f(z), b(z)/f(z))$ has the same layer-stripped Fourier coefficients. The layer-stripping method is closely related to the half-Cholesky method previously mentioned when setting $f(z) = a(z)$, then the half-Cholesky method is just applying layer-stripping on

$$\left(1, b(z)/a(z)\right) =: \left(1, c(z)\right).$$

And of course, the two methods have the same computational complexity.

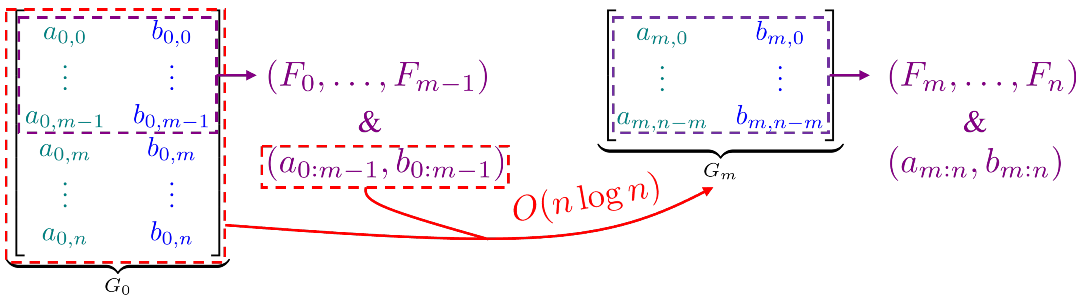

FIGURE 2: The divide-and-conquer scheme of iNLFFT.

3.5. Inverse Nonlinear Fast Fourier Transform

Starting from the reformulated layer-stripping method, let us devise a divide-and-conquer scheme that speeds up the computation of the Fourier coefficients.

First of all, notice that our task is simple:

(i) Given the sequence of coefficients $(a_{0,0}, a_{0,1}, \dots, a_{0,n})$ and $(b_{0,0}, b_{0,1}, \dots, b_{0,n})$.

(ii) Compute the sequence $(F_0, F_1, \dots, F_n)$ and the functions $(a_{0:n}, b_{0:n})$, which is the NLFT of the sequence $(F_0, F_1, \dots, F_n)$.

We can separate this task into:

(i) Given the sequences of coefficients $(a_{0,0}, a_{0,1}, \dots, a_{0,n})$ and $(b_{0,0}, b_{0,1}, \dots, b_{0,n})$, separate the sequences into four: $(a_{0,0}, a_{0,1}, \dots, a_{0,m-1})$, $(b_{0,0}, b_{0,1}, \dots, b_{0,m-1})$, $(a_{0,m}, a_{0,1}, \dots, a_{0,n})$, and $(b_{0,m}, b_{0,1}, \dots, b_{0,n})$. The value $m = \lceil n/2 \rceil$.

(ii) Use the first two sequences to compute the sequence $(F_0, F_1, \dots, F_{m-1})$ and the functions $(a_{0:m-1}, b_{0:m-1})$, which is the NLFT of the sequence $(F_0, F_1, \dots, F_{m-1})$.

(iii) Since
$$(a_{m:n}, b_{m:n}) = (a_{0:m-1}, b_{0:m-1})^{-1}(a_{0:n}, b_{0:n}) = (a^*_{0:m-1}, -b_{0:m-1})^{-1}(a_{0:n}, b_{0:n}),$$
we can use the functions $(a_{0:m-1}, b_{0:m-1})$ and $(a_{0:n}, b_{0:n})$ to obtain $(a_{m:n}, b_{m:n})$ via polynomial multiplication.

(iv) Given $(a_{m:n}, b_{m:n})$, which is equivalent to the sequences of coefficients $(a_{0,m}, a_{0,m+1}, \dots, a_{0,n})$ and $(b_{0,m}, b_{0,m+1}, \dots, b_{0,n})$, we can compute the sequence $(F_m, F_{m+1}, \dots, F_n)$ and the functions $(a_{m:n}, b_{m:n})$, which is the NLFT of the sequence $(F_m, F_{m+1}, \dots, F_n)$.

The above description is summarized in Figure 2.

Notice that step (ii) and step (iv) of the scheme above is exactly the same scheme mentioned at the start of Section 3.5. Hence, they each can be divided into two more sub-tasks. The division terminates when the sequence has length 1, then the computation of F_k is trivial. This is the inverse nonlinear fast Fourier transform. The computation complexity of this scheme originates from polynomial multiplication, which can be efficiently implemented via convolution theorem. It is known that the polynomial multiplication of two degree-n polynomials requires $O(n \log n)$ complexity. The total complexity for the iNLFFT method is then

$$n \log n + 2\left(\frac{n}{2} \log \frac{n}{2}\right) + 4\left(\frac{n}{4} \log \frac{n}{4}\right) + \cdots = O(n \log^2 n). \tag{37}$$

4. STABILITY ANALYSIS

In this section, we provide the essential theorems from the work by Ni et al. [NSYL25] that shows the stability of the algorithms given that the complementary polynomial a^* has no zeros over $\overline{\mathbb{D}}$. To make the readers have easy cross-reference, the notations in this section are changed so that they match those used in the work by Ni et al. Furthermore, when "Lemma" and "Theorem" are mentioned, they are also referencing that from the work by Ni et al. This section merely serves as a summary of the important points in the paper.

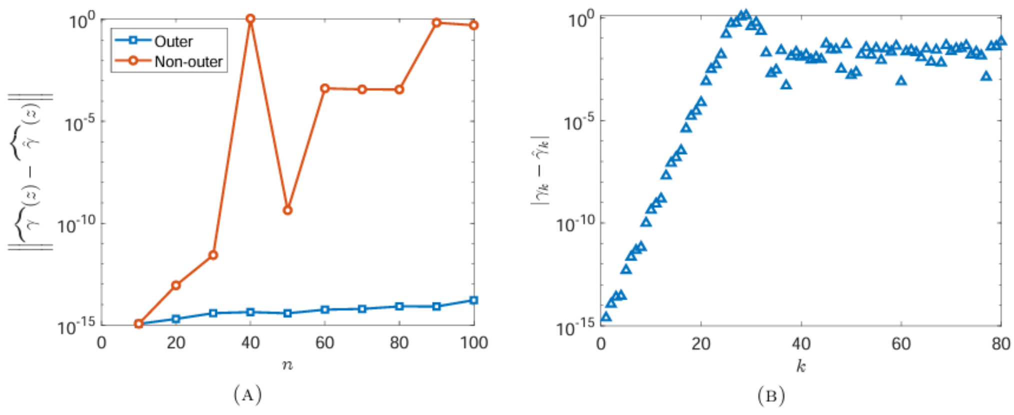

FIGURE 3: The instability without outerness condition, Figure 1 in [NSYL25]. (A) The residual $||\hat{\gamma} - \hat{\gamma}||_{L^\infty(\mathbb{T})}$ for different degree n. (B) For $n = 80$, the error for each component γ_k.

4.1. Importance of the Outerness Condition

The paper by Ni et al. constructs an example where stability fails without the outerness assumption. Let $b(z)$ be a randomly chosen degree $n-1$ polynomial with real coefficients such that $||b(z)||_{L^\infty(\mathbb{T})} \leq \frac{1}{2}$. Let $a_o(z)$ be its complementary Laurent polynomial with $a_o^*(z)$ outer. Define $a_{no}(z) = \omega z^{-(n-1)} a_o^*(z) = \omega z^{-(n-1)} a_o(z^{-1})$, where the sign $\omega \in \{\pm 1\}$ is chosen to ensure $a_{no}(\infty) > 0$. By definition, $a_{no}(z) a_{no}^*(z) = a_o(z) a_o^*(z)$, which implies that $(a_{no}, b) \in \mathcal{S}$ with $a_{no}^*(z)$ being non-outer.

Performing the layer stripping algorithm on (a_{no}, b) reveals significant instability (Figure 3). As the polynomial degree n increases, the difference between the NLFT of the computed Fourier coefficient sequence $\hat{\gamma}$ and the ground truth is $O(1)$. Note that in our previous sections, we used F in place of γ. For $n = 80$, the error in each component γ_k shows accumulation and exponential growth during the iterations, consistent with the worst-case upper bound in [H19]. This demonstrates that without the outerness condition on $a^*(z)$, the numerical stability of the layer stripping algorithm is not guaranteed due to the potential accumulation of round-off errors, which can propagate and amplify exponentially as the computation proceeds.

4.2. Stability of the Layer Stripping Algorithm

To analyze the numerical stability of the layer stripping algorithm, we must examine the effects of round-off error in the recursive computation of the Fourier coefficient vector $\gamma = (\gamma_1, \ldots, \gamma_n)$ from a given pair (a_0, b_0). The key step is to formulate this computation as a structured linear system $L\gamma = b_0$, with $L = UH^{-1}$,

$$U = \begin{bmatrix} a_{0,1} & & & \\ a_{1,1} & a_{0,2} & & \\ \vdots & \vdots & \ddots & \\ a_{n-1,1} & a_{n-2,2} & \cdots & a_{0,n} \end{bmatrix},$$

and $H = \mathrm{diag}(a_{0,1}, \ldots, a_{0,n})$. See Equation 33. Let us first address the backward stability of this system. Lemma 5.3 establishes that the computed phase vector $\hat{\gamma}$ is in fact the exact solution to a perturbed linear system:

$$\left(\hat{U}\hat{H}^{-1} + E\right)\hat{\gamma} = \frac{1}{a_{0,0}} b_0,$$

where E is a perturbation matrix that satisfies an entry-wise bound of the form $|E_{ij}| \leq 30n\epsilon_{\mathrm{ma}} |(UH^{-1})_{ij}|$ provided $n\epsilon_{\mathrm{ma}} < 0.01$, where ϵ_{ma} is the machine epsilon. This is proven by induction: the base case for $n = 1$ is trivial, and in the inductive step, the error matrix E is constructed from the previous $n-1$ step using perturbations in $\hat{U}$ and $\hat{H}$. The matrix $M = \hat{U}\hat{H}^{-1}$ is partitioned into blocks to isolate the contribution of $\hat{\gamma}_0$, and by applying floating-point error bounds to each operation, a global matrix error bound is obtained. In norm, this yields

$$\|E\| \leq O\left(n^2 \epsilon_{\mathrm{ma}}\right) \|L\|.$$

This implies that the algorithm is backward stable: it returns the exact solution to a nearby problem.

To analyze forward error, we must estimate the condition number $\kappa(L) = \|L\| \cdot \|L^{-1}\|$. Lemma 5.4 provides this estimate under the crucial assumption that $a^*(z)$ is outer and that $(a_0, b_0) \in \mathcal{S}_\eta$. The outer property implies that $|a_k^*(z)| \geq \sqrt{\eta(2-\eta)}$ on the unit circle, and in particular, this ensures that the diagonal entries $a_{0,k}$ in H satisfy

$$|a_{0,k}| \geq \sqrt{\eta}, \quad \text{for all } k = 1, \ldots, n.$$

Since $H = \operatorname{diag}(a_{0,1}, \ldots, a_{0,n})$, we obtain

$$\|H^{-1}\| = \max_k |a_{0,k}^{-1}| \leq \eta^{-1/2},$$

and thus $\|L\| = \|UH^{-1}\| \leq \|U\| \cdot \|H^{-1}\| = O\left(\eta^{-1/2}\right)$. The inverse bound $\|L^{-1}\| = O\left(\eta^{-1}\right)$ is derived by constructing the inverse recursively via the structure of the layer stripping algorithm, and bounding the growth of entries from the recurrence relations. Hence,

$$\kappa(L) = \|L\| \cdot \|L^{-1}\| \leq O\left(\eta^{-3/2}\right).$$

Combining these two results, Theorem 5.5 concludes with a forward error estimate using the sensitivity of linear systems:

$$\frac{\|\hat{\gamma} - \gamma\|}{\|\gamma\|} \leq \kappa(L) \cdot \frac{\|E\|}{\|L\|} \leq O\left(n^2 \eta^{-3/2} \epsilon_{\mathrm{ma}}\right).$$

However, this estimate can be tightened. The full proof further refines the error by bounding the Frobenius norm $\|\hat{L} - L\|_F$ using the structure $L = UH^{-1}$ and its approximation $\hat{L} = \hat{U}\hat{H}^{-1}$. Since $K = LDL^*$ with $D = \operatorname{diag}(|a_{0,1}|^2, \ldots, |a_{0,n}|^2)$, and K is positive definite, one applies the LU perturbation theory to deduce that:

$$\frac{\|\hat{L} - L\|}{\|L\|} \leq O\left(n^3 \eta^{-3/2} \epsilon_{\mathrm{ma}}\right),$$

and thus the final forward error becomes

$$\frac{\|\hat{\gamma} - \gamma\|}{\|\gamma\|} \leq O\left(n^3 \eta^{-5/2} \epsilon_{\mathrm{ma}}\right).$$

This leads to the bit-precision requirement: to ensure error below ϵ, we must use at least $r = \log_2(n^3 \eta^{-5/2} \epsilon^{-1}) = O\left(\log(n\eta^{-1}\epsilon^{-1})\right)$ bits of precision.

4.3. Stability of the Inverse Nonlinear Fast Fourier Transform Algorithm

To establish the forward stability of the iNLFFT algorithm, Theorem 5.6 asserts that given a pair $(a_0, b_0) \in \mathcal{S}_\eta$ with outer complementary polynomial $a^*(z)$, the iNLFFT algorithm outputs a phase vector $\hat{\gamma}$ satisfying

$$\frac{\|\hat{\gamma} - \gamma\|}{\|\gamma\|} = O\left(\eta^{-1} h(n, \eta) \epsilon_{\mathrm{ma}}\right),$$

where $h(n, \eta) = \exp(O\left(\log^2 n + \log n \log \eta^{-1}\right))$ captures the quasi-polynomial growth of intermediate error propagation.

This conclusion depends critically on Lemma 5.7, which establishes the backward stability of the iNLFFT algorithm. It shows that there exists a perturbation matrix E and vector q such that the computed $\hat{\gamma}$ solves the perturbed system:

$$(L + E)\hat{\gamma} = \frac{1}{a_{0,0}}(b_0 + q),$$

where the norm bounds

$$\|E\| \leq h(n, \eta)\epsilon_{\mathrm{ma}}\|L\|, \qquad \|q\| \leq h(n, \eta)\epsilon_{\mathrm{ma}}\|b_0\|$$

hold provided $\epsilon_{\mathrm{ma}} \leq C_0^{-1} n^{-3} \eta^3 h(n, \eta)^{-1}$ for a universal constant C_0.

To prove this lemma, the authors proceed by induction on n, assuming $n = 2^t$. The base case $n = 1$ is trivial, and for the inductive step, they divide the problem into two halves. Let $m = n/2$ and write the input polynomials (a_0, b_0) as G_0, the $n \times 2$ matrix of coefficients. The midpoint polynomial pair (a_m, b_m) is denoted G_m. Let $\hat{G}_m$ be the floating-point approximation computed recursively.

The first key estimate is Lemma 5.8, which analyzes the propagation of error in the recursive computation of polynomials $\xi_n(z)$ and $\eta_n(z)$. Note that in our work, instead of ξ and η, we used $a^*_{m:n}$ and $b_{m:n}$ instead. Writing their coefficient vectors as ξ_n and η_n, and denoting their computed values as $\hat{\xi}_n$, $\hat{\eta}_n$, the lemma shows

$$\|\hat{\eta}_n - \eta_n\| \le g(n)\epsilon_{\mathrm{ma}}, \quad \|\hat{\xi}_n - \xi_n\| \le g(n)\epsilon_{\mathrm{ma}} \sum_{k=0}^{n-1} |\gamma_k|, \quad \|\hat{\xi}_n\| \le \sum_{k=0}^{n-1} |\gamma_k|,$$

where $g(n) \le 20C_4n^2$ for some constant C_4 derived from bounding recursive terms.

Building on this, Lemma 5.9 provides estimates on the midpoint error:

$$\|\hat{b}_m - b_m\| \le \left(C_5 m^{5/2}\eta^{-1} + C_6\sqrt{m}\eta^{-2}h(m)\right)\|b_0\|\epsilon_{\mathrm{ma}},$$

$$\|\hat{a}_m - a_m\| \le \left(C_5 m^{5/2}\eta^{-1} + C_6\sqrt{m}\eta^{-2}h(m)\right)\epsilon_{\mathrm{ma}},$$

with constants C_5, C_6 depending on matrix norm bounds. The Frobenius norm $\|\hat{G}_m - G_m\|_F$ is bounded by twice the above quantity, and is later used in estimating the difference between the Cholesky factors of structured matrices $K = GG^*$.

Lemma 5.10 then estimates the error introduced in the lower-half recursive solve due to errors in $\hat{G}_m$. Defining L_{down} as the lower $m \times m$ system from the recursive step, we have

$$\|\tilde{L}_{\mathrm{down}} - L_{\mathrm{down}}\| \le 4C_1 m\eta^{-3/2}\left(C_5 m^2\eta^{-1} + C_6\sqrt{m}\eta^{-2}h(m)\right)\epsilon_{\mathrm{ma}}\|L_{\mathrm{down}}\|.$$

Finally, the inductive recurrence of the backward error function $h(n)$ satisfies

$$h(n) = C\left(m^{3/2}\eta^{-4}h(m) + m^{7/2}\eta^{-3}\right),$$

which solves to

$$h(n) \le \exp\left(O\left(\log^2 n + \log n \log \eta^{-1}\right)\right),$$

confirming that the backward error grows quasi-polynomially.

Combining the backward stability of Lemma 5.7 and the condition number bound $\kappa(L) = O\left(\eta^{-1}\right)$ from Lemma 5.4, Theorem 5.6 applies the sensitivity inequality

$$\frac{\|\hat{\gamma} - \gamma\|}{\|\gamma\|} \le 2\|L^{-1}\|\left(\frac{\|q\|}{\|b_0\|} + \frac{\|E\|}{\|L\|}\right),$$

yielding

$$\frac{\|\hat{\gamma} - \gamma\|}{\|\gamma\|} = O\left(\eta^{-1}h(n,\eta)\epsilon_{\mathrm{ma}}\right),$$

provided the assumption $n^3\eta^{-3}h(n,\eta)\epsilon_{\mathrm{ma}} \le 1$ holds.

Therefore, to achieve accuracy ϵ, it suffices to use machine precision

$$\epsilon_{\mathrm{ma}} = \epsilon \cdot e^{-O\left(\log^2 n + \log n \log \eta^{-1}\right)},$$

or, equivalently, $r = \log_2 \epsilon_{\mathrm{ma}}^{-1} = O\left(\log^2 n + \log n \log \eta^{-1} + \log \epsilon^{-1}\right)$ bits of precision.

Thus, Theorem 5.6, supported by Lemmas 5.7 to 5.10, provides a complete quantitative stability analysis of the iNLFFT algorithm. The central insight is that the recursive divide-and-conquer structure of iNLFFT, although faster, introduces potential error amplification unless tightly controlled. The outer function assumption plays a critical role in bounding the conditioning of the problem, and is again indispensable for numerical robustness.

4.4. Role of Outerness in Stability Analysis

The outerness of the complementary polynomial $a^*(z)$ plays a central and indispensable role in the stability analysis of both the layer stripping and the iNLFFT algorithms.

In the layer stripping algorithm, the phase vector γ is recovered by solving a structured linear system $L\gamma = b_0$, where $L = UH^{-1}$ and $H = \mathrm{diag}(a_{0,1}, \ldots, a_{0,n})$ collects the leading coefficients of the polynomials $a^*_k(z)$. If $a^*(z)$ has zeros inside or on the unit circle, then some entries $a_{0,k}$ may become arbitrarily small, which causes the diagonal matrix H to be ill-conditioned or even singular. This leads to instability in computing $L = UH^{-1}$ and results in an unbounded condition number $\kappa(L)$. The outerness of $a^*(z)$ ensures that

$$\min_{|z|\le 1} |a^*(z)| \ge \delta > 0,$$

which implies that all diagonal entries of H are bounded below by $\sqrt{\eta}$ for some $\eta > 0$. Consequently, Lemma 5.4 shows that

$$\kappa(L) \le O\left(\eta^{-3/2}\right),$$

guaranteeing that the system $L\gamma = b_0$ is well-conditioned. This condition is essential in Theorem 5.5 to convert the backward error bound into a forward error bound and to ensure the algorithm's numerical stability.

In the iNLFFT algorithm, the role of outerness becomes even more intricate due to its divide-and-conquer recursive structure. In each recursive step, the midpoint matrix $G_m = (a_m, b_m)$ is computed from previous layers and is used to define a lower-dimensional system L_{down}. If $a^*(z)$ is not outer, then $a_m(z)$ may become small or degenerate, which causes G_m to be ill-conditioned and disrupts the stability of the lower-level recursive calls. Lemma 5.9 quantifies this by showing that the midpoint error grows proportionally to η^{-1} and η^{-2}:

$$\|\hat{a}_m - a_m\| \le \left(C_5 m^{5/2}\eta^{-1} + C_6\sqrt{m}\eta^{-2}h(m)\right)\epsilon_{\text{ma}}.$$

Similarly, Lemma 5.10 shows that the perturbation in the submatrix L_{down} is bounded in norm by

$$\|\tilde{L}_{\text{down}} - L_{\text{down}}\| \le O\left(m^{7/2}\eta^{-5/2}h(m)\epsilon_{\text{ma}}\right),$$

implying that outerness controls the error propagation rate through recursion. If $a^*(z)$ were not outer, then η could be arbitrarily small, causing the backward error function $h(n,\eta)$ to grow super-polynomially and the required bit precision to become impractically large.

In summary, the outerness of $a^*(z)$ is essential for numerical stability. It ensures that key matrices remain well-conditioned in both layer stripping and iNLFFT, and prevents uncontrolled error growth. Without this assumption, the inverse nonlinear Fourier transform becomes ill-posed and practically unsolvable.

5. CONCLUSION

5.1. Teamwork

The introduction is written by the two authors together. While Section 2, the RHW algorithm, and the iNLFFT is written by the first author, the Weiss algorithm and the error analysis is written by the second author. The first author determines the main direction of the whole writing, and distributed harder-to-read proofs to the second author who is far more mathematically inclined. We have used GPT-4o and Gemini 2.5 Pro for organizing the key points in some papers and summarizing the proofs, especially in Section 4 and Section 5.2.

5.2. Future Directions

The stability framework developed in this work extends naturally to the setting of infinite quantum signal processing (QSP), as demonstrated in recent studies [ALMTW24; L25; NSYL25]. In this infinite-dimensional regime, the Fourier coefficient sequence F becomes countably infinite, and the associated QSP matrix-valued function is represented as the nonlinear Fourier transform (NLFT) of an infinite sequence. The same outer function condition continues to characterize the stability and well-posedness of the inverse problem, although technical complications arise in handling convergence and analytic continuation.

Several natural questions emerge from this line of research. One major open problem is whether a stable inversion algorithm exists when the complementary polynomial $a^*(z)$ is not outer. In such cases, the inverse NLFT becomes highly ill-conditioned, and it is unclear whether one can find a regularized or generalized notion of inversion that remains numerically tractable. A related challenge is the design of a stable algorithm for the so-called fully-coherent regime, characterized by $\|b\|_\infty = 1$. This case lies on the boundary of the admissible region $\mathcal{S}_\eta$ and corresponds to singular QSP matrices, for which standard perturbation theory breaks down.

Another avenue of investigation is the deeper structural understanding of QSP behavior observed in numerical practice. In particular, the work of Wang, Dong, and Lin [WDL22] reveals that, despite the highly nonconvex nature of the QSP cost function, standard optimization algorithms can converge reliably to a global minimum when initialized at a fixed symmetric guess Φ_0. Their analysis identifies a "maximal solution" lying near Φ_0 and proves that the cost function is locally strongly convex in this region under the assumption $\|f\|_\infty = O(d^{-1})$. However, this condition is overly conservative compared to empirical observations, which suggest robust convergence even when $\|f\|_\infty$ is a constant independent of degree. Explaining this discrepancy between theory and practice remains an important theoretical challenge. Moreover, behaviors such as the clustering of global minima, slow convergence to non-maximal solutions, and the influence of symmetry breaking are only partially understood from the perspective of NLFT and warrant further investigation.

Despite recent progress introduced by Rossi and Chuang [RC22], multivariable QSP (M-QSP) remains only partially understood. The class of achievable polynomial transforms is more restricted than expected from bounded-norm and parity constraints. While the multivariable Fejér-Riesz theorem guarantees unitary completions for stable cases, its applicability to unstable transforms is unclear, and it remains conjectured to be the only fundamental mechanism behind M-QSP. Moreover, current constructions critically rely on query oracles sharing a common eigenbasis, which limits generalization to multi-qubit settings. Future work should explore alternative ansätze, fully characterize the expressive power of M-QSP, and identify relaxed conditions that enable lifting to the multivariable quantum singular value transformation framework.

Finally, an important theoretical question is whether lower bounds can be established for the complexity and the bit precision required for stable inverse QSP algorithms. While the current upper bounds are quasi-optimal, a matching lower bound would clarify the fundamental limits of efficient inversion. This includes both asymptotic complexity in the length of the sequence n and the bit-length r needed to maintain a given level of accuracy.

These open problems suggest a rich interplay between operator theory, harmonic analysis, and numerical computation in the context of quantum signal processing and nonlinear Fourier analysis. Progress on these directions would further strengthen the mathematical foundations of QSP and broaden its practical applicability.

REFERENCES

[MRTC21] John M. Martyn, Zane M. Rossi, Andrew K. Tan, and Isaac L. Chuang. “Grand Unification of Quantum Algorithms”. In: *PRX Quantum* 2.4 (Dec. 2021). ISSN: 2691-3399. DOI: 10.1103/prxquantum.2.040203. URL: http://dx.doi.org/10.1103/PRXQuantum.2.040203.

[GSLW19] András Gilyén, Yuan Su, Guang Hao Low, and Nathan Wiebe. “Quantum singular value transformation and beyond: exponential improvements for quantum matrix arithmetics”. In: *Proceedings of the 51st Annual ACM SIGACT Symposium on Theory of Computing*. STOC ’19. ACM, June 2019, pp. 193–204. DOI: 10.1145/3313276.3316366. URL: http://dx.doi.org/10.1145/3313276.3316366.

[H19] Jeongwan Haah. “Product Decomposition of Periodic Functions in Quantum Signal Processing”. In: *Quantum* 3 (Oct. 2019), p. 190. ISSN: 2521-327X. DOI: 10.22331/q-2019-10-07-190. URL: http://dx.doi.org/10.22331/q-2019-10-07-190.

[Y22] Lexing Ying. “Stable factorization for phase factors of quantum signal processing”. In: *Quantum* 6 (Oct. 2022), p. 842. ISSN: 2521-327X. DOI: 10.22331/q-2022-10-20-842. URL: http://dx.doi.org/10.22331/q-2022-10-20-842.

[DLNW24a] Yulong Dong, Lin Lin, Hongkang Ni, and Jiasu Wang. “Infinite quantum signal processing”. In: *Quantum* 8 (Dec. 2024), p. 1558. ISSN: 2521-327X. DOI: 10.22331/q-2024-12-10-1558. URL: https://doi.org/10.22331/q-2024-12-10-1558.

[WDL22] Jiasu Wang, Yulong Dong, and Lin Lin. “On the energy landscape of symmetric quantum signal processing”. In: *Quantum* 6 (Nov. 2022), p. 850. ISSN: 2521-327X. DOI: 10.22331/q-2022-11-03-850. URL: http://dx.doi.org/10.22331/q-2022-11-03-850.

[DLNW24b] Yulong Dong, Lin Lin, Hongkang Ni, and Jiasu Wang. “Robust Iterative Method for Symmetric Quantum Signal Processing in All Parameter Regimes”. In: *SIAM Journal on Scientific Computing* 46.5 (2024), A2951–A2971. DOI: 10.1137/23M1598192. URL: https://doi.org/10.1137/23M1598192.

[AMT24] Michel Alexis, Gevorg Mnatsakanyan, and Christoph Thiele. *Quantum signal processing and nonlinear Fourier analysis*. 2024. arXiv: 2310.12683 [quant-ph]. URL: https://arxiv.org/abs/2310.12683.

[T05] Ya-Ju Tsai. “SU(2) nonlinear fourier transform”. PhD thesis. UCLA, 2005. URL: https://www.proquest.com/dissertations-theses/i-su-2-nonlinear-fourier-transform/docview/305033060/se-2.

[L25] Lorenzo Laneve. *Generalized Quantum Signal Processing and Non-Linear Fourier Transform are equivalent*. 2025. arXiv: 2503.03026 [quant-ph]. URL: https://arxiv.org/abs/2503.03026.

[ALMTW24] Michel Alexis, Lin Lin, Gevorg Mnatsakanyan, Christoph Thiele, and Jiasu Wang. *Infinite quantum signal processing for arbitrary Szegő functions*. 2024. arXiv: 2407.05634 [quant-ph]. URL: https://arxiv.org/abs/2407.05634.

[NY24] Hongkang Ni and Lexing Ying. *Fast Phase Factor Finding for Quantum Signal Processing*. 2024. arXiv: 2410.06409 [quant-ph]. URL: https://arxiv.org/abs/2410.06409.

[NSYL25] Hongkang Ni, Rahul Sarkar, Lexing Ying, and Lin Lin. *Inverse nonlinear fast Fourier transform on SU(2) with applications to quantum signal processing*. 2025. arXiv: 2505.12615 [quant-ph]. URL: https://arxiv.org/abs/2505.12615.

[AG89] Gregory S. Ammar and William B. Gragg. “Numerical experience with a superfast real Toeplitz solver”. In: *Linear Algebra and its Applications* 121 (1989), pp. 185–206. ISSN: 0024-3795. DOI: https://doi.org/10.1016/0024-3795(89)90701-5. URL: https://www.sciencedirect.com/science/article/pii/0024379589907015.

[RC22] Zane M. Rossi and Isaac L. Chuang. “Multivariable quantum signal processing (M-QSP): prophecies of the two-headed oracle”. In: *Quantum* 6 (Sept. 2022), p. 811. ISSN: 2521-327X. DOI: 10.22331/q-2022-09-20-811. URL: http://dx.doi.org/10.22331/q-2022-09-20-811.

[SK95] Ali H. Sayed and Thomas Kailath. “Fast algorithms for generalized displacement structures and lossless systems”. In: *Linear Algebra and its Applications* 219 (1995), pp. 49–78. ISSN: 0024-3795. DOI: https://doi.org/10.1016/0024-3795(93)00193-4. URL: https://www.sciencedirect.com/science/article/pii/0024379593001934.