Alma Mater Studiorum · Università di Bologna

Scuola di Scienze
Dipartimento di Fisica e Astronomia
Corso di Laurea in Fisica

A code to perform harmonic analysis of a time series

Relatore:
Prof. Massimo Giovannozzi

Correlatore:
Prof. Francesco Giacomini

Presentata da:
Francesco Orso Pancaldi

Anno Accademico 2022/2023

Sommario

Nella fisica degli acceleratori, un parametro importante è il tune di betatrone, definito come la frequenza di oscillazione trasversale del fascio. La ricerca del tune è matematicamente equivalente al calcolo della frequenza principale (ed eventualmente delle frequenze successive) di una sequenza di numeri complessi. Lo scopo di questo lavoro di tesi è produrre un programma C++ che implementi alcuni algoritmi noti di ricerca della frequenza principale e delle frequenze successive, e confermare che gli errori di questi metodi concordino con l'andamento teorico. Questa procedura è stata svolta sia per sequenze di ampiezza costante, che di ampiezza variabile nel tempo. È stato inoltre verificato che, nel secondo caso, metodi alternativi per il calcolo dell'inviluppo non sono migliori di quello usato al momento. Infine, è stato studiato e implementato un nuovo algoritmo per il calcolo simultaneo delle due frequenze dominanti. È stato verificato che, se le frequenze sono abbastanza vicine, il nuovo algoritmo è più preciso degli altri considerati.

Abstract

In accelerator physics, an important parameter is the betatron tune, defined as the transverse oscillation frequency of the beam. Finding the tune is mathematically equivalent to calculating the principal frequency (and possibly the successive frequencies) of a sequence of complex numbers. The aim of this thesis work is to produce a C++ program that implements some known algorithms for finding the main frequency and subsequent frequencies, and to confirm that the errors of these methods agree with the theoretical trend. This procedure was carried out for sequences of both constant and time-varying amplitude. It was also verified that, in the latter case, alternative methods for calculating the envelope are no better than the one currently used. Finally, a new algorithm was studied and implemented implemented for the simultaneous calculation of the two dominant frequencies. It was verified that if the frequencies are close enough, the new algorithm is more accurate than the other ones considered.

Contents

Chapter 1

Introduction

1.1 Aim of this thesis

A circular particle accelerator can be thought of as a torus, with the charged particle beam travelling in the toroidal direction. However, the beam also oscillates in the transverse plane in the horizontal and vertical directions. The determination of the frequencies of the oscillations in the transverse plane carries much information about the properties of the dynamics of charged particles. The beam position can be measured at a given section of the accelerator, and the analysis of the turn-by-turn measurements allows the determination of the oscillation frequencies. In more abstract terms, the problem can be expressed as: given a sequence of (possibly complex) numbers (representing the outcome of the beam position measurements), what are the main frequencies in the series of numbers?

Efficient methods to do this need to be devised, and in fact, this represents an active research area and many useful advanced algorithms were found (see, e.g. [1] and [6]).

In this thesis work, we developed a C++ program that applies the most important algorithms described in these references. The correctness of the implementation was confirmed by studying synthetic signals and determining the behaviour of the errors associated with the determination of the signal frequencies, which is then compared with the theoretical expectations.

Additionally, a new algorithm was found that simultaneously calculates two frequencies, which has excellent performance in terms of error in the determination of the frequency. Furthermore, it is robust in the determination of close-by frequencies.

1.2 Notation and basic theory

As customary in the theory of signal processing, we use the term "signal" to indicate a time series, i.e. an (ordered) sequence of values, usually complex. A complex signal can be thought as a vector in $\mathbb{C}^N$, so it will be represented here as

$$z = \{z(n)\}_{n\in[N]} = (z(1), z(2), \ldots, z(N)) \tag{1.2.1}$$

where $z(n)$ is the nth component z, and the definition

$$[N] \coloneqq \{1, 2, \ldots, N\} \tag{1.2.2}$$

will be used. From now on, the terms "signal" and "vector" will be used almost interchangeably, except in the rare cases in which the vector cannot be interpreted as a time series (for example, if the said vector is a Fourier Transform).

Since we are working with complex numbers, we will always use the letter i to indicate $\sqrt{-1}$ and never as a summation index.

We will now define the scalar product that will be used in the rest of this work. We consider the sesquilinear form

$$z \cdot w \coloneqq \frac{1}{N}\sum_{n=1}^{N} z(n)w^*(n) \quad z, w \in \mathbb{C}^n \tag{1.2.3}$$

where $w^*(n)$ is the complex conjugate of $w(n)$. It is a rescaled version of the usual scalar product, so it is trivial to check that it verifies all the properties that make it a scalar product. An important note is that, differently from the usual convention in physics, this scalar product is linear in the first argument and conjugate-linear in the second.

We now consider Z, the discrete Fourier Transform of a signal z. It is a vector in $\mathbb{C}^N$, whose kth component is defined as

$$Z(k) \coloneqq \frac{1}{N}\sum_{n=1}^{N} z(n) \exp\left(-2\pi i \frac{k}{N} n\right). \tag{1.2.4}$$

We notice that, if we define the vector u_ν to be the harmonic of frequency ν, sampled over N points, we have

$$u_\nu(n) \coloneqq \exp(2\pi i \nu n) \tag{1.2.5}$$

and then

$$Z(k) = z \cdot u_{k/N}. \tag{1.2.6}$$

We then see that the discrete Fourier Transform of a signal can be thought of as a scalar product between the signal itself and an appropriate vector of $\mathbb{C}^N$. Furthermore, we note that

$$u_{k/N} \cdot u_{m/N} = \delta_{km} \tag{1.2.7}$$

where δ_{km} is the Kronecker symbol, so the vectors $\{u_{k/N}\}_{k\in[N]}$ are an orthonormal basis of $\mathbb{C}^N$.

We have seen that, if sampled over N points, the N harmonics with frequencies $\{k/N\}_{k\in[N]}$ correspond to N linearly independent vectors in $\mathbb{C}^N$. So the components of the discrete Fourier Transform of a signal z of lenth N can be thought of as the projection of z onto the vectors of the harmonic basis $\{u_{k/N}\}_{k\in[N]}$.

In general, for a given frequency ν we define the projection of z onto u_ν as

$$\phi(\nu) \coloneqq z \cdot u_\nu \tag{1.2.8}$$

we see that

$$Z(k) = \phi\left(\frac{k}{N}\right), \tag{1.2.9}$$

and, for a generic $L \in \mathbb{Z}$,

$$\phi(\nu + L) = \phi(\nu) \tag{1.2.10}$$

which is known as frequency aliasing. Therefore in this work we use (without loss of generality) $\nu \in]0, 1]$ for each frequency ν.

The task we want to address is to determine the M main frequencies in a given signal z. That is, the M frequencies whose amplitudes (referring to Eq. (1.2.8)) are the largest in modulus. For the sake of brevity, the frequencies will be ordered on the basis of the corresponding amplitudes, and we will then refer to the frequencies as the first frequency, second frequency, and so on.

A strategy for computing the first M frequencies in a signal might be to calculate its Discrete Fourier Transform (DFT), then choose the frequencies based on the value of $|\phi(k/N)|$. Therefore, the first frequency would be the one corresponding to the largest $|\phi(k/N)|$ and so on.

Given Eq. (1.2.7), we see that

$$z \cdot u_{k/N} = \left[z - \phi\left(\frac{m}{N}\right) u_{m/N}\right] \cdot u_{k/N} \tag{1.2.11}$$

so, except for the mth component (corresponding to the projection onto $u_{m/N}$), the DFTs of z and $z - \phi(m/N)\, u_{m/N}$ are equal. This means that the previous strategy is equivalent (in the sense that produces the same results) to the following:

- Compute the DFT of z, or equivalently we project z onto the N vectors of the harmonics basis. The first frequency corresponds to the largest $|\phi(k/N)|$, which we assume to be m/N.

- Replace z_{m-1} with z_m, defined as

$$z_m \coloneqq z_{m-1} - \phi\left(\frac{m}{N}\right) u_{m/N} \tag{1.2.12}$$

  such that its nth component is

$$z_1(n) \coloneqq z(n) - \phi\left(\frac{m}{N}\right) \exp\left(2\pi i \frac{m}{N} n\right). \tag{1.2.13}$$

  The signal z_1 corresponds to z but with its first frequency removed. In general, z_m is z without its first m frequencies, so $z_0 = z$.

- Repeat the first two steps M times, so that a list of M frequencies is found.

The precision of this strategy is determined by that of the DFT. Even if we only focus on the first frequency, we see that the error associated with the "plain" DFT is $1/N$. This means that the frequency k/N corresponding to the largest $|\phi(k/N)|$ differs from the true frequency ν_1 by $\mathcal{O}(1/N)$.

Other algorithms, treated later, give a better approximation of the true frequency and can be applied in this case. However, a problem arises when algorithms other than the DFT are used. Indeed, given two generic frequencies ν_h, ν_m, we have

$$u_{\nu_h} \cdot u_{\nu_m} = \frac{1}{N} e^{\pi i(\nu_h - \nu_m)} \frac{\sin[N\pi(\nu_h - \nu_m)]}{\sin[\pi(\nu_h - \nu_m)]} \tag{1.2.14}$$

so the vectors u_{ν_h}, u_{ν_m} are not orthonormal, although one can verify that, as expected, they do are orthonormal if $\nu_h, \nu_m \in \{k/N\}_{k\in[N]}$.

Therefore, the previous recursive strategy must be changed. In particular, the subsequent vectors onto which the signal is projected must be orthonormalised before proceeding to the next step of the approach. Notice that this was done implicitly when the vectors were $\{u_{k/N}\}_{k\in[N]}$. So, the final strategy to compute the first M frequencies of a signal z is (here it is be described its mth step):

- Estimate the current dominant frequency using an algorithm of choice. This frequency, which shall be called $\overline{\nu}_m$, is not guaranteed to be (and is usually not) in $\{k/N\}_{k\in[N]}$.

- Define $w_{\overline{\nu}_m}$, which is the result of the orthonormalisation of $u_{\overline{\nu}_m}$ with $\{w_{\overline{\nu}_k}\}_{k\in[m-1]}$. If Gram-Schmidt orthonormalisation process is used, then

$$w_{\overline{\nu}_m} \coloneqq \frac{u_{\overline{\nu}_m} - \sum_{k=1}^{m-1} (u_{\overline{\nu}_m} \cdot w_{\overline{\nu}_k})\, w_{\overline{\nu}_k}}{||u_{\overline{\nu}_m} - \sum_{k=1}^{m-1} (u_{\overline{\nu}_m} \cdot w_{\overline{\nu}_k})\, w_{\overline{\nu}_k}||} \tag{1.2.15}$$

where $||\cdot||$ is the norm induced by Eq. (1.2.3).

- Replace z_m with
$$z_m \coloneqq z_{m+1} - \phi(\overline{\nu}_m)\, w_{\overline{\nu}_m} \tag{1.2.16}$$
which is the original signal without its first m frequencies. Then add the orthonormalised vector $w_{\overline{\nu}_m}$ to the basis.

1.3 Numerical orthornomalization

1.3.1 Gram-Schmidt algorithm

It is known that the ordinary Gram-Schmidt method is numerically unstable, as it is often prone to significant numerical errors due to catastrophic cancellations (see [5] for example), a problem that manifests when subtractions of close numbers are performed. This is a very similar phenomenon to what occurs when errors in measurements are propagated. Indeed, consider two results of measurements $x \pm \Delta x$ and $y \pm \Delta y$, with relative errors $\Delta x/x$, $\Delta y/y$. The relative error on their difference is

$$\left|\frac{\Delta x - \Delta y}{x - y}\right| = \left|\frac{x\frac{\Delta x}{x} - y\frac{\Delta y}{y}}{x - y}\right| \tag{1.3.1}$$

and it can be much larger then the original relative errors, if x and y are sufficiently close.

When adapting this example to floating point arithmetic, it should be considered that x and y cannot take every value possible because a computer approximates them. Still, the order of magnitude of the numbers (which is what matter to us here) is the same.

In the Gram-Schmidt algorithm, the troubling subtraction occurs if two vectors are almost parallel. Using a different notation, for reasons that will become clear later, we want to orthonormalise the set of vectors $\{a_1, \dots, a_M\} = \{a_m\}_{m\in[M]}$ in $\mathbb{C}^N$. Assume $||a_1|| = ||a_2|| = 1$ and $|a_2 \cdot a_1| \approx 1$. The first assumption can actually be relaxed a little, by only requiring that a_1 and a_2 be not too small in norm, but since the vectors that will be orthonormalised are the harmonic vectors u_ν, we consider the case that matters to us. Since

$$||a_2 - (a_2 \cdot a_1)a_1|| = \sqrt{1 - |a_2 \cdot a_1|^2} \tag{1.3.2}$$

we see that $a_2 - (a_2 \cdot a_1)\, a_1$ is small in norm, which means its components are small too. Therefore, since at least some of the components of a_1 and a_2 are not small, catastrophic

cancellation happens in the calculations. Then, if we define q_1 and q_2 as orthonormalised vectors, it is probable that

$$q_2 \cdot q_1 = \delta \tag{1.3.3}$$

with $|\delta| \ll 1$.

We now want to compare how this error influences the calculation of the other elements of the orthonormalised basis if two different algorithms are used, namely the classical[1] Gram-Schmidt and the modified Gram-Schmidt. In the latter algorithm, the projections of the starting vectors $\{a_m\}_{m\in[M]}$ on some element of the orthonormalised basis q_j are subtracted as soon as the said vector is calculated. It means that when we compute the projection of a_m onto q_k, we calculate

$$\left[a_m - \sum_{j=1}^{k-1}(a_m \cdot q_j)q_j\right] \cdot q_k \tag{1.3.4}$$

instead of

$$a_m \cdot q_k. \tag{1.3.5}$$

It is easy to verify that, since the elements of the basis are orthogonal, the two operations are mathematically equivalent. However, the modified Gram-Schmidt algorithm behaves better numerically. To see this, we calculate the non-normalised third vector of the basis $\widetilde{q_3}$, equal to a_3 minus its projections onto a_1 and a_2. Proceeding analogously to [4], we obtain for the classical Gram-Schmidt algorithm

$$\begin{aligned}\widetilde{q_3} \cdot q_1 &= [a_3 - (a_3 \cdot q_1)q_1 - (a_3 \cdot q_2)q_2] \cdot q_1 \\ &= -(a_3 \cdot q_2)\delta \\ &= \mathcal{O}(\delta)\end{aligned} \tag{1.3.6}$$

$$\begin{aligned}\widetilde{q_3} \cdot q_2 &= -(a_3 \cdot q_1)\delta \\ &= \mathcal{O}(\delta)\end{aligned}$$

while for the modified Gram-Schmidt algorithm

$$\begin{aligned}\widetilde{q_3} \cdot q_1 &= [a_3 - (a_3 \cdot q_1)q_1 - ((a_3 - (a_3 \cdot q_1)q_1) \cdot q_2)q_2] \cdot q_1 \\ &= [a_3 - (a_3 \cdot q_1)q_1 - (a_3 \cdot q_2)q_2 + \delta(a_3 \cdot q_1)q_2] \cdot q_1 \\ &= -(a_3 \cdot q_2)\delta + (a_3 \cdot q_1)\delta^2 \\ &= \mathcal{O}(\delta)\end{aligned} \tag{1.3.7}$$

$$\begin{aligned}\widetilde{q_3} \cdot q_2 &= [a_3 - (a_3 \cdot q_1)q_1 - (a_3 \cdot q_2)q_2 + \delta(a_3 \cdot q_1)q_2] \cdot q_2 \\ &= 0.\end{aligned}$$

[1] The Gram-Schmidt algorithm considered until now.

which shows that the numerical errors are lower.

1.3.2 QR decomposition

An alternative way to orthonormalise a set of vectors is by means of QR decomposition. Given a full-rank matrix $A \in M_{N\times M}(\mathbb{C})$ (so it is $M \leq N$), it holds

$$A = QR \tag{1.3.8}$$

for some matrices $Q \in M_{N\times N}(\mathbb{C})$, $R \in M_{N\times M}(\mathbb{C})$ with

$$\begin{aligned} Q^T &= Q^{-1} \\ R_{jk} &= 0 \iff j > k \end{aligned} \tag{1.3.9}$$

so that Q is orthogonal and R is upper triangular. Furthermore, the latter condition implies that

$$A = Q'R' \tag{1.3.10}$$

where $Q' \in M_{N\times M}(\mathbb{C})$ is the matrix containing the first M columns of Q and $R' \in M_{M\times M}(\mathbb{C})$ is the matrix containing the first M rows of R. The product $Q'R'$ is called the *thin* QR decomposition of A, since it contains all the information but requires fewer data.

We can use QR decomposition to orthonormalise a set of M vectors of $\mathbb{C}^N$ $\{a_m\}_{m\in[M]}$, if we choose A such that a_m is its mth column (then A is full-rank if $\{a_m\}_{m\in[M]}$ are linearly independent). Indeed, the columns of Q', $\{q_m\}_{m\in[M]}$, are an orthonormal system with the same span as $\{a_m\}_{m\in[M]}$.

The Gram-Schmidt algorithm proves the existence of Q' and R'. Indeed, if we have a set of vectors $\{a_m\}_{m\in[M]}$ and orthonormalise them, obtaining the set $\{q_m\}_{m\in[M]}$, we can choose Q' such that q_m is its mth column. Also, we can choose R' such that the element in the j-th row and k-th column is

$$\begin{aligned} R'_{jk} &\coloneqq a_j \cdot q_i \qquad j < k \\ R'_{jk} &\coloneqq 0 \qquad j > k \\ R'_{jk} &\coloneqq ||a_j|| \qquad j = k \end{aligned} \tag{1.3.11}$$

then we see that the constrains on Q' and R' following from the fact that Q was orthogonal and R was upper triangular are satisfied.

However, QR decomposition is not unique. Indeed, if we define a matrix $S \in M_{M\times M}(\mathbb{C})$ as

$$S_{jk} \coloneqq \exp(i\varphi_j)\,\delta_{jk}, \qquad \varphi_j \in \mathbb{R} \tag{1.3.12}$$

then

$$A = Q'R' = (Q'S)\,(S^{-1}R') \tag{1.3.13}$$

and it is easy to check that $Q'S$ is orthogonal and $S^{-1}R'$ is upper triangular.

If we require

$$R'_{jj} > 0 \qquad \forall j \in [M] \tag{1.3.14}$$

then both the regular QR decomposition and the thin one are unique, also the columns of Q' are the vectors that one obtains after applying the Gram-Schmidt algorithm on $\{a_m\}_{m\in[M]}$.

In the end, we can define an alternative strategy to orthonormalise a set of vectors:

- We calculate the QR decomposition, no matter the algorithm used, of the matrix whose columns are the vectors we want to orthonormalise.
- We choose the matrix S of Eq. (1.3.13) as

$$S_{jk} := \exp[i \arg(R_{jj})]\delta_{jk}. \tag{1.3.15}$$

- Then the following holds

$$\arg[(S^{-1}R')_{jj}] = \arg(S_{jj}^{-1}\, R'_{jj}) = 0 \tag{1.3.16}$$

  and the columns of $Q'S$ are orthonormalised vectors of our interest.

This short digression in the domain of the QR decomposition was made to show that there are many other ways to orthonormalise a set of vectors. Quoting from [5]:

> Householder QR is much less sensitive to roundoff error than Gram-Schmidt, even with modification, although Gram-Schmidt is more efficient if an explicit representation of Q' desired.

In the code developed in this work, orthonormalisation via the modified Gram-Schmidt algorithm was implemented with the aim of efficiency (since the vectors are in $\mathbb{C}^N$ with N up to 10^5 or 10^6). If other algorithms are preferred, the orthonormalisation can be made with the strategy shown above. For details on the implementation of additional orthonormalisation algorithms, see Section 4.1.

1.4 Amplitude error

We consider as usual a signal that is a weighted sum of harmonics

$$z(n) = \sum_k A_k \exp\left(2\pi i \nu_k n\right) \tag{1.4.1}$$

where the amplitudes $\{A_k\}_{k\in\dots}$ are the amplitudes of the various harmonics. Then, if we determine the first frequency $\overline{\nu}_1$, our best estimate for A_1 is

$$\overline{A}_1 = \phi(\overline{\nu}_1) = \sum_k A_k (u_{\nu_k} \cdot u_{\overline{\nu}_1}) \tag{1.4.2}$$

and the scalar product of two harmonics is given by Eq. (1.2.14), from which one obtains that

$$\overline{A}_1 - A_1 = \mathcal{O}(1/N) \tag{1.4.3}$$

no matter the error made on ν_1. Indeed, even if the frequency were calculated with infinite precision, that is, $\overline{\nu_1} = \nu_1$, there would be other harmonics in the signal, whose projection on u_{ν_1} would be $\mathcal{O}(1/N)$.

This remark was made for two reasons: the first one being that there are no ways to improve the precision in the calculation of $\overline{A}_1$, since the error comes from the presence of the other harmonics, about which we cannot assume anything.

The second is that the error on $\overline{A}_1$ propagates onto the determination of the following frequencies. So, if a signal is such that $\overline{A}_1$ cannot be computed with good precision, then we cannot caclulate accurately any of its following frequencies. A frequent example of this is when $|\nu_2 - \nu_1| \ll 1$, since in that case the scalar product of the harmonics $u_{\overline{\nu}_1}$ and u_{ν_2} is significantly different from zero. For this reason, the new "Two frequency" algorithm was implemented to address this case. For more information, see Section 2.4.

Chapter 2

Constant-amplitude signals

This chapter deals with the algorithms used for the analysis of a signal made of harmonics with constant amplitude, i.e. of the form

$$z(n) = \sum_{l=1}^{L} A_l \exp(2\pi i \nu_l n), \qquad n \in [N] \tag{2.0.1}$$

with

$$\nu_l \in]0, 1], \qquad A_l \in \mathbb{C}, \qquad l \in [L] \tag{2.0.2}$$

also, if no two amplitudes have approximately the same modulus, we can assume without loss of generality

$$|A_l| > |A_{l+1}|, \qquad 1 \leq l \leq L. \tag{2.0.3}$$

The other case is troublesome for the algorithms of Sections 2.1, 2.2 and 2.3 since they all rely on having a single harmonic with much smaller perturbations. The algorithm in Section 2.4 does not have these constraints and can be used successfully if, for example,

$$|A_1| \approx |A_2| > |A_3|. \tag{2.0.4}$$

The algorithms of Sections 2.1, 2.2 and 2.3 were tested on signals of the form

$$\begin{aligned} z(n) &= A_1 \exp(2\pi i \nu_1 n) \\ z(n) &= A_1 \exp(2\pi i \nu_1 n) + \sum_{k=2}^{6} B_k \exp(2\pi i k \nu_1 n) \end{aligned} \tag{2.0.5}$$

with

$$\begin{gathered} A_1, B_k \in \mathbb{C}, \qquad |B_k| = |A_1|/10 = 1/10, \qquad 2 \leq k \leq 6 \\ 64 = 2^6 \leq N \leq 2^{22} = 4194304, \\ \nu_1 = 0.163618 \end{gathered} \tag{2.0.6}$$

and $\arg(A_1)$, $\{\arg(B_k)\}_{2\leq k\leq 6}$ chosen randomly. The value of ν_1 was chosen with the aim of minimising effects of resonance with other harmonics.

The frequency error $|\overline{\nu}_1 - \nu_1|$ was plotted against the number of points in the signal, which is always a power of 2. That's because we compute the DFTs of the signal, which are implemented as Fast Fourier Transforms (FFTs).

The signals with just a single harmonic were considered to ensure the fact that the error was constant and equal to the machine error, since in that case the formulae of the algorithms are exact. The ones with a single harmonic and some disturbing additional were considered to confirm that the error scales with the number of points as theoretically predicted.

To test the impact of the orthonormalisation procedure on the accuracy of the algorithms, and to compare the results with the single-frequency case, the algorithms of Sections 2.1 and 2.2 were also tested on signals of the form

$$z(n) = \sum_{l=1}^{4} \left[A_l \exp(2\pi i \nu_l n) + \sum_{k=2}^{6} B_{lk} \exp(2\pi i k \nu_l n) \right] \tag{2.0.7}$$

with

$$\begin{gathered} A_l, B_{lk} \in \mathbb{C}, \qquad |B_{lk}| = |A_l|/10, \qquad l \in [4], \qquad 2 \leq k \leq 6 \\ |A_l| = |A_1| \,(\,2/3\,)^{l-1} = (\,2/3\,)^{l-1} \\ 64 = 2^6 \leq N \leq 2^{22} = 4194304 \\ (\nu_1, \nu_2, \nu_3, \nu_4) = (0.163618, 0.826492, 0.365437, 0.593813) \end{gathered} \tag{2.0.8}$$

and $\{\arg(A_l)\}_{l\in[4]}$, $\{\arg(B_{lk})\}_{l\in[4],2\leq k\leq 6}$ chosen randomly. Like in the single harmonic case, the value of $\{\nu_k\}_{k\in[4]}$ was chosen with the aim of minimising effects of resonance with other harmonics.

The error on the fourth frequency $|\overline{\nu}_4 - \nu_4|$ were plotted against the number of points in the signal, that is always a power of 2. We chose to plot the error on the fourth (and last) frequency, since any errors in determining other frequencies or amplitudes will propagate and result in lower precision on ν_4. Then, if the error behaves as expected, it is a confirmation that the calculation of the previous frequencies and amplitudes also went as expected.

Since the signals used for testing the algorithm of Section 2.4 are not used anywhere else, they will be described in that section.

To avoid overloading the notation, in the following sections we will use ν_{j_m} instead of j_m/N to represent the j_mth frequency of the Fourier Transform. Usually, ν_{j_1} will be the frequency of the first peak of the DFT, ν_{j_2} of the second and so on.

2.1 Interpolated FFT

This algorithm is called *interpolated* FFT since knowing the behaviour of the peak and its neighbours allows reducing the error in the determination of the main frequency. It was studied in [1].

If we consider a signal that is a single harmonic

$$z(n) = A_1 \exp(2\pi i \nu_1 n) \tag{2.1.1}$$

and we calculate its FFT, the j_1th component of the FFT is given by the projection onto $u_{\nu_{j_1}}$, namely

$$\phi(\nu_{j_1}) = \frac{A_1}{N} e^{\pi i(\nu_1 - \nu_{j_1})} \frac{\sin[N\pi(\nu_1 - \nu_{j_1})]}{\sin[\pi(\nu_1 - \nu_{j_1})]} \tag{2.1.2}$$

where it was used Eq. (1.2.14). Then we calculate

$$\begin{aligned} -\frac{\phi(\nu_{j_1\pm1})e^{\pm\frac{\pi i}{N}}}{\phi(\nu_{j_1})} &= -\frac{\sin[N\pi(\nu_1 - \nu_{j_1} \mp 1/N)]}{\sin[N\pi(\nu_1 - \nu_{j_1})]} \frac{\sin[\pi(\nu_1 - \nu_{j_1})]}{\sin[\pi(\nu_1 - \nu_{j_1} \mp 1/N)]} \\ &= \frac{\sin[\pi(\nu_1 - \nu_{j_1})]}{\sin[\pi(\nu_1 - \nu_{j_1} \mp 1/N)]} \\ &= \{\cos(\pi/N) \mp \sin(\pi/N)\cot[\pi(\nu_1 - \nu_{j_1})]\}^{-1} \end{aligned} \tag{2.1.3}$$

from which an expression for ν_1 can be found as

$$\nu_1 = \frac{j_1}{N} \pm \frac{1}{\pi} \arctan\left[\frac{\phi(\nu_{j_1\pm1})e^{\pm\frac{\pi i}{N}}\sin(\pi/N)}{\phi(\nu_{j_1}) + \phi(\nu_{j_1\pm1})e^{\pm\frac{\pi i}{N}}\cos(\pi/N)}\right]. \tag{2.1.4}$$

We note that this expression for ν_1, the frequency we want to find, contains only quantities that can be found by calculating the FFT of the signal z.

Equation (2.1.4) holds for any choice of j_1, but if additional (disturbing) harmonics are present, the components of the FFT $\{\phi(k/N)\}_{k\in[N]}$ differ from those of the pure harmonic. The components of the FFT that contain the most information about the original harmonic u_{ν_1} are the ones with the smallest relative change due to the other harmonics, so those near ν_1. So, the best choice for j_1 is the one such that $|\phi(\nu_{j_1})| = |\phi(j_1/N)|$ is maximum.

If we take the modulus of the first and last row of Eq. (2.1.3), we obtain

$$\left|\frac{\phi(\nu_{j_1\pm1})}{\phi(\nu_{j_1})}\right| = |\cos(\pi/N) \mp \sin(\pi/N)\cot[\pi(\nu_1 - \nu_{j_1})]|^{-1} \tag{2.1.5}$$

To determine the sign of the right hand side, we use the fact that it is equal to the second row of Eq. (2.1.3) and we study its sign. In each of the cases considered, we also choose which sign to use in the previous equation.

- If $\nu_1 - \nu_{j_1} < 0$, then $|\phi(\nu_{j_1-1})| > |\phi(\nu_{j_1+1})|$. So, we choose to work with $|\phi(\nu_{j_1})|$ and $|\phi(\nu_{j_1-1})|$, since they are the largest two amplitudes. So we use the lower sign in equation ($-$ in $\pm$ and $+$ in $\mp$).
  Also, since the peak is in ν_{j_1}th bin, it is $\nu_{j_1} - \frac{1}{2N} < \nu_1 < \nu_{j_1} + \frac{1}{2N}$. Thus, $\nu_1 - \nu_{j_1} + 1/N > 0$.

- If $\nu_1 - \nu_{j_1} > 0$, with analogous reasoning, we choose the upper sign, and we obtain $\nu_1 - \nu_{j_1} + 1/N < 0$.

The case $\nu_1 - \nu_{j_1} = 0$ is irrelevant, as in theory ν_1 is irrational.

We conclude that the second row of Eq. (2.1.3) is negative; then it is the opposite of its modulus.

We also have implicitly assumed that $|\phi(\nu_{j_1\pm1})|$ exists for every value of ν_{j_1}. This is a consequence of the periodicity of the DFT.

We obtain then

$$\nu_1 = \frac{j_1}{N} \pm \frac{1}{\pi} \arctan\left[\frac{|\phi(\nu_{j_1\pm1})| \sin(\pi/N)}{|\phi(\nu_{j_1})| + |\phi(\nu_{j_1\pm1})| \cos(\pi/N)}\right] \tag{2.1.6}$$

which is the formula for the real Interpolated FFT (as opposed to Eq. (2.1.4), that might be called complex Interpolated FFT).

Now we want to study the errors of both the real (Eq. (2.1.6)) and complex (Eq. (2.1.4)) Interpolated FFTs. In general, since ν_1 is the true frequency unknown to us, the left-hand sides of Eq. (2.1.4) should be replaced by $\overline{\nu}_1$, our best guess for ν_1. If there are no disturbing harmonics, the signal is a pure harmonic and $\nu_1 = \overline{\nu}_1$. So, theoretically, we have infinite precision, but since we are working with floating-point arithmetic, we expect an error roughly equal to the machine precision. If there are additional harmonics, then $\phi(\nu_{j_1})$ and $\phi(\nu_{j_1\pm1})$ differ from the values they would have for a pure harmonic by $\mathcal{O}(1/N)$. Then, for the complex case, we make the following substitutions in Eq. (2.1.4)

$$\begin{aligned} &\phi(\nu_{j_1}) \to \phi(\nu_{j_1}) + \frac{\alpha}{N}, \qquad \alpha \in \mathbb{C} \\ &\phi(\nu_{j_1\pm1}) \to \phi(\nu_{j_1\pm1}) + \frac{\alpha_\pm}{N}, \qquad \alpha_\pm \in \mathbb{C} \end{aligned} \tag{2.1.7}$$

The error on the frequency is then

$$\begin{aligned}
\overline{\nu}_1 - \nu_1 &= \mp \frac{1}{\pi} \arctan \left\{ \frac{\phi(\nu_{j_1 \pm 1}) e^{\pm \frac{\pi i}{N}} \sin(\pi/N)}{\phi(\nu_{j_1}) - \phi(\nu_{j_1 \pm 1}) e^{\pm \frac{\pi i}{N}} \cos(\pi/N)} \right\} \\
&\quad \pm \frac{1}{\pi} \arctan \left\{ \frac{[\phi(\nu_{j_1 \pm 1}) + \alpha_\pm/N] e^{\pm \frac{\pi i}{N}} \sin(\pi/N)}{[\phi(\nu_{j_1}) + \alpha/N] - [\phi(\nu_{j_1 \pm 1}) + \alpha_\pm/N] e^{\pm \frac{\pi i}{N}} \cos(\pi/N)} \right\} \\
&= \mp \frac{\phi(\nu_{j_1 \pm 1}) e^{\pm \frac{\pi i}{N}} \pi}{\phi(\nu_{j_1}) - \phi(\nu_{j_1 \pm 1}) e^{\pm \frac{\pi i}{N}}} \frac{1}{N} + \mathcal{O}\left(\frac{1}{N^3}\right) \\
&\quad \pm \frac{\phi(\nu_{j_1 \pm 1}) e^{\pm \frac{\pi i}{N}} \pi}{\phi(\nu_{j_1}) - \phi(\nu_{j_1 \pm 1}) e^{\pm \frac{\pi i}{N}}} \frac{1}{N} + \mathcal{O}\left(\frac{1}{N^2}\right) \\
&= \mathcal{O}\left(\frac{1}{N^2}\right)
\end{aligned} \tag{2.1.8}$$

where Eq. (2.1.4) was used.

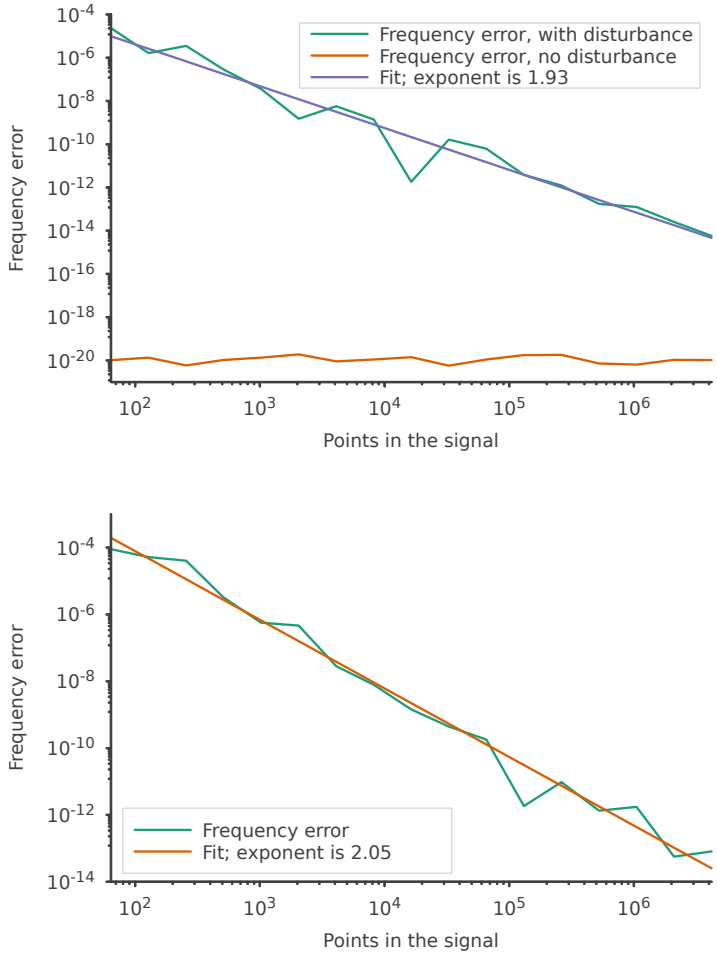

Figure 2.1: Errors for the complex Interpolated FFT algorithm. Top is single frequency case (with and without disturbing harmonics), bottom is multiple frequency case. The plots are in logarithmic scale and the function fitted is $y = C/x^{\alpha}$, with α shown in the legend.

We see that the error behaves as theoretically predicted.

The complex and real variants of the algorithms are expected to have the same properties, and this is verified later. For this reason, only the real variant is studied in [1]. They were both implemented in the program to verify that an algorithm which also used the phases of the elements of the FFT (instead of their modulus only) does not lose precision.

With the same exact reasoning of Eq. (2.1.8), and by using

$$|z + \mathcal{O}(1/N)| = |z| + \mathcal{O}(1/N), \qquad z \in \mathbb{C} \tag{2.1.9}$$

we obtain that the errors on the real Interpolated FFT are $\mathcal{O}(1/N^2)$.

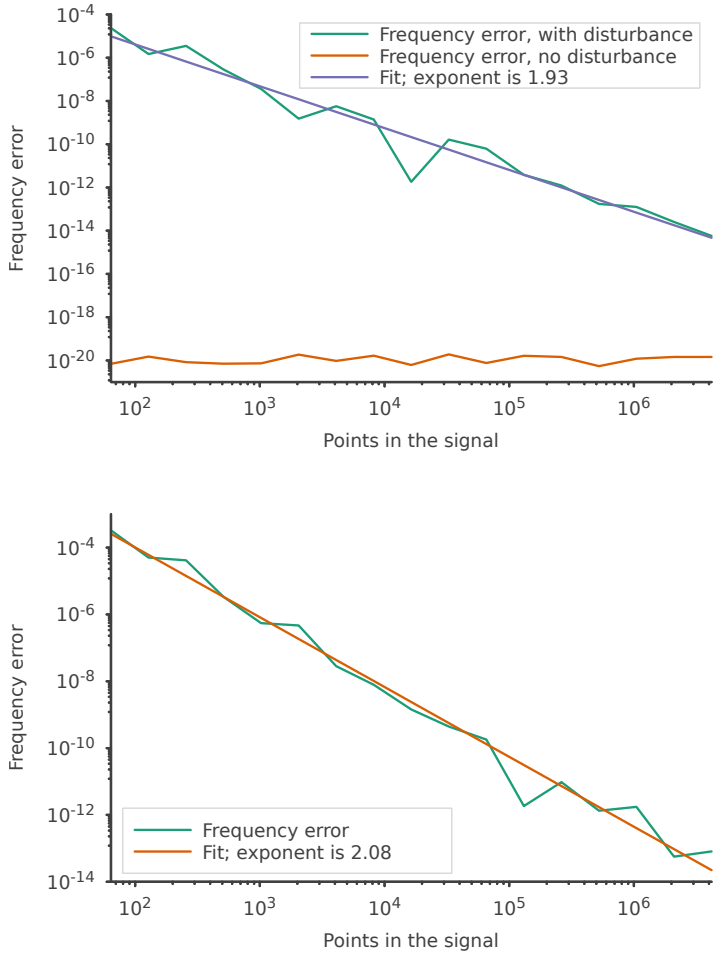

Figure 2.2: Errors for the real Interpolated FFT algorithm. Top is single frequency case (with and without disturbing harmonics), bottom is multiple frequency case. The plots are in logarithmic scale and the function fitted is $y = C/x^{\alpha}$, with α shown in the legend.

We see that the error behaves as theoretically predicted.

2.2 Interpolated FFT with windowing

We now consider a *windowed* signal. That is, the j_1th component of its DFT is

$$\phi(\nu_{j_1}) = \frac{1}{N}\sum_{n=1}^{N} z(n)\chi(n)\exp(2\pi i\nu_{j_1} n) \tag{2.2.1}$$

where $\chi = \{\chi(n)\}_{n\in[N]}$ is the window or filter. If we choose the Hanning filter

$$\chi(n) = 2\sin^2\left(\frac{n\pi}{N}\right) \tag{2.2.2}$$

as it was done in [1], we get for a pure harmonic of amplitude A_1 and frequency ν_1

$$\begin{aligned}\phi(\nu_{j_1}) &= \frac{A_1}{N}\sum_{n=1}^{N}\left\{e^{2\pi i(\nu_1-\nu_{j_1})n} - \frac{e^{2\pi i(\nu_1-\nu_{j_1-1})n}}{2} - \frac{e^{2\pi i(\nu_1-\nu_{j_1+1})n}}{2}\right\} \\ &= \frac{A_1}{N}e^{\pi i(\nu_1-\nu_{j_1})}\left\{\frac{\sin[N\pi(\nu_1-\nu_{j_1})]}{\sin[\pi(\nu_1-\nu_{j_1})]}\right. \\ &\left.+e^{-\frac{\pi i}{N}}\frac{\sin[N\pi(\nu_1-\nu_{j_1}+1/N)]}{2\sin[\pi(\nu_1-\nu_{j_1}+1/N)]} + e^{\frac{\pi i}{N}}\frac{\sin[N\pi(\nu_1-\nu_{j_1}-1/N)]}{2\sin[\pi(\nu_1-\nu_{j_1}-1/N)]}\right\} \\ &= -\frac{A_1\sin^2[\pi/N]\sin[N\pi(\nu_1-\nu_{j_1})]\cos[\pi(\nu_1-\nu_{j_1})]e^{\pi i(\nu_1-\nu_{j_1})}}{N\sin[\pi(\nu_1-\nu_{j_1}+1/N)]\sin[\pi(\nu_1-\nu_{j_1})]\sin[\pi(\nu_1-\nu_{j_1}-1/N)]}.\end{aligned} \tag{2.2.3}$$

We choose ν_{j_1} to be the frequency corresponding to the peak in the DFT. Then we calculate

$$\begin{aligned}\frac{|\phi(\nu_{j_1\pm1})|}{|\phi(\nu_{j_1})|} &= \left|\frac{\cos[\pi(\nu_1-\nu_{j_1}\mp1/N)]\sin[\pi(\nu_1-\nu_{j_1}\pm1/N)]}{\cos[\pi(\nu_1-\nu_{j_1})]\sin[\pi(\nu_1-\nu_{j_1}\mp2/N)]}\right| \\ &= -\frac{\sin[2\pi(\nu_1-\nu_{j_1})]\pm\sin[2\pi/N]}{\sin[2\pi(\nu_1-\nu_{j_1})]\cos[2\pi/N]\mp2\cos^2[\pi(\nu_1-\nu_{j_1})]\sin[2\pi/N]}\end{aligned} \tag{2.2.4}$$

since, given $|\nu_1-\nu_{j_1}|<1/N$, the choice of sign determines the sign of the sines.

By clearing the denominators and using

$$\begin{aligned}&\frac{|\nu_1-\nu_{j_1}|}{N}<\frac{1}{2} \implies \cos[2\pi(\nu_1-\nu_{j_1})]>0 \implies \\ &2\cos^2[\pi(\nu_1-\nu_{j_1})] = 1-\sqrt{1-\sin^2[2\pi(\nu_1-\nu_{j_1})]}\end{aligned} \tag{2.2.5}$$

we get

$$\begin{aligned}&|\phi(\nu_{j_1\pm1})|\sqrt{1-x^2\sin^2(2\pi/N)} \\ &= x[|\phi(\nu_{j_1})|+|\phi(\nu_{j_1\pm1})|\cos(2\pi/N)]\pm(|\phi(\nu_{j_1})|-|\phi(\nu_{j_1\pm1})|)\end{aligned} \tag{2.2.6}$$

where it was defined

$$x \coloneqq \frac{\sin[2\pi(\nu_1-\nu_{j_1})]}{\sin(2\pi/N)}. \tag{2.2.7}$$

Eq. (2.2.6) can be recast as

$$\gamma_2x^2+\gamma_1x+\gamma_0=0 \tag{2.2.8}$$

where

$$\begin{aligned}\gamma_2 &= |\phi(\nu_{j_1})|^2 + |\phi(\nu_{j_1\pm1})|^2 + 2|\phi(\nu_{j_1})||\phi(\nu_{j_1\pm1})|\cos(2\pi/N)\\ \gamma_1 &= \pm 2[|\phi(\nu_{j_1})| + |\phi(\nu_{j_1\pm1})|\cos(2\pi/N)][|\phi(\nu_{j_1})| - |\phi(\nu_{j_1\pm1})|]\\ \gamma_0 &= |\phi(\nu_{j_1})|^2 - 2|\phi(\nu_{j_1})||\phi(\nu_{j_1\pm1})|\end{aligned} \tag{2.2.9}$$

and for each choice of the sign there are two solutions for x, but we see that:

- If $|\phi(\nu_{j_1+1})| > |\phi(\nu_{j_1-1})|$, then $\nu_1 > \nu_{j_1}$ so $x > 0$. Since in this case $-\gamma_1 < 0$ (and it is always $\gamma_2 > 0$), the solution of the equation with minus sign is negative, it must be excluded. We concluded that if we choose the plus sign in the equation, the only acceptable solution is the one with the plus sign.
- With the same reasoning, if we choose the minus sign in the equation, the only acceptable solution is the one with the minus sign.

So we get

$$\nu_1 = \nu_{j_1} + \frac{1}{2\pi}\arcsin\left[\frac{-\gamma_1 \pm \sqrt{\gamma_1^2 - 4\gamma_0\gamma_2}}{2\gamma_2}\sin\left(\frac{2\pi}{N}\right)\right] \tag{2.2.10}$$

where the sign in $\pm$ is the same of the sign in γ_1.

We now want to study the errors for this algorithm. If additional harmonics are present, from Eq. (2.2.3) we see that we need to make the following substitutions in $\gamma_0, \gamma_1, \gamma_2$

$$\begin{aligned}\phi(\nu_{j_1}) &\to \phi(\nu_{j_1}) + \frac{\alpha}{N^3} \qquad \alpha \in \mathbb{C}\\ \phi(\nu_{j_1\pm1}) &\to \phi(\nu_{j_1\pm1}) + \frac{\alpha_\pm}{N^3} \qquad \alpha_\pm \in \mathbb{C}\end{aligned} \tag{2.2.11}$$

and after long calculations we obtain that the error scales as $\mathcal{O}(1/N^4)$ [1].

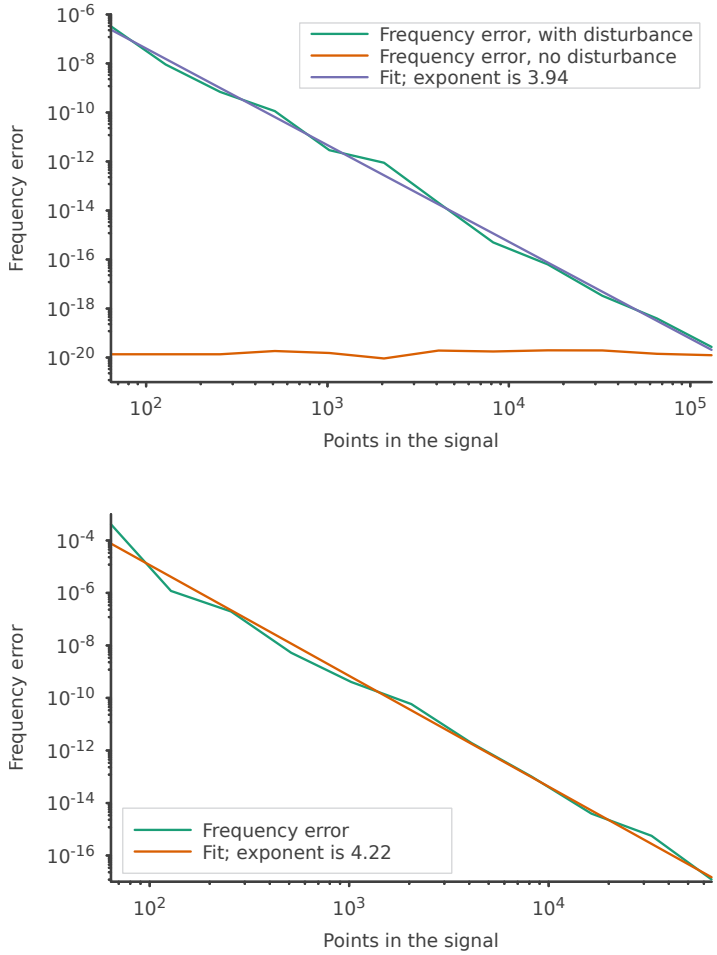

Figure 2.3: Errors for the Interpolated FFT with windowing algorithm. Top is single frequency case (with and without disturbing harmonics), bottom is multiple frequency case. The plots are in logarithmic scale and the function fitted is $y = C/x^{\alpha}$, with α shown in the legend.

We see that the error behaves as theoretically predicted.

2.3 Birkhoff Average

An alternative approach, which is not FFT-based, for the determination of the frequency is the use of the Average Phase Advance (APA)

$$\overline{\nu}_1 = \frac{1}{N-1} \sum_{n=1}^{N-1} \arg\left[\frac{z(n+1)}{z(n)}\right] \tag{2.3.1}$$

where $\arg\big[z(n+1)\big] - \arg\big[z(n)\big]$ was not used to avoid problems caused by choosing the arguments in different foldings of $\mathbb{C}$.

To improve this method, we consider the average with Birkhoff weights

$$w_n(t) \coloneqq \exp\left[\frac{1}{t^n(1-t)^n}\right], \qquad t \in]0,1[\tag{2.3.2}$$

for $n = 1$. The new formula is

$$\overline{\nu}_1 = \frac{\sum_{n=1}^{N-1} w\left(\frac{n}{N}\right) \arg\left[\frac{z(n+1)}{z(n)}\right]}{\sum_{n=1}^{N-1} w\left(\frac{n}{N}\right)}. \tag{2.3.3}$$

It can be proven (see [6] and references therein) that the error goes to 0 more rapidly than any power of N. However, in both [6] and in our simulations, it was found that the typical error scales as $\mathcal{O}(1/N^7)$, due to the finite machine precision.

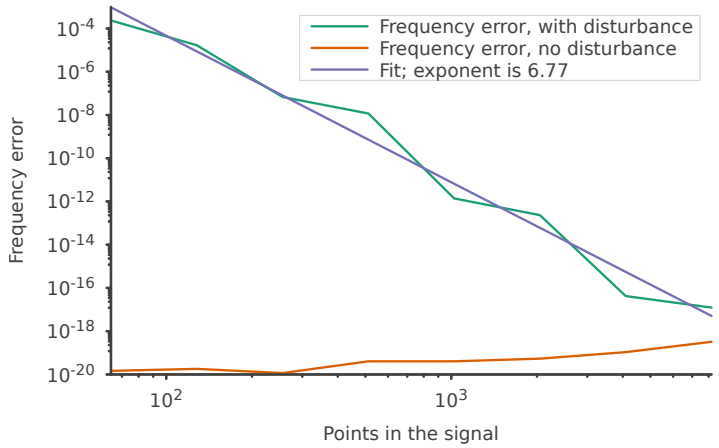

Figure 2.4: Errors for the Birkhoff Average algorithm, in the single frequency case (with and without disturbing harmonics). The plots are in logarithmic scale and the function fitted is $y = C/x^\alpha$, with α shown in the legend.

We see that the error behaves as found in [6].

2.4 Two-frequency approach

It might happen that the first two frequencies ν_1, ν_2 are very close. In this case, the algorithm of choice calculates well the first frequency, but fails on the second one. This is because a significant error is made in removing the harmonic of frequency $\overline{\nu_1}$, thus altering the deduction of all subsequent frequencies (see Section 1.4 for more details).

To address this problem, a new algorithm is proposed with the aim of calculating the first two frequencies of the signal simultaneously.

We assume here that the signal has only two harmonics

$$z(n) = A_1 \exp(2\pi i \nu_1 n) + A_2 \exp(2\pi i \nu_2 n) \tag{2.4.1}$$

and without loss of generality we assume that $\nu_1 < \nu_2$, but we do not require $|A_1| > |A_2|$. We also define the indices of the two peaks in the FFT to be j_1, j_2.

With these assumptions it can be proven (see Appendix A) that the two frequencies are given by

$$\nu_\pm = \frac{-\gamma_1 \pm \sqrt{\gamma_1^2 - 4\gamma_2\gamma_0}}{2\gamma_2} \tag{2.4.2}$$

and $\nu_1 = \min\{\nu_-, \nu_+\}, \nu_2 = \max\{\nu_-, \nu_+\}$. The coefficients $\gamma_2, \gamma_1, \gamma_0$ are reported in Eq. (A.0.17).

If there are additional (disturbing) harmonics, we have to replace the ϕ terms in Eq. (A.0.17) with terms like those of Eq. (2.1.7) and this gives the following

$$\begin{aligned} \gamma_2 &\to \gamma_2 + \mathcal{O}(1/N^3) \\ \gamma_1 &\to \gamma_1 + \mathcal{O}(1/N^4) \\ \gamma_0 &\to \gamma_0 + \mathcal{O}(1/N^5) \end{aligned} \tag{2.4.3}$$

where it was used $\sin(\alpha/N) = \mathcal{O}(1/N)$, for a generic $\alpha \in \mathbb{C}$. Then, by using

$$\left[\alpha + \mathcal{O}(1/N^\beta)\right]^\gamma = \alpha + \mathcal{O}(1/N^\beta), \qquad \alpha \in \mathbb{C},\ \beta \in \mathbb{N},\ \gamma \in \mathbb{R} \tag{2.4.4}$$

we find that

$$\nu \to \nu + \mathcal{O}(1/N^3)\,. \tag{2.4.5}$$

Therefore, in the presence of additional harmonics, we expect the error on the determination of the two frequencies to scale as $\mathcal{O}(1/N^3)$, which represents an improvement with respect to the case of the Interpolated FFT.

We test the algorithm on signals like the one in Eq. (2.4.1) and also on ones of the form

$$z(n) = \sum_{l=1}^{2} \left[A_l \exp(2\pi i \nu_l n) + \sum_{k=2}^{6} B_{lk} \exp(2\pi i k \nu_l n) \right] \tag{2.4.6}$$

with

$$\begin{gathered} A_l, B_{lk} \in \mathbb{C}, \qquad l \in [4], \qquad 2 \le k \le 6 \\ |B_{1k}| = |B_{2k}| = |A_{1k}|/10 = |A_{2k}|/10 = 1/10 \\ 64 = 2^6 \le N \le 2^{22} = 4194304 \\ (\nu_1, \nu_2) = (0.163618, 0.163718) \end{gathered} \tag{2.4.7}$$

and $\{\arg(A_l)\}_{l\in[2]}$, $\{\arg(B_{lk})\}_{l\in[4],2\le k\le 6}$ chosen randomly.

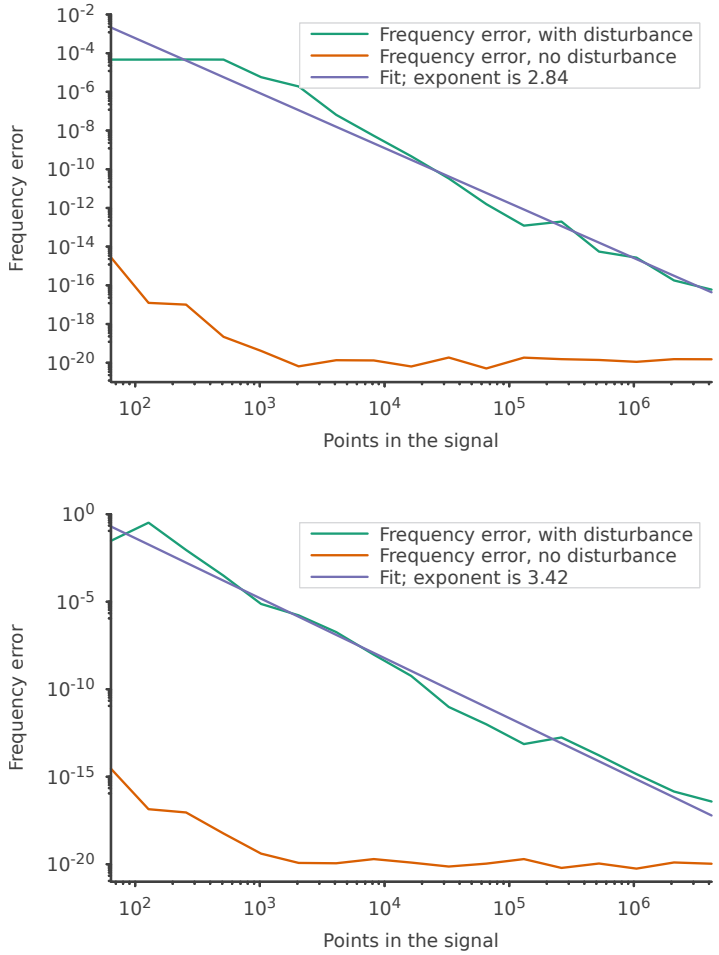

Figure 2.5: Errors for the two frequency algorithm. Top and bottom are the first and second frequencies, respectively. The plots are in logarithmic scale, and the fitted function is $y = C/x^{\alpha}$, with α shown in the legend.

We see that the error behaves as theoretically predicted.

We now study how this algorithm compares to the others. We use a signal like that in Eq. (2.4.6) for fixed $N = 2^{16} = 65536$ (a realistic case) and decreasing $|\nu_2 - \nu_1|$. Since we want the points equally spaced in log scale, for the mth iteration it is

$$\nu_2 = \nu_1 + 2^{-m/2} = 0.163618 + 2^{-m/2}, \qquad m \in [50] \tag{2.4.8}$$

Also, since we want the first frequency to always be ν_1 (remember that in our definition the first frequency is the one with the largest amplitude), we used

$$\begin{aligned} |A_{1k}| &= 10\,|B_{1k}| = 1 \\ |A_{2k}| &= 10\,|B_{2k}| = 9/10, \end{aligned} \qquad 2 \leq k \leq 6 \tag{2.4.9}$$

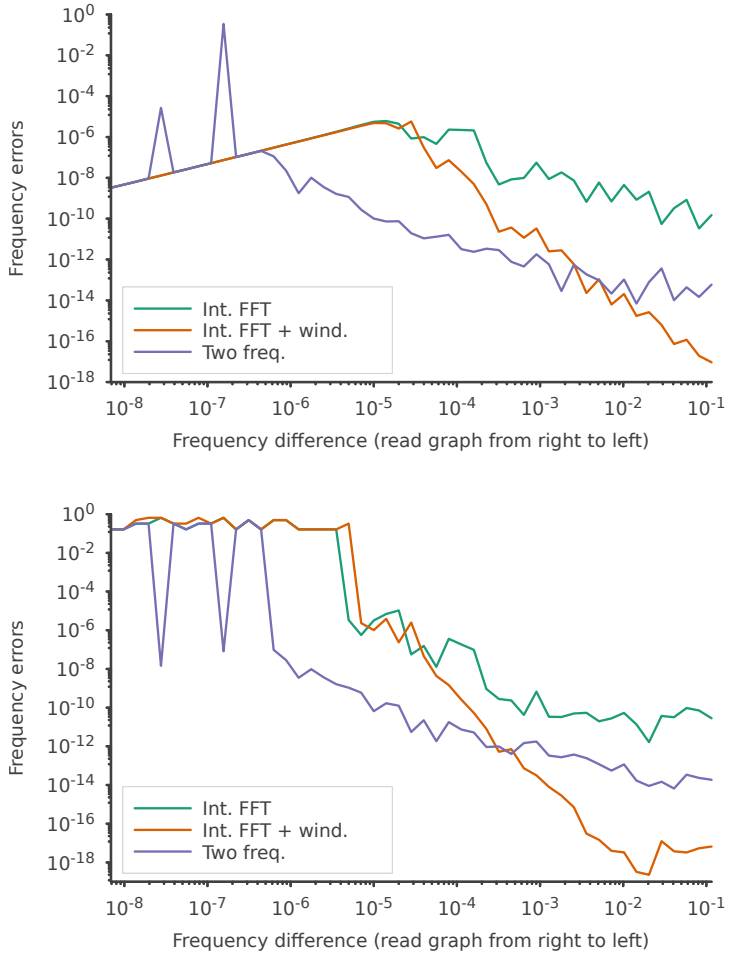

Figure 2.6: Comparison of errors of the Interpolated FFT, Interpolated FFT with windowing and Two-frequency algorithms. Top is the first frequency, bottom is the second frequency.

The Birkhoff Average algorithm was not studied because it performs badly when there are two or more harmonics with a similar amplitude. Several comments are in order.

First, we notice that the Interpolated FFT and Interpolated FFT with windowing algorithms have approximately the same precision for $|\Delta\nu| = |\nu_2 - \nu_1| < 10^{-5}$. This is because the resolution of the DFT, i.e. the smallest difference between two frequencies that are recognised as belonging to different bins, is $1/N = 1.5 \cdot 10^{-5}$. So for a smaller $|\Delta\nu|$, the two frequencies are recognised to be a single one of twice the amplitude. Since the difference between the algorithms is how they respond to the presence of additional harmonics, in that case they behave identically. Also, the error decreases as $|\Delta\nu|$ diminishes because the smaller $|\Delta\nu|$ is, the less influence it also has on the bins adjacent to the peak and the smaller the error results.

Second, we see that both the Interpolated FFT algorithms fail in determining the second frequency if $|\Delta\nu| < 10^{-5}$. This is expected, since the projection of u_{ν_2} on $u_{\overline{\nu}_1}$ is

significantly different from zero and the signal $z_1 = z - \phi(\overline{\nu}_1)u_{\overline{\nu}_1}$ has also lost a significant part of u_{ν_2} (see Section 1.4 for more details on this matter).

Third, the Two-frequency algorithm has erratic behaviour for $|\Delta\nu| < 10^{-6}$. This is because it sees the two frequencies as a single one and correctly deduces only one of the two, but sometimes it is the first one and sometimes the second one. Indeed, we see that the peaks in the first graph correspond exactly to the dips in the second graph. But why does this happen for $|\Delta\nu| < 10^{-6}$ and not for $|\Delta\nu| < 10^{-5}$ like the others? Our guess is that since the algorithm searches for two frequencies, it better captures the information contained in the bins adjacent to the peak.

Fourth, the Two-frequency algorithm is more precise than the other known algorithms in the range $10^{-6} < |\Delta\nu| < 10^{-3}$. That is indeed the range in which the disturbances caused by the second frequency significantly alter the DFT near the first peak.

Chapter 3

Varying-amplitude signals

This chapter deals with the algorithms used for the analysis of signals made of harmonics whose amplitudes vary with time. The signals studied are of the form

$$x(n) = E(n)\Re\left[\sum_{l=1}^{L} A_l \exp(2\pi i \nu_l n)\right], \qquad n \in [N] \tag{3.0.1}$$

where $\Re$ is the real part and E is a function $[N] \mapsto \mathbb{R}$ that describes the modulation of the amplitude. The constraints on the various quantities are similar to those of Eq. (2.0.1).

Our goal is to compute the envelope function E, in order to construct the normalised signal

$$\widetilde{x}(n) = \frac{x(n)}{E(n)} \tag{3.0.2}$$

and calculate the signal frequencies with the techniques of Chapter 2. The studies carried out are intended to replicate those reported in [6].

3.1 Theory

In this section, we will describe the theory of the Hilbert Transform and its usage in calculating the envelope of a continuous-time signal. Then we will derive the discrete-time equivalent of those formulae, that are the ones we are interested in. For a more complete description, see [2, 3]. The letters $\mathcal{H}$ and $\mathcal{F}$ will be used to denote respectively the Hilbert and Fourier Transforms.

3.1.1 Continuous-time signals

We consider a function $f = f(t) : \mathbb{R} \mapsto \mathbb{R}$, which can be thought of as a continuous-time signal. Assuming that the integral is well behaved (both at the singular point of the integrand and at $\pm\infty$) we define the Hilbert Transform of f as follows

$$(\mathcal{H}f)(t) \coloneqq \frac{1}{\pi} \int_{-\infty}^{+\infty} \frac{f(\tau)}{t-\tau}\, d\tau \tag{3.1.1}$$

so it is the convolution of f with the function $h(t) = 1/\pi t$. Then, by the convolution theorem, we have the following

$$(\mathcal{F}\mathcal{H}f)(\nu) = (\mathcal{F}h)(\nu)\,(\mathcal{F}f)(\nu) = -i\,\mathrm{sgn}(\nu)(\mathcal{F}f)(\nu) \tag{3.1.2}$$

where it was used the continuous Fourier Transform with frequency ν

$$(\mathcal{F}g)(\nu) \coloneqq \int_{-\infty}^{+\infty} g(t) e^{-2\pi i \nu t}\, dt \tag{3.1.3}$$

and the sign function

$$\mathrm{sgn}(\nu) \coloneqq \begin{cases} +1 & \nu > 0 \\ 0 & \nu = 0 \\ -1 & \nu < 0\,. \end{cases} \tag{3.1.4}$$

We note that the Hilbert Transform acts on the frequency components of f by shifting those with negative frequency by $\pi/2$ in the forward direction and those with negative frequency by $\pi/2$ in the backward direction. The component with null frequency is removed, which can also be thought of as a consequence of the fact that the Hilbert Transform of a constant is zero.

If we now consider the signal $g = f + i\,\mathcal{H}f$, we see that

$$(\mathcal{F}g)(\nu) = (\mathcal{F}f)(\nu) + i\,[-i\,\mathrm{sgn}(\nu)(\mathcal{F}f)(\nu)] = [1 + \mathrm{sgn}(\nu)]\,(\mathcal{F}f)(\nu) \tag{3.1.5}$$

so $\mathcal{F}g$ has no negative frequency components, and g is called an analytic signal. We have constructed a complex signal g from the real signal f, and this allows us to calculate its instantaneous amplitude and phase as

$$A(t) = |g(t)|, \qquad \varphi(t) = \arg\big[g(t)\big] \tag{3.1.6}$$

and we want to verify whether $A = E$ as, in this the positive, we have found a way to calculate the envelope of a generic real continuous-time signal.

To show this, we need Bedrosian's theorem, that states: given two signals f_1, f_2 such that exists some $\sigma \in]0, 1/2[$ with the property $F_1(\nu) = (\mathcal{F}f_1)(\nu) = 0$ for $|\nu| > \sigma$ and $F_2(\nu) = (\mathcal{F}f_2)(\nu) = 0$ for $|\nu| < \sigma$, then

$$\mathcal{H}(f_1 f_2) = f_1 \, \mathcal{H}f_2 \, . \tag{3.1.7}$$

Note that the conditions assumed imply that f_1 and f_2 are such that the first has only the low-frequency part of the Fourier spectrum, while the opposite holds for f_2. Therefore, there is no superposition between the Fourier spectra of the two functions.

To prove the theorem, we calculate

$$\begin{aligned} &(\mathcal{F}\mathcal{H}(f_1 f_2))(\nu) \\ &= -i \operatorname{sgn}(\nu) \int_{-\infty}^{+\infty} F_1(\nu - \mu) F_2(\mu) \, d\mu \end{aligned} \tag{3.1.8}$$

and we note that the integrand might be non-zero only in $I = [\nu - \sigma, \nu + \sigma] \bigcap (] - \infty, -\sigma] \bigcup [\sigma, +\infty[)$ where if $\nu < 0$ the first interval must be flipped.

By explicitly computing the two cases $\nu > 0$ and $\nu < 0$, we see that in both cases I is only one interval and holds $\operatorname{sgn}(\nu) = \operatorname{sgn}(\mu)$ $\forall \mu \in I$. Then the previous equation continues as

$$\begin{aligned} &= \int_I F_1(\nu - \mu) \left[-i \operatorname{sgn}(\mu) F_2(\mu) \right] \, d\mu \\ &= (\mathcal{F}(f_1 \, \mathcal{H}f_2))(\nu) \, , \end{aligned}$$

which becomes the desired result after applying $\mathcal{F}^{-1}$.

If f is an amplitude-modulated real signal

$$f(t) = E(t) \, \Re \left\{ \exp\big[i\varphi(t) \big] \right\} = E(t) \cos\big[\varphi(t)\big], \tag{3.1.9}$$

then Bedrosian's theorem implies

$$\begin{aligned} g(t) &= E(t) \cos\big[\varphi(t)\big] + iE(t) \mathcal{H} \left\{ \cos\big[\varphi(t)\big] \right\} \\ &= E(t) \exp\big[i\varphi(t) \big] \\ &\Longrightarrow \\ A(t) &= |g(t)| = E(t) \end{aligned} \tag{3.1.10}$$

if the envelope represents the low-frequency part, as it is in the cases we will consider in this thesis work.

3.1.2 Discrete-time signals

We now want to find the discrete-time equivalent of the formulae showed in the last section. As we will show next, the construction of said formulae is non-trivial.

The first problem that arises is due to the fact that the convolution is now performed on a finite domain: if we define

$$(x * y)(n) \coloneqq \sum_{m=1}^{N} x(m)y(n-m) \tag{3.1.11}$$

there are terms like $y(1-m),\ m \in [N]$ that are not defined. This problem can be easily solved if instead of the usual convolution, we do the *periodic* convolution, in which the signals x and y are periodically extended.

The second problem is finding the signal to convolve with. A natural choice might be

$$h(n) = \frac{1}{\pi n}$$

but this choice does not reproduce the property $\mathcal{H}^2 = -\mathcal{I}$ which can be derived from Eq. (3.1.1) for the continuous case (for example, see the calculations carried out in [3] for $x(n) = \sin(\alpha n),\ \alpha \in \mathbb{R}$). To find a sensible choice for the signal h, we employ the continuous Fourier Transform

$$\begin{aligned}(\mathcal{F}x)(\nu) &= \sum_{n=1}^{N} x(n)e^{2\pi i\nu n} \\ (\mathcal{F}^{-1}X)(t) &= \int_{-1/2}^{1/2} X(\nu)e^{2\pi i\nu t}\,d\nu\end{aligned} \tag{3.1.12}$$

with $\nu \in]-1/2, 1/2]$, since $\mathcal{F}f$ is periodic with period 1.

Proceeding as [3], by interpreting x as a piecewise constant function with support $]0, N]$ and intervals of length 1

$$\begin{aligned}(\mathcal{H}x)(n) &= \frac{1}{\pi}\int_{-\infty}^{+\infty} \frac{d\eta}{n-\eta}\int_{-1/2}^{1/2} X(\nu)e^{2\pi i\nu\eta}\,d\nu \\ &= \frac{1}{\pi}\int_{-1/2}^{1/2} X(\nu)\left[-\pi i\,\mathrm{sgn}(\nu)e^{2\pi i\nu n}\right]\,d\nu \\ &= -i\sum_{k=1}^{N} x(k)\int_{-1/2}^{1/2} e^{2\pi i\nu(n-k)}\,d\nu \\ &= \frac{1}{\pi}\sum_{k=1,\,k\neq n}^{N} x(k)\,\frac{1-(-1)^{n-k}}{n-k}\end{aligned} \tag{3.1.13}$$

so the other function in the convolution is

$$h(n) = \begin{cases} \dfrac{1-(-1)^n}{\pi n} & n \neq 0 \\ 0 & n = 0\,. \end{cases} \tag{3.1.14}$$

We can then rewrite the Hilbert Transform of a discrete-time, finite-length signal in a better known way

$$(\mathcal{H}x)(k) = \begin{cases} \dfrac{2}{\pi} \displaystyle\sum_{n \text{ odd}} \dfrac{x(n)}{k-n} & k \text{ even} \\ \dfrac{2}{\pi} \displaystyle\sum_{n \text{ even}} \dfrac{x(n)}{k-n} & k \text{ odd}\,. \end{cases} \tag{3.1.15}$$

The third and last problem is finding a discrete counterpart to Bedrosian's theorem. This can be done with the same calculations of Eq. (3.1.8), with a small change: the first integral should be in $\mu \in]-1/2, 1/2[$ and should be $I = [\nu-\sigma, \nu+\sigma] \bigcap (]-1/2, -\sigma] \bigcup [\sigma, +1/2[)$, which still lets us conclude $\operatorname{sgn}(\nu) = \operatorname{sgn}(\mu)\ \forall \mu \in I$.

An additional note that will be useful later on: from the last step of Eq. (3.1.13) we see that

$$\begin{aligned} h(n) &= \frac{1-(-1)^n}{\pi n} \\ &= \int_{-1/2}^{1/2} [-i \operatorname{sgn}(\nu)] e^{2\pi i \nu n}\, d\nu \\ &= \mathcal{F}^{-1}[-i \operatorname{sgn}(\nu)](n) \\ &\Longrightarrow \\ (\mathcal{F}h)(\nu) &= -i \operatorname{sgn}(\nu). \end{aligned} \tag{3.1.16}$$

Notice that this formula only works if we choose the domain of $\mathcal{F}x$ to be $]-1/2, 1/2]$ while in principle, due to its periodicity, we could choose it to be any interval of length 1. From Eq. (3.1.16), we see that if one wants $\nu \in]\nu_0, \nu_0+1]$, then $\operatorname{sgn}(\nu-\nu_0-1/2)$ must be used.

3.2 Simulations

We consider a signal of the form of Eq. (3.0.1) with $|A_1| = 1$, and we want to find ν_1. This can be done in two ways:

- We calculate E using the Hilbert Transform, then apply the Interpolated FFT (with or without windowing) algorithm to the normalised signal defined in Eq. (3.0.2).

- We just use the Birkhoff Average algorithm, since it only relies on the phases of the signal.

For the second method, we expect it to behave identically as in the constant-amplitude case. For this reason, we will focus on the first method. In particular, we remember that we found two different ways of calculating the Hilbert Transform of a discrete-time, finite-length signal: the first one being Eq. (3.1.15) and the second one being

$$\mathcal{H}x = \mathcal{F}^{-1}\left[-i\,\mathrm{sgn}\left(\frac{1}{2}-\frac{k}{N}\right)\mathcal{F}x\right] \tag{3.2.1}$$

from Eq. (3.1.16), after changing the range of the frequency from $]-1/2, 1/2]$ to $]0, 1]$.

The algorithms for the reconstruction of the envelope were tested on signals of the form

$$z(n) = E(n)\left[A_1\exp(2\pi i\nu_1 n) + \sum_{k=2}^{6} A_k \exp(2\pi i k\nu_1 n)\right] \tag{3.2.2}$$

with

$$\begin{gathered} E(n) = \exp\left\{\left[2.5088\,\sin\left(\frac{n\pi}{10^4}\right)\right]^2\right\} \\ A_1, A_k \in \mathbb{C}, \qquad |A_k| = |A_1|/10 = 1/10, \qquad 2 \le k \le 6 \\ 64 = 2^6 \le N \le 2^{20} = 1048576 \\ \nu_1 = 0.163618 \end{gathered} \tag{3.2.3}$$

where E is the same as in [6].

We study now the performance of the two methods of the Hilbert Transform in reconstructing the envelope. We consider the signal of Eq. (3.2.2) and calculate its Hilbert Transform with the two methods mentioned previously, and the reconstructed envelope is

$$\overline{E}^2(n) = x^2(n) + (\mathcal{H}x)^2(n) \tag{3.2.4}$$

We notice that the signal in Eq. (3.2.2) is not in the form of the one in Eq. (3.1.10) due to the presence of the disturbing harmonics $\{u_{k\nu_1}\}_{2\le k\le 6}$. Indeed we see that

$$\begin{aligned} \overline{E}^2(n) &= E^2(n)\sum_{k,h=1}^{6} A_k A_h^* \exp\big[2\pi i n(\nu_k-\nu_h)\big] \\ &= E^2(n)\left\{\sum_{k=1}^{M}|A_k|^2 + 2\sum_{k=1,h>k}^{M} A_k A_h^* \cos\big[2\pi n(\nu_k-\nu_h)\big]\right\}, \end{aligned} \tag{3.2.5}$$

and the normalised signal is

$$\widetilde{x}(n) = \frac{\displaystyle\sum_{k=1}^{6} \big\{ \Re(A_k) \cos\big[2\pi n \nu_j\big] - \Im(A_k) \sin\big[2\pi n \nu_j\big] \big\}}{\sqrt{1 + \displaystyle\sum_{k=2}^{6} |A_k|^2 + 2 \sum_{k=1, h>k}^{M} A_k A_h^* \cos\big[2\pi n (\nu_k - \nu_h)\big]}} \,. \tag{3.2.6}$$

Remembering that the $\{A_k\}_{2 \le k \le 6}$ are in general small (in our simulations we chose $|A_1|/|A_k| = 10$), we can expand the denominator as

$$(1+z)^{-1/2} = 1 - z/2 + \mathcal{O}(|z|^2), \qquad |z| \ll 1. \tag{3.2.7}$$

Then we see that the spectrum of the signal includes other frequencies other than ν_1 and $\{k\nu_1\}_{2 \le k \le 6}$, and their amplitude of has the order of magnitude of $|A_k|$. So the DFT still has a peak for the frequency ν_1, and the algorithms of Section 2 can still be applied.

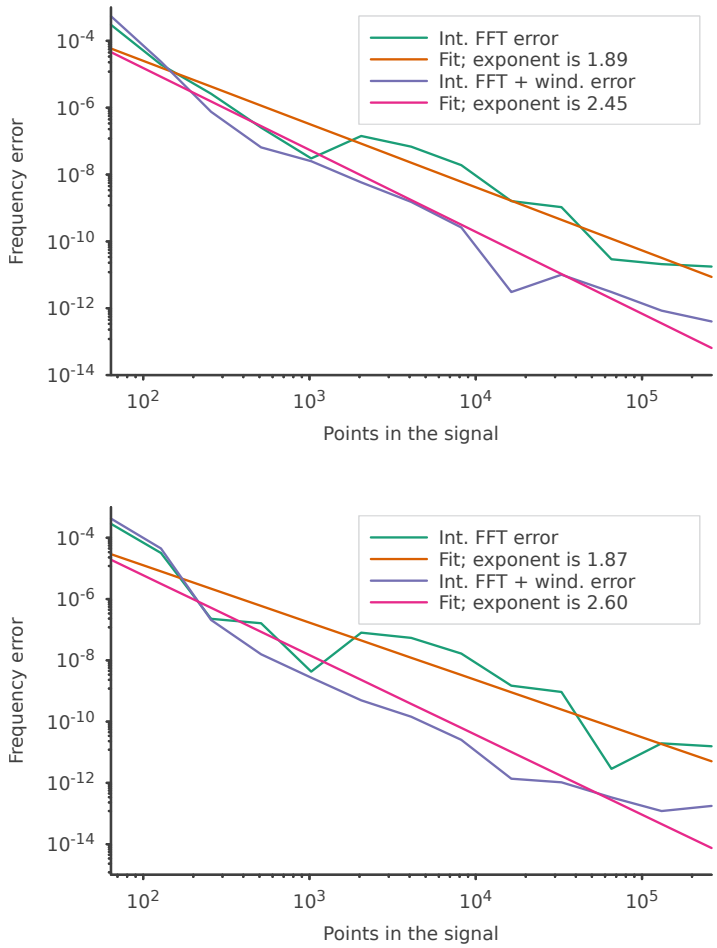

Figure 3.1: Errors for the the Interpolated FFT an Interpolated FFT with windowing algorithms, when the signal is normalised using different methods for calculating the Hilbert Transform. Top is the Fourier Transform method, bottom is the convolution method.

We see that the two methods have essentially the same performance, for both the Interpolated FFT and Interpolated FFT with windowing algorithms. We conclude that the convolution method is not an improvement over the widely used Fourier Transform method. Furthermore we note that, as observed in [6] the Hilbert Transform does not recover the performance of the Interpolated FFT with windowing algorithm.

Chapter 4

Description of the code

4.1 Structure of the program and programming choices

The code developed in this thesis work in divided into two parts:

- The main part, contains the algorithms used to calculate the main frequencies of a signal.
- The optional part, contains only the `FitPlot` component, where are implemented the functions used to plot the performance of the algorithms and find their power law.

The optional part in built on top of the main part, but not vice versa. This means that, if desired, a user can include only the main part by adjusting the `CMakeLists.txt` file.

The required dependencies are

- The library `fftw3`, for the implementation of fast and efficient FFT algorithms; For convenience of implementation, we use `fftw3l`, its `long double` variant.
- The library `sciplot`, a C++ wrapper for `gnuplot`, used for plotting. It is only used in the optional part.

A brief overview of the main part

The main part inclused a few component, whose interfaces are implemented in the following header files:

- `types.hpp` is the only header without a source file. It contains the definitions

of the constants used, and of all types (often using type aliases to have better customizability).

- `signal.hpp` contains the declaration of all the functions used to work with the signals.
- `tuneCalculator.hpp` contains the definitions of all the algorithms used to calculate the main frequencies. The functions defined here are the only ones that are meant to be called by the user.

In the above list, each header includes all of the previous ones.

What to include

`hats.hpp` is a convenience header file that includes all the header files of the main part. If the optional part is also desired, include also `fitPlot.hpp`

Floating point numbers and machine precision

The type aliases used in `types.hpp` allow, for example, to change the precision of the floating point numbers used. This is done by simply changing the definition of the type alias `FPType`. Testing showed that sometimes errors could reach 10^{-18}, and for this reason, `FPType` = `long double` is the default (and suggested) option.

Almost all the functions used are from the C++ standard library, so they support the `float`, `double` and `long double` types. Then, if a floating point type smaller than `long double` is used, memory usage and computation time are reduced.

An exception is the function `FFT`, which is a wrapper for the homonymous function from the `fftw3` library. For ease of installation, it is suggested to always install the `fftw3l` library and lose some efficiency in the computation, since with this setup the `FFT` function from `fftw3` will always work with long double types. This means that `FFT` always internally converts floating point variables to `long double` type, and at the end converts the result into the `FPType` type.

It is left open the possibility of introducing new types, for example the 128-bit floating point type from the `boost` library. However, it is expected that the functions of the C++ standard library will not work with this new type, requiring changes to the code.

Templates

Since it was often necessary to pass a function as an argument of another function (for example, if one wants to calculate the first M frequencies using a tune calculator of

choice), templates were largely used in the program. For this reason, `.inl` files were used, since a template must be defined in the header. An `.inl` file allows us to separate the definition from the declaration, resulting in better readability of the headers.

Each function is briefly described in the header by a few lines of comments. Modern editors usually show said comments, by hovering with the mouse onto a function typed in the user's program.

To ensure that the template parameters (that is, the functions passed) behave as expected, static asserts are used in every template. They confirm at compilation time that the signature of the function is as expected.

Implementation of other orthonormalisation algorithms

See Section 1.3 for a theoretical overview of the topic.

The change in the orthonormalisation strategy can be implemented simply by changing the behaviour of the function `Orthonormalise` in `signal.cpp`. If one prefers to define a new function `OrthonormaliseQR`, then also the function `MultipleFrequencies` should be changed, as it is the only function of the program that calls `Orthonormalise`. In the future, `MultipleFrequencies` might be made a template that accepts as a parameter the function that defines the orthonormalisation strategy.

For maintenance reasons, it is advised to respect the signature[1] of the function `Orthonormalise`. The details on the interface are given in Section 4.3.

4.2 Developement and testing

The code was developed and run in a `DevContainer` provided by Docker, with `AlmaLinux 9` as platform.

The functions were tested with `doctest`. Tests were written for each function and are run automatically at each commit thanks to GitLab's Continuous Integration feature. The algorithms that calculate the main frequencies were tested on synthetic signals like the ones used in the simulations, e.g. those of Eqs. (2.0.5) and (2.0.7). Since generating signals involves random numbers, the first 100 numbers generated are saved in order check if the generators are implemented in the same way in different machines.

[1] The parameter and return types of the function

4.3 Interface

We give here an overview of the interface of the code and some useful additional information.
The most important type aliases used are:

- `FPType` is a floating point type. As mentioned in section 4.1, it supports the C++ native floating point types, and the user is encouraged to change it if desired. Aliases for it are `Frequency`, `AmplitudeModulus` and `Phase`.
- `Complex` is `std::complex<FPType>`. An alias for it is `Amplitude`.
- `ComplexSignal` is `std::vector<Complex>`.
- `SignalSize` is the largest `unsigned int` type available in C++, implemented as `ComplexSignal::size_type`.

It is left open the possibility of converting some type aliases (like `Frequency` or `Amplitude`) into structs.
The generic tune calculator is implemented as

```
Frequency TuneCalculator(ComplexSignal const&)
```

except for `BirkhoffAverage` and `TwoFrequency`, which are

```
Frequency BirkhoffAverage(ComplexSignal const&, Exponent = 1)
std::pair<Frequency, Frequency> TwoFrequency(ComplexSignal const&)
```

where the parameter of `Exponent` type has a default value of 1.
The function that calculates multiple frequencies is implemented as

```
template <class TuneCalculator>
std::vector<Frequency> MultipleFrequencies(ComplexSignal const&,
    int, TuneCalculator const&)
```

where the parameter of `int` type is the number of frequencies to calculate. It is left open the possibility of defining different stop conditions for this function, perhaps by replacing the `int` with another template parameter.
The function that orthonormalises a vector with respect to an already orthonormal set is

```
void Orthonormalise(std::vector<ComplexSignal> const&,
    ComplexSignal &)
```

and, as mentioned in Sections 1.3 and 4.1, it is left open the possibility of implementing alternative algorithms.

Conclusions

We reviewed the algorithms used for finding the main frequency of a time series and implemented the said algorithms correctly in a C++ program. A slight discrepancy with the results of other simulations was found in the case of an amplitude-modulated signal for the Interpolated FFT with windowing algorithm.

The new two-frequency algorithm was studied and implemented and it was found that if $10^{-6} < |\nu_2 - \nu_1| < 10^{-3}$, where ν_1 and ν_2 are the two most dominant frequencies of the signal, it represents an improvement over the currently known algorithms.

Bibliography

[1] R. Bartolini et al. "Tune Evaluation in Simulations and Experiments". In: *Particle Accelerators Vol. 52 147-177* (1996).

[2] Stefan L. Hahn. *The Transforms and Applications Handbook: Second Edition*. CRC Press, 2000.

[3] Frederick W. King. *Hilbert Tranforms*. Cambridge Univeristy Press, 2009.

[4] Thomas Trogdon. *The Modified Gram-Schmidt Procedure*. Link; accessed 05 Oct. '23. 2016.

[5] James V. Lambers. *Numerical Linear Algebra*. Link; accessed 05 Oct. '23. 2017.

[6] M. Giovannozzi, G. Franchetti, and G. Russo. "New Techniques to Compute the Linear Tune". In: *JACoW IPAC'21 4142-4145* (2021).

Appendix A

Proof of the two-frequency formulas

Notation

$$\begin{aligned}
&\phi_m = \phi(\nu_{j_m}), \qquad m \in \{1,2\} \\
&\phi_{m\pm} = \phi(\nu_{j_m \pm 1}) e^{\pm \frac{\pi i}{N}}, \qquad m \in \{1,2\} \\
&\Delta = j_2 - j_1 \\
&\Delta_{k j_m} = \nu_k - \nu_{j_m}, \qquad k, m \in \{1,2\}
\end{aligned} \tag{A.0.1}$$

$$\begin{aligned}
\alpha_k &= \frac{\phi_{1+} \sin\left[\pi(\nu_k - \nu_{j_1} - \frac{1}{N})\right] - \phi_1 \sin\left[\pi(\nu_k - \nu_{j_1})\right]}{\phi_{1-} \sin\left[\pi(\nu_k - \nu_{j_1} + \frac{1}{N})\right] - \phi_1 \sin\left[\pi(\nu_k - \nu_{j_1})\right]} \\
\beta_k &= \frac{\phi_{2+} \sin\left[\pi(\nu_k - \nu_{j_2} - \frac{1}{N})\right] - \phi_2 \sin\left[\pi(\nu_k - \nu_{j_2})\right]}{\phi_{2-} \sin\left[\pi(\nu_k - \nu_{j_2} + \frac{1}{N})\right] - \phi_2 \sin\left[\pi(\nu_k - \nu_{j_2})\right]}
\end{aligned} \tag{A.0.2}$$

Notice that in α_k and β_k the only unknown is ν_k.

Calculations

Given a signal in the form

$$z(n) = A_1 \exp(2\pi i \nu_1 n) + A_2 \exp(2\pi i \nu_2 n), \qquad n \in [N] \tag{A.0.3}$$

and two indexes $j_1, j_2 \in [N]$, we consider the projection of z onto the harmonics of frequencies $\nu_{j_1} = j_1/N$, $\nu_{j_2} = j_2/N$, $\nu_{j_1 \pm 1} = \nu_{j_1} \pm 1/N$, $\nu_{j_2 \pm 1} = \nu_{j_2} \pm 1/N$.

The j_mth element of the DFT is

$$\phi(\nu_{j_m}) = \sum_{k=1}^{2} \left\{ \frac{A_k}{N} \sum_{n=1}^{N} \left[\exp\{2\pi i(\nu_k - \nu_{j_m})\}\right]^n \right\} \tag{A.0.4}$$

and similarly for $\phi(\nu_{j_m \pm 1})$.

The kth addend of $\phi(\nu_{j_m})$, multiplied by N/A_k, is

$$\begin{aligned} &\sum_{n=1}^{N} \left[\exp\{2\pi i \Delta_{kj_m}\}\right]^n = \frac{e^{2\pi i \Delta_{kjm}(N+1)} - e^{2\pi i \Delta_{kjm}}}{e^{2\pi i \Delta_{kjm}} - 1} \\ &= e^{\pi i(\nu_k - \nu_{jm})(N+1)} \frac{\sin[N\pi(\nu_k - \nu_{j_m})]}{\sin\left[\pi(\nu_k - \nu_{j_m})\right]} . \end{aligned} \tag{A.0.5}$$

The kth addend of $\phi(\nu_{j_m \pm 1})$, multiplied by N/A_k, is

$$\begin{aligned} &\sum_{n=1}^{N} \left[\exp\{2\pi i(\Delta_{kj_m} \mp 1/N)\}\right]^n = e^{2\pi i(\Delta_{kj_m} \mp \frac{1}{N})} \frac{e^{2\pi i(\Delta_{kj_m} \mp \frac{1}{N})N} - 1}{e^{2i(\pi \Delta_{kj_m} \mp \frac{\pi}{N})} - 1} = \\ &= e^{\pi i(\Delta_{kj_m} \mp \frac{1}{N})} e^{iN\pi\Delta_{kj_m}} \frac{\sin(N\pi\Delta_{kj_m})}{\sin(\pi\Delta_{kj_m} \mp \frac{\pi}{N})} = e^{\pi i \Delta_{kj_m}(N+1)} e^{\mp \frac{\pi i}{N}} \frac{\sin(N\pi\Delta_{kj_m})}{\sin(\pi\Delta_{kj_m} \mp \frac{\pi}{N})} = \\ &= e^{\pi i(\nu_k - \nu_{jm})(N+1)} e^{\mp \frac{\pi i}{N}} \frac{\sin[N\pi(\nu_k - \nu_{j_m})]}{\sin\left[\pi(\nu_k - \nu_{j_m}) \mp \frac{\pi}{N}\right]} . \end{aligned} \tag{A.0.6}$$

Given that (h is the index in $\{1, 2\}$ that is not k)

$$\begin{aligned} &\phi_{m\pm} \sin\left[\pi(\nu_k - \nu_{j_m \pm 1})\right] - \phi_m \sin\left[\pi(\nu_k - \nu_{j_m})\right] = \\ &= \frac{A_k}{N} e^{\pi i(\nu_k - \nu_{jm})(N+1)} \sin[N\pi(\nu_k - \nu_{j_m})] \\ &+ \frac{A_h}{N} e^{\pi i(\nu_h - \nu_{jm})(N+1)} \sin[N\pi(\nu_h - \nu_{j_m})] \frac{\sin\left[\pi(\nu_k - \nu_{j_m}) \mp \frac{\pi}{N}\right]}{\sin\left[\pi(\nu_h - \nu_{j_m}) \mp \frac{\pi}{N}\right]} \\ &- \frac{A_k}{N} e^{\pi i(\nu_k - \nu_{jm})(N+1)} \sin[N\pi(\nu_k - \nu_{j_m})] \\ &- \frac{A_h}{N} e^{\pi i(\nu_h - \nu_{jm})(N+1)} \sin[N\pi(\nu_h - \nu_{j_m})] \frac{\sin\left[\pi(\nu_k - \nu_{j_m})\right]}{\sin\left[\pi(\nu_h - \nu_{j_m})\right]} = \\ &= \frac{A_h}{N} e^{\pi i(\nu_h - \nu_{jm})(N+1)} \sin[N\pi(\nu_h - \nu_{j_m})] \left[\frac{\sin\left[\pi(\nu_k - \nu_{j_m}) \mp \frac{\pi}{N}\right]}{\sin\left[\pi(\nu_h - \nu_{j_m}) \mp \frac{\pi}{N}\right]} - \frac{\sin\left[\pi(\nu_k - \nu_{j_m})\right]}{\sin\left[\pi(\nu_h - \nu_{j_m})\right]} \right] \end{aligned} \tag{A.0.7}$$

we obtain

$$\alpha_k = \frac{\phi_{1+}\sin\left[\pi(\nu_k-\nu_{j_1})-\frac{\pi}{N}\right]-\phi_1\sin\left[\pi(\nu_k-\nu_{j_1})\right]}{\phi_{1-}\sin\left[\pi(\nu_k-\nu_{j_1})+\frac{\pi}{N}\right]-\phi_1\sin\left[\pi(\nu_k-\nu_{j_1})\right]} =$$
$$\left\{\frac{\sin\left[\pi(\nu_k-\nu_{j_1})-\frac{\pi}{N}\right]}{\sin\left[\pi(\nu_h-\nu_{j_1})-\frac{\pi}{N}\right]}-\frac{\sin\left[\pi(\nu_k-\nu_{j_1})\right]}{\sin\left[\pi(\nu_h-\nu_{j_1})\right]}\right]\Big/\left[\frac{\sin\left[\pi(\nu_k-\nu_{j_1})+\frac{\pi}{N}\right]}{\sin\left[\pi(\nu_h-\nu_{j_1})+\frac{\pi}{N}\right]}-\frac{\sin\left[\pi(\nu_k-\nu_{j_1})\right]}{\sin\left[\pi(\nu_h-\nu_{j_1})\right]}\right\}$$
$$= \frac{1/\sin\left[\pi(\nu_h-\nu_{j_1})-\frac{\pi}{N}\right]}{1/\sin\left[\pi(\nu_h-\nu_{j_1})+\frac{\pi}{N}\right]}\cdot$$
$$\frac{-\cos[\pi(\nu_k-\nu_{j_1})]\sin\left[\pi(\nu_h-\nu_{j_1})\right]\cos\frac{\pi}{N}+\sin\left[\pi(\nu_k-\nu_{j_1})\right]\cos[\pi(\nu_h-\nu_{j_1})\big]\sin\frac{\pi}{N}}{\cos[\pi(\nu_k-\nu_{j_1})]\sin\left[\pi(\nu_h-\nu_{j_1})\right]\cos\frac{\pi}{N}-\sin\left[\pi(\nu_k-\nu_{j_1})\right]\cos[\pi(\nu_h-\nu_{j_1})]\sin\frac{\pi}{N}} =$$
$$= -\frac{\sin\left[\pi(\nu_h-\nu_{j_1})+\frac{\pi}{N}\right]}{\sin\left[\pi(\nu_h-\nu_{j_1})-\frac{\pi}{N}\right]} \tag{A.0.8}$$

and, with the same reasoning,

$$\beta_k = -\frac{\sin\left[\pi(\nu_k-\nu_{j_1})+\frac{\pi(1-\Delta)}{N}\right]}{\sin\left[\pi(\nu_k-\nu_{j_1})-\frac{\pi(1+\Delta)}{N}\right]}\,. \tag{A.0.9}$$

From Eqs. (A.0.8) and (A.0.9) we obtain, by clearing the denominators and expanding the sines,

$$\frac{(\alpha_k-1)\sin\frac{\pi}{N}}{(\alpha_k+1)\cos\frac{\pi}{N}} = \tan(\pi\Delta_{hj_1}) = \frac{\beta_k\sin\frac{\pi(1+\Delta)}{N}-\sin\frac{\pi(1-\Delta)}{N}}{\beta_k\cos\frac{\pi(1+\Delta)}{N}+\cos\frac{\pi(1-\Delta)}{N}}\,. \tag{A.0.10}$$

Then, by equating the first and third member and clearing denominators, we obtain

$$\alpha_k\beta_k\sin\frac{\pi\Delta}{N}+\alpha_k\sin\frac{\pi(\Delta-2)}{N}+\beta_k\sin\frac{\pi(\Delta+2)}{N}+\sin\frac{\pi\Delta}{N}=0\,. \tag{A.0.11}$$

We note that both the numerators and the denominators of α_k and β_k are linear functions of $\sin\left[\pi(\nu_k-\nu_{j_1})\right]$ and $\cos\left[\pi(\nu_k-\nu_{j_1})\right]$. This means that if we clear the denominators, the left hand side of Eq. (A.0.11) is a weighted sum of $\sin^2\left[\pi(\nu_k-\nu_{j_1})\right]$, $\sin\left[\pi(\nu_k-\nu_{j_1})\right]\cos\left[\pi(\nu_k-\nu_{j_1})\right]$ and $\cos^2\left[\pi(\nu_k-\nu_{j_1})\right]$ that can be rewritten as

$$\begin{aligned}&\gamma_2\sin^2\left[\pi(\nu_k-\nu_{j_1})\right]+\gamma_1\sin\left[\pi(\nu_k-\nu_{j_1})\right]\cos\left[\pi(\nu_k-\nu_{j_1})\right]\\&+\gamma_0\cos^2\left[\pi(\nu_k-\nu_{j_1})\right]=0\,,\end{aligned} \tag{A.0.12}$$

and dividing by $\cos^2[\pi(\nu_k - \nu_{j_1})]$ we get

$$\gamma_2 \tan^2 \pi(\nu_k - \nu_{j_1}) + \gamma_1 \tan \pi(\nu_k - \nu_{j_1}) + \gamma_0 = 0 \tag{A.0.13}$$

which can be solved for the unknown ν_k.

After some calculations starting from Eq. (A.0.11), the γ coefficients are found to be

$$\begin{aligned}
\gamma_2 = {} & \phi(\nu_{j_1+1})\phi(\nu_{j_2+1}) e^{\frac{2\pi i}{N}} \cos\frac{\pi}{N} \sin\frac{\pi\Delta}{N} \cos\frac{\pi(\Delta+1)}{N} \\
& - \phi(\nu_{j_1+1})\phi(\nu_{j_2}) e^{\frac{\pi i}{N}} 2\cos^2\frac{\pi}{N} \cos\frac{\pi\Delta}{N} \sin\frac{\pi(\Delta-1)}{N} \\
& + \phi(\nu_{j_1+1})\phi(\nu_{j_2-1}) \cos\frac{\pi}{N} \cos\frac{\pi(\Delta-1)}{N} \sin\frac{\pi(\Delta-2)}{N} \\
& - \phi(\nu_{j_1})\phi(\nu_{j_2+1}) e^{\frac{\pi i}{N}} \cos\frac{\pi}{N} \sin\frac{2\pi(\Delta+1)}{N} \\
& + \phi(\nu_{j_1})\phi(\nu_{j_2}) 2\cos^2\frac{\pi}{N} \sin\frac{2\pi\Delta}{N} \\
& - \phi(\nu_{j_1})\phi(\nu_{j_2-1}) e^{-\frac{\pi i}{N}} \cos\frac{\pi}{N} \sin\frac{2\pi(\Delta-1)}{N} \\
& + \phi(\nu_{j_1-1})\phi(\nu_{j_2+1}) \cos\frac{\pi}{N} \cos\frac{\pi(\Delta+1)}{N} \sin\frac{\pi(\Delta+2)}{N} \\
& - \phi(\nu_{j_1-1})\phi(\nu_{j_2}) e^{-\frac{\pi i}{N}} 2\cos^2\frac{\pi}{N} \cos\frac{\pi\Delta}{N} \sin\frac{\pi(\Delta+1)}{N} \\
& + \phi(\nu_{j_1-1})\phi(\nu_{j_2-1}) e^{-\frac{2\pi i}{N}} \cos\frac{\pi}{N} \sin\frac{\pi\Delta}{N} \cos\frac{\pi(\Delta-1)}{N}
\end{aligned}$$

$$\begin{aligned}
\gamma_1 = {} & -\phi(\nu_{j_1+1})\phi(\nu_{j_2+1}) e^{\frac{2\pi i}{N}} \sin\frac{\pi\Delta}{N} \sin\frac{\pi(\Delta+2)}{N} \\
& + \phi(\nu_{j_1+1})\phi(\nu_{j_2}) e^{\frac{\pi i}{N}} 2\cos^2\frac{\pi}{N} 2\cos\frac{\pi}{N} \sin\frac{\pi(\Delta+1)}{N} \sin\frac{\pi(\Delta-1)}{N}, \\
& - \phi(\nu_{j_1+1})\phi(\nu_{j_2-1}) \sin\frac{\pi\Delta}{N} \sin\frac{\pi(\Delta-2)}{N} \\
& + \phi(\nu_{j_1})\phi(\nu_{j_2+1}) e^{\frac{\pi i}{N}} 2\cos\frac{\pi}{N} \sin^2\frac{\pi(\Delta+1)}{N} \\
& - \phi(\nu_{j_1})\phi(\nu_{j_2}) 4\cos^2\frac{\pi}{N} \sin^2\frac{\pi\Delta}{N} \\
& + \phi(\nu_{j_1})\phi(\nu_{j_2-1}) e^{-\frac{\pi i}{N}} 2\cos\frac{\pi}{N} \sin^2\frac{\pi(\Delta-1)}{N} \\
& - \phi(\nu_{j_1-1})\phi(\nu_{j_2+1}) \sin\frac{\pi\Delta}{N} \sin\frac{\pi(\Delta+2)}{N} \\
& + \phi(\nu_{j_1-1})\phi(\nu_{j_2}) e^{-\frac{\pi i}{N}} 2\cos\frac{\pi}{N} \sin\frac{\pi(\Delta+1)}{N} \sin\frac{\pi(\Delta-1)}{N}
\end{aligned}$$

$$
\begin{aligned}
&- \phi(\nu_{j_1-1})\phi(\nu_{j_2-1})e^{-\frac{2\pi i}{N}} \sin\frac{\pi\Delta}{N}\sin\frac{\pi(\Delta-2)}{N},\\
\\
\gamma_0 = &\phi(\nu_{j_1+1})\phi(\nu_{j_2+1})e^{\frac{2\pi i}{N}} \sin\frac{\pi}{N}\sin\frac{\pi\Delta}{N}\sin\frac{\pi(\Delta+1)}{N}\\
&- \phi(\nu_{j_1+1})\phi(\nu_{j_2})e^{\frac{\pi i}{N}} \sin\frac{2\pi}{N}\sin\frac{\pi\Delta}{N}\sin\frac{\pi(\Delta-1)}{N}\\
&+ \phi(\nu_{j_1+1})\phi(\nu_{j_2-1}) \sin\frac{\pi}{N}\sin\frac{\pi(\Delta-1)}{N}\sin\frac{\pi(\Delta-2)}{N}\\
&- \phi(\nu_{j_1-1})\phi(\nu_{j_2+1}) \sin\frac{\pi}{N}\sin\frac{\pi(\Delta+1)}{N}\sin\frac{\pi(\Delta+2)}{N}\\
&+ \phi(\nu_{j_1-1})\phi(\nu_{j_2})e^{-\frac{\pi i}{N}} \sin\frac{2\pi}{N}\sin\frac{\pi\Delta}{N}\sin\frac{\pi(\Delta+1)}{N}\\
&- \phi(\nu_{j_1-1})\phi(\nu_{j_2-1})e^{-\frac{2\pi i}{N}} \sin\frac{\pi}{N}\sin\frac{\pi\Delta}{N}\sin\frac{\pi(\Delta-1)}{N}.
\end{aligned}
\tag{A.0.15}
$$

Now, since these coefficients can be calculated from the Fourier transform of the signal, we have found a new way to calculate ν_1 and ν_2: we solve Eq. (A.0.13) with complex unknown ν, the two solutions are the two frequencies of our interest. We expect their imaginary part to be approximately equal to the machine error, so we can treat the two solutions as if they were real. Then, our best guess for ν_1 and ν_2 are respectively the smallest and the largest of the two, since without loss of generality $\nu_1 < \nu_2$.