

SMS Computer Lab: Signal Processing

Herrera Arias, Manuel Andres¹

¹Département de Physique,
Faculté des Sciences d'Orsay,
Université Paris-Saclay

Abstract

This report explores the fundamental principles and applications of Digital Signal Processing (DSP) through a series of computational exercises implemented in Python. We analyse the properties of the Discrete Fourier Transform (DFT), focusing on critical phenomena such as aliasing, spectral leakage, and the effects of windowing functions. Practical applications include the design of RC and RLC filters for audio noise reduction and the implementation of a Windowed Fourier Transform (WFT) for musical notes detection. Furthermore, a spectral manipulation algorithm was developed for voice identification and pitch shifting, demonstrating the efficiency of frequency-domain processing in altering vocal timbre. The results confirm the theoretical expectations of sampling theory and illustrate the versatility of spectral methods in analysing complex non-stationary signals.

I. INTRODUCTION

The purpose of this report is to document and analyse the results obtained from the three sessions of the Signal Processing Computer Laboratory. These sessions were designed to provide practical, hands-on experience in the analysis, manipulation, and visualisation of digital signals using computational tools.

The core objective of the lab work was to translate fundamental theoretical concepts in digital signal processing (DSP), such as sampling theory and Fast Fourier Transform (FFT), into practical computational algorithms. All exercises were executed using Python, leveraging the powerful capabilities of the `NumPy` library for array manipulation and efficient numerical computation, and `Matplotlib` for spectral and time-domain visualisation.

This document presents the methodology, key results, and analysis for the six distinct programming exercises completed. By implementing these exercises, we aimed to solidify the understanding of critical DSP phenomena, including the effects of sampling rate on spectral content (aliasing) and the practical application of the FFT for frequency-domain analysis.

Additionally, all source codes, high-resolution plots, and processed audio files described in this work are available for review in the following GitHub repository [1].

II. WARMUP

To initialise the computational workflow and gain proficiency with the `NumPy` and `Matplotlib` libraries, a series of introductory exercises were performed. These tasks focused on function visualisation, linear algebra solvers, and basic numerical calculus.

A. Polynomial Function Visualization

First, the polynomial function $f(x) = ax^3 + bx^2 + cx + d$ was evaluated for two distinct sets of coefficients:

- (i) $a = 0.4, b = -3.0, c = 2.0, d = 10.0$
- (ii) $a = -0.3, b = 1.0, c = 2.0, d = 2.0$

The functions were plotted over the interval $x \in [-2, 6]$ as shown in Fig. 1. This exercise allowed us to verify the correct implementation of vectorised operations in Python.

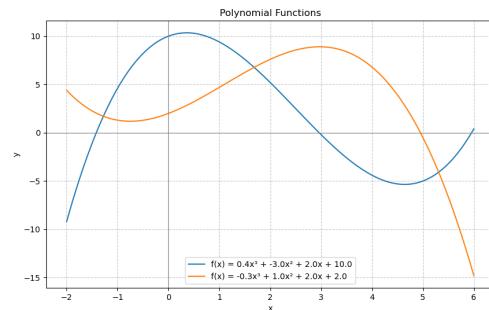


FIG. 1: Comparison of the two polynomial functions over the specified range.

B. Linear Systems of Equations

The second task involved solving a 4×4 system of linear equations using the `np.linalg.solve(A, b)` function, which utilises LU decomposition for efficient computation. The system is defined as follows:

$$\begin{pmatrix} 1 & 2 & 3 & 4 \\ 135 & 88 & 37 & 13 \\ -47 & 22 & -11 & 8 \\ 7 & 15 & -31 & -43 \end{pmatrix} \begin{pmatrix} x \\ y \\ z \\ t \end{pmatrix} = \begin{pmatrix} 22 \\ 337 \\ 332 \\ 12 \end{pmatrix} \quad (1)$$

The numerical solution obtained for the unknowns was:

$$\mathbf{X} = [x, y, z, t] = [-5, 9, 7, -3] \quad (2)$$

C. Numerical Differentiation

Finally, a numerical computation of the derivative df/dx was implemented using the Central Difference Method:

$$f'(x) \approx \frac{f(x+h) - f(x-h)}{2h} \quad (3)$$

where h represents the step size. This numerical result was compared against the analytical derivative $f'(x) = 3ax^2 + 2bx + c$. Both the function and its corresponding derivative are presented in Figure 2, demonstrating high accuracy between the numerical approximation and the theoretical expectation.

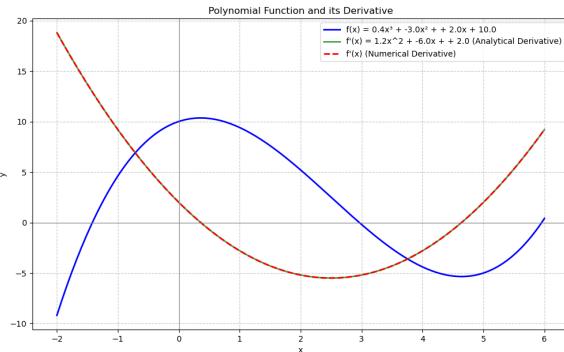


FIG. 2: Numerical vs. Analytical derivative of the polynomial function.

III. BASIC FOURIER PROPERTIES

To investigate the fundamental properties of the Fourier transform, several analytical functions were analysed using the Fast Fourier Transform (FFT). All signals were sampled at $f_s = 10$ Hz over a duration of 100 seconds to minimise spectral leakage and ensure sufficient frequency resolution.

A. Gate Function: Scaling and Shifting

The Fourier transform of a gate function (rectangular pulse) of widths $T = 5, 10, 20$ s was computed. Theoretically, the Fourier transform of a gate function is a **sinc**

function, where the spectral width is inversely proportional to the temporal width T . Furthermore, each gate was shifted by 35 seconds to observe the effect of the Time-Shifting Property: $\mathcal{F}\{f(t - t_0)\} = F(\nu)e^{-i2\pi\nu t_0}$.

Figure 3 illustrates the gate functions and the real part of their FFTs compared with the analytical solutions. The results for the centered gates match the **sinc** profile perfectly. However, for the shifted functions, the real part of the FFT exhibits rapid oscillations (peaks). This behaviour occurs because shifting a function in time introduces a linear phase frequency-dependent term, $e^{-i2\pi\nu t_0}$. While the magnitude of the spectrum remains invariant, the real part, $\text{Re}\{F(\nu)e^{-i2\pi\nu t_0}\}$, involves a cosine modulation ($\cos(2\pi\nu t_0)$). As the shift t_0 increases, the frequency of these oscillations in the spectral domain increases, resulting in the observed peaky appearance of the real component.

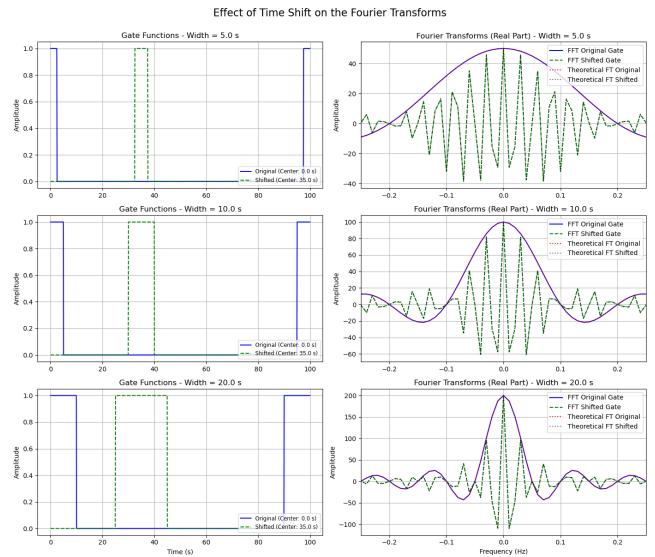


FIG. 3: Gate functions with different widths and their corresponding spectral real parts.

B. Gaussian Functions and Inversion

Gaussian functions were evaluated with standard deviations σ corresponding to the widths used in the previous section. As shown in Figure 4, the Fourier transform of a Gaussian is itself a Gaussian. We observed the reciprocal relationship between domains: a wider Gaussian in the time domain results in a narrower, more peaked spectrum in the frequency domain, consistent with the Uncertainty Principle in signal processing. Additionally, the Inverse Fast Fourier Transform (IFFT) was applied to the spectra, recovering the original time-domain signals with negligible numerical error.

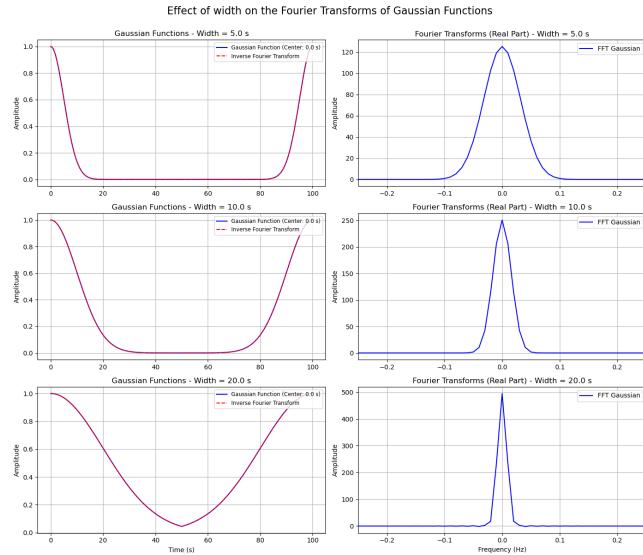


FIG. 4: Gaussian profiles in time and frequency domains, including inverse FFT verification.

C. Derivative Property

The derivative of a Gaussian function was computed following the Derivative Property of the Fourier Transform:

$$\mathcal{F} \left\{ \frac{df}{dt} \right\} = i2\pi\nu F(\nu) \quad (4)$$

The procedure involved computing the FFT of the Gaussian, multiplying the result by $i2\pi\nu$, and then performing the IFFT to return to the time domain. As depicted in Figure 5, the numerically derived function matches the analytical derivative perfectly. This demonstrates the efficiency of spectral methods for performing calculus on sampled data, avoiding finite-difference methods in the time domain.

IV. WINDOWING & ALIASING

The objective of this section is to investigate the limitations imposed by discrete sampling and the effects of finite observation intervals on spectra. A pure multi-tone cosine signal was constructed with a sampling frequency $f_s = 1000$ Hz and $N = 5000$ samples, corresponding to a time interval of $t \in [0, 5]$ s. The signal consists of seven distinct frequencies: $\nu = \{20, 35.3, 412, 550, 950, 1050, 1450\}$ Hz.

A. Aliasing and the Nyquist Limit

The Fast Fourier Transform (FFT) of the cosine signal was computed, and the resulting spectrum is presented in Figure 6. While the lower frequencies are correctly identified, the components at 550, 950, 1050, and 1450 Hz do not appear. This occurs because the maximum frequency that

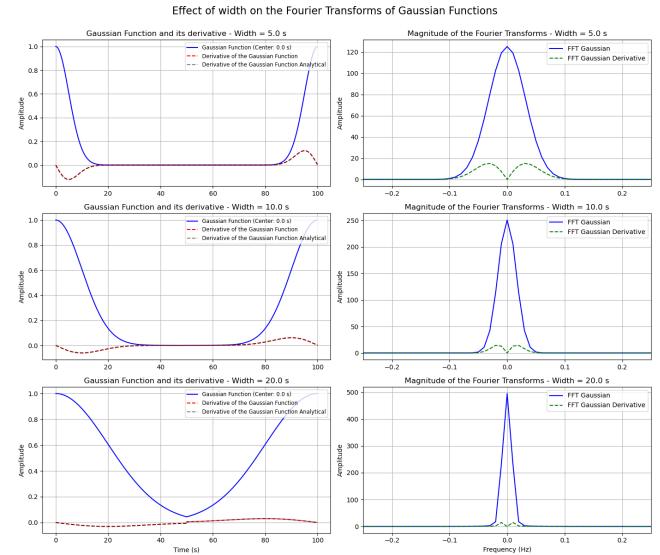


FIG. 5: Gaussian function, its spectrum, the spectral derivative, and the comparison between the numerical and analytical time-domain derivatives.

can be uniquely represented is the Nyquist frequency, defined as $f_{Nyquist} = f_s/2 = 500$ Hz. However, information about those high frequencies is contained in the spectrum as a mirror reflection, in a phenomenon called *aliasing*.

Aliasing occurs when the sampling rate is insufficient to capture the rapid oscillations of a high-frequency signal. When a frequency ν exceeds $f_{Nyquist}$, it folds back into the baseband $[0, f_{Nyquist}]$. Specifically, frequencies are mapped to $|\nu - k \cdot f_s|$. Consequently, 550 Hz and 1450 Hz appear at 450 Hz ($|550 - 1000|$ and $|1450 - 2000|$), while 950 Hz and 1050 Hz are aliased to 50 Hz ($|950 - 1000|$ and $|1050 - 1000|$).

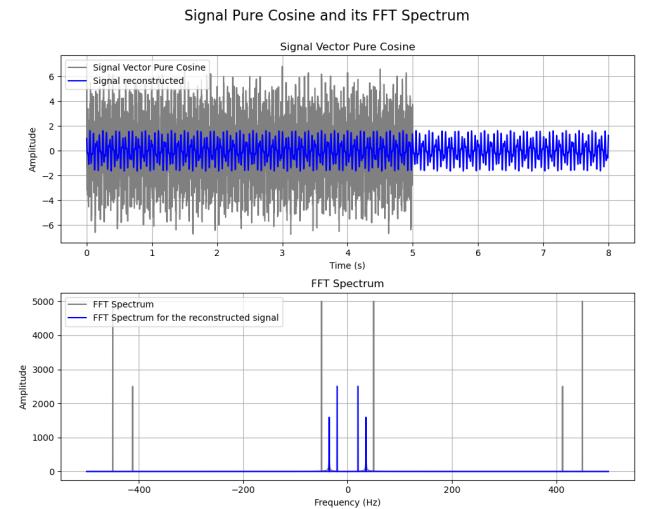


FIG. 6: Spectrum of the cosine signal showing correct frequency peaks and aliased components. Reconstruction of the signal for the first two frequencies.

B. Signal Reconstruction and Periodicity

Using the computed FFT, a filtered spectrum containing only the first two frequencies ($\nu_1 = 20$ Hz, $\nu_2 = 35.3$ Hz) was isolated. By applying the Inverse Fast Fourier Transform (IFFT), the signal was reconstructed and extended beyond the original 5-second window. As shown in Figure 6, the reconstruction is seamless because the FFT inherently assumes that the signal is periodic outside the sampled interval.

C. Windowing Effects

Finally, the effect of the observation window duration was studied by computing the FFT for segments of $\Delta t = 0.5, 1$, and 2 seconds. As evidenced in Figure 7, even the shortest window (0.5 s) allows for the identification of the primary frequencies, though the spectral peaks become better defined as the window duration increases. This behaviour is governed by the relationship between temporal duration and spectral resolution ($\Delta\nu \approx 1/\Delta t$). A shorter window results in spectral leakage, where the energy of a single frequency spreads into adjacent bins. Mathematically, multiplying the signal by a short rectangular window in time is equivalent to convolving its spectrum with a wide **sinc** function in the frequency domain; as the window shrinks, the **sinc** widens, degrading the ability to resolve closely spaced frequencies.

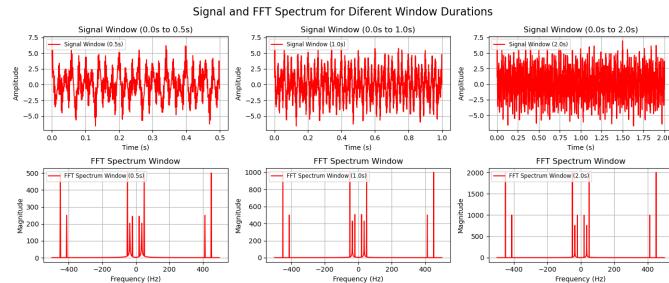


FIG. 7: FFT for three different windows: (0.1, 1.0, 2.0) seconds

V. CONVOLUTION AND SIGNAL FILTERING

This section investigates the impact of filtering techniques on the spectral representation of a signal. A multi-tone synthetic signal was generated using the following expression:

$$s(t) = \sum_{k=1}^5 A_k \cos(2\pi\nu_k t) \quad (5)$$

with amplitudes $A = \{5.0, 2.0, 1.0, 1.5, 2.3\}$ and frequencies $\nu = \{22, 29, 87, 90, 94\}$ Hz. The signal was sampled at $f_s = 1000$ Hz over a duration of $T = 5.0$ s ($N = 5000$).

A. Noise Characterization

White noise with a Gaussian distribution was added to the original signal to simulate experimental interference. The Fast Fourier Transform (FFT) was computed for both the clean and noisy signals, as shown in Figure 8.

The clean signal spectrum displays perfectly defined peaks at the constitutive frequencies, with zero energy elsewhere. In contrast, the noisy signal spectrum preserves these peaks but exhibits a constant baseline of energy across the entire frequency range. This phenomenon occurs because Gaussian white noise is “white” in the spectral sense; it possesses a flat power spectral density. Mathematically, a random distribution in the time domain maps to a broad, stochastic distribution in the frequency domain, effectively raising the noise floor across all frequencies.

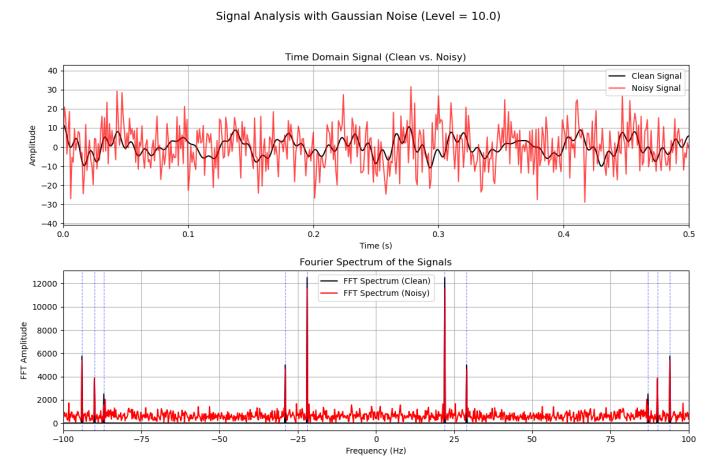


FIG. 8: Time-domain signals and spectral comparison between the clean and noisy data.

B. Comparative Filtering: Rectangular vs. Gaussian

The noisy signal was processed using two distinct temporal filters with a characteristic width of 20 ms:

1. **Moving Average (Rectangular) Filter:** A simple average of neighbouring points within a 20 ms window.
2. **Gaussian Filter:** A convolution with a Gaussian kernel ($\sigma = 20$ ms).

The results, illustrated in Figure 9, show the filtered signals and their respective FFTs. While the Gaussian filter successfully preserves the peaks of the characteristic frequencies (including the higher ones at 87-94 Hz), the rectangular filter nearly eliminates the high-frequency components.

This difference stems from the frequency response of the filters. A rectangular window in time corresponds to a **sinc** function in frequency, which has prominent side lobes and zeros that can accidentally delete specific frequencies. More importantly, the moving average acts as a harsh low-pass filter; since the high frequencies (87-94 Hz) have peri-

ods comparable to the 20 ms window, the rectangular filter averages out their oscillations to near zero. The Gaussian filter, however, has a smooth low-pass response (another Gaussian in frequency) that provides a gentler roll-off, allowing more energy from the high-frequency components to pass through while still suppressing the stochastic noise.

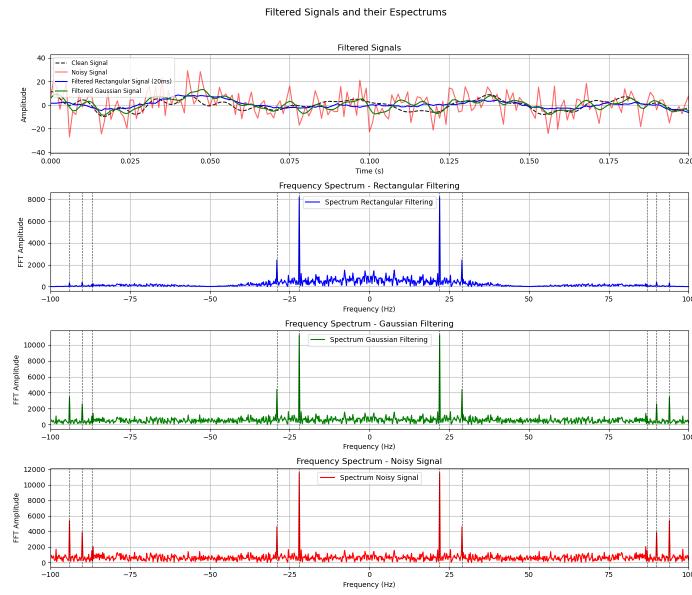


FIG. 9: FFT of the noisy signal after applying Rectangular and Gaussian filters.

VI. AUDIO SIGNAL FILTERING

The objective of this module was to study the implementation of RC and RLC filters for audio processing, specifically targeting the removal of noise and the isolation of tonal components in music and speech recordings. All processed audio files and the corresponding Python implementations can be found in the project repository [1].

A. Cascade RC Low-Pass Filtering

A cascade of two low-pass RC filters was implemented to process a music track. The transfer function for a single RC low-pass filter is defined as:

$$T(f) = \frac{1}{1 + i(f/f_c)} \quad (6)$$

In a cascade configuration, the total transfer function is the product of the individual stages, $T_{total}(f) = T_1(f) \cdot T_2(f)$. While the initial parameters suggested cut-off frequencies of 2.8 kHz and 3.2 kHz, I opted for lower cut-offs at **400 Hz and 1000 Hz** to more clearly demonstrate the filter's impact.

As shown in Figure 10, the filtered spectrum shows a significant attenuation of high-frequency components. Despite the aggressive cut-off, the melody and vocals remain

intelligible. This is because the fundamental frequencies of the voice and instruments reside in the lower end of the spectrum. However, the audio sounds muffled, as if heard from another room. This occurs because the filter eliminates the high-frequency harmonics and brilliance of the recording, which are essential for spatial clarity and sibilance, leaving only the body or lower-mid frequencies of the signal.

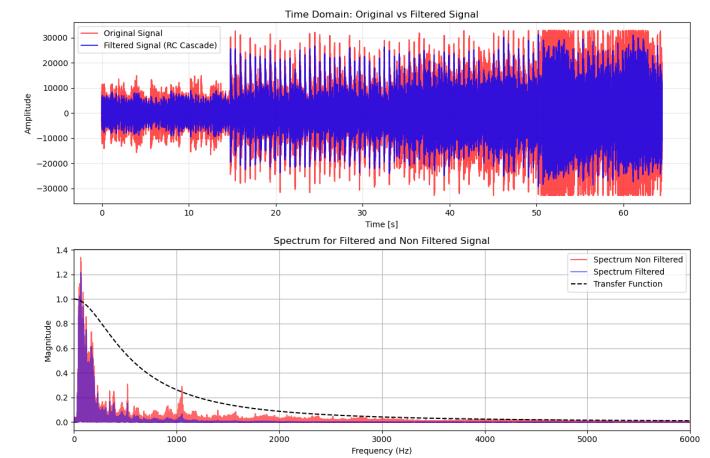


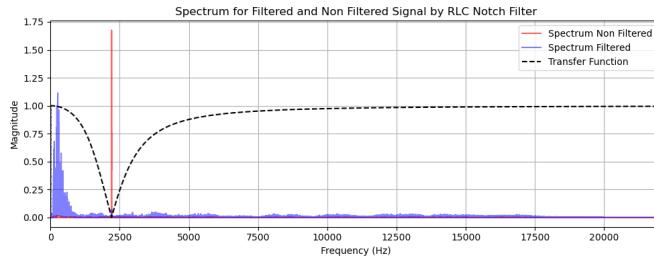
FIG. 10: Time-domain and spectral comparison of the original and cascade-filtered audio.

B. Speech Transcription and Noise Removal

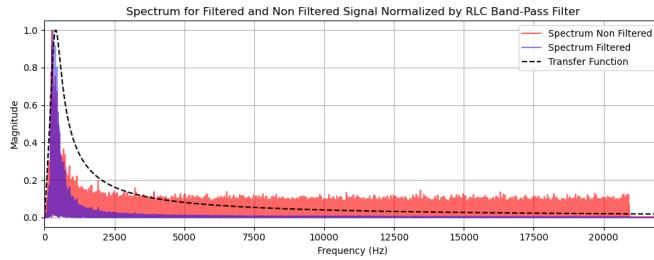
We proceeded to filter two noisy speech recordings with distinct interference characteristics:

Case 1: Tonal Interference (Beep): The first recording contained a constant high-pitched beep at a specific frequency. To address this, an **RLC Notch Filter** was designed with a center frequency of 2206 Hz. As illustrated in Figure 11a, the filter successfully notched out the interference without significantly affecting the speech bandwidth, resulting in a perfectly clear audio output.

Case 2: White Noise: The second recording presented a significant challenge due to intense background white noise. Despite multiple filtering attempts, the stochastic nature of the noise (which spans the entire spectrum) made total elimination impossible without degrading the speech itself. The most effective result was achieved using an **RLC Band-Pass Filter** twice, centered at 400 Hz. By isolating the typical frequency range of the human voice and suppressing the noise outside this band, the speech became slightly more audible. However, since the noise energy within the band-pass region remains, the signal-to-noise ratio improvement was limited. Figure 11b shows the spectral result of this attempt.



(a) RLC Notch Filter result.



(b) RLC Band-Pass result.

FIG. 11: Spectral analysis of speech enhancement techniques.

VII. WINDOWED FOURIER TRANSFORM

The Windowed Fourier Transform (WFT) was implemented to overcome the limitations of the standard FFT in analysing non-stationary signals, such as music, where frequency content evolves over time. By applying a sliding window, we can achieve a time-frequency representation of the signal.

A. Implementation and Spectrogram Analysis

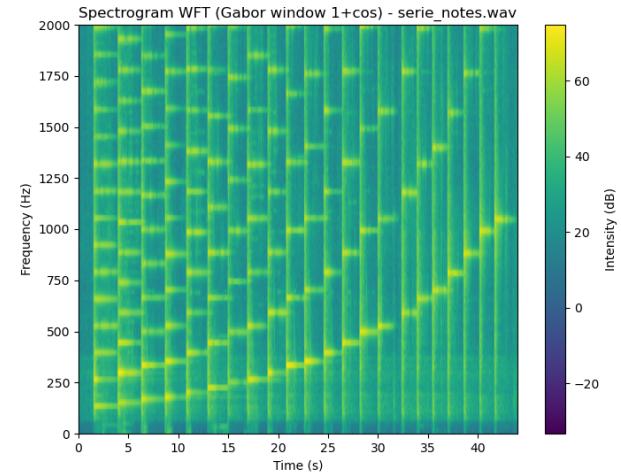
A WFT was developed using a **Gabor window function**, defined as $w(x) = 1 + \cos(x)$ for $x \in [-\pi, \pi]$. This windowing process is crucial as it smoothly tapers the edges of each temporal frame, significantly reducing spectral leakage. The resulting spectrogram visualises the intensity of frequencies as a function of time.

This method was applied to the files `serie_notes.wav` and `serie_notes_pedale.wav`, which consist of a piano playing a chromatic scale in ascending order. As shown in Figure 12, the spectrogram reveal a staircase pattern.

The primary staircase at the bottom corresponds to the fundamental frequencies (f_0). The additional parallel patterns above it are the harmonics ($2f_0, 3f_0, \dots$), which define the characteristic timbre of the piano but represent the same musical pitch.

B. Note Identification and Pitch Detection

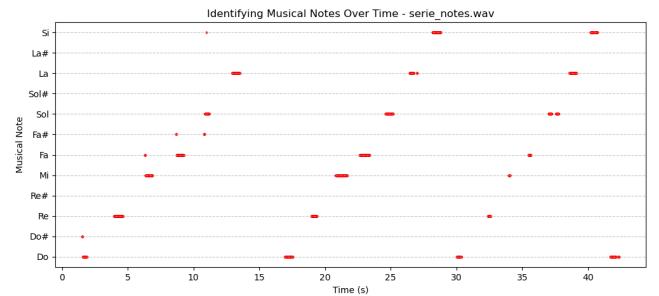
To identify the specific notes being played, we utilized the mathematical relationship between frequency and semitones relative to the standard pitch La = 440 Hz:

FIG. 12: Spectrogram for `serie_notes.wav` showing the ascending chromatic scales. The lower step represents the fundamental frequency (f_0), while the parallel lines above represent the higher-order harmonics.

$$n = 12 \cdot \log_2 \left(\frac{f}{440} \right) \quad (7)$$

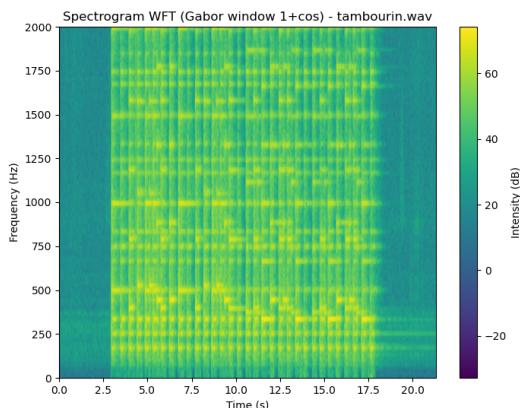
where n represents the number of semitones away from La. By detecting the frequency with the maximum intensity in each time frame, we mapped the spectral peaks to musical notes.

The results, presented in Figure 13, show a consistent ascending sequence. The identification confirms that the scale spans several octaves, where each octave represents a doubling of the fundamental frequency.

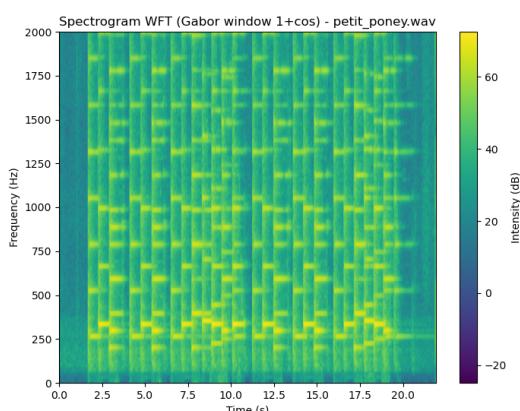
FIG. 13: Chronological sequence of detected musical notes. Detected notes for: `serie_notes`

C. Musical Piece Analysis and Histograms

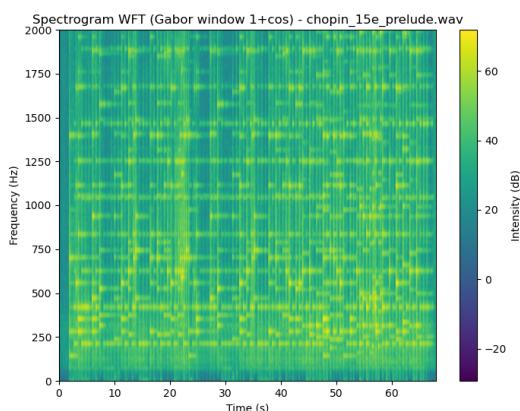
The WFT algorithm was further tested on three distinct recordings: *Tambourin*, *Petit Poney*, and *Chopin's Prelude No. 15*. The spectrograms (Fig. 14) reveal the rhythmic and harmonic complexity of each piece.



(a) Tambourin



(b) Petit Poney



(c) Chopin Prelude

FIG. 14: Comparative spectrograms of the analyzed musical pieces.

Finally, I generated histograms (Fig. 15) to visualise the distribution of notes. This statistical approach allows us to identify the tonality or the most dominant notes in each composition, providing a signature of the musical work's harmonic structure.

D. Voice Identification and Transformation

This final module explores the physiological and spectral differences between human voices and the implementation of digital signal processing techniques to modify vocal timbre. For this study, the phrase "*May the Force be with you*" was recorded by three different subjects: a female speaker (Person 1), a male speaker (Person 2), and a Google Translate synthetic voice (Bot).

E. Spectral Analysis and Subject Identification

The spectrograms for each subject were computed using the WFT and are presented in Figure 16. The visual representation reveals distinct horizontal bands, which correspond to the harmonic series of the vocal signal.

A clear distinction is observed in the harmonic spacing. Person 2, possessing a deeper (lower-pitched) voice, exhibits harmonics that are more closely packed compared to Person 1. This is consistent with acoustic theory: a lower fundamental frequency (f_0) results in a smaller frequency interval between successive harmonics (nf_0). The Bot's spectrum shows spacing similar to Person 1, reflecting its design to emulate a female vocal range.

To achieve a robust identification, we calculated the Average Spectral Footprint by averaging the intensity of each frequency bin over the total duration of the recording. The results are shown in Figure 17.

In the low-frequency range (0–500 Hz), Person 2 displays peaks at shorter intervals, with an estimated $f_0 \approx 125$ Hz, while Person 1 shows wider spacing with $f_0 \approx 250$ Hz. The Bot's spectral behavior, however, lacks the natural harmonic decay of human speech. The Bot's inconsistent harmonic pattern arises from its synthesis method (likely concatenative or parametric synthesis), which often prioritizes intelligibility and smooth transitions over the faithful reproduction of the physical harmonics found in natural human voice.

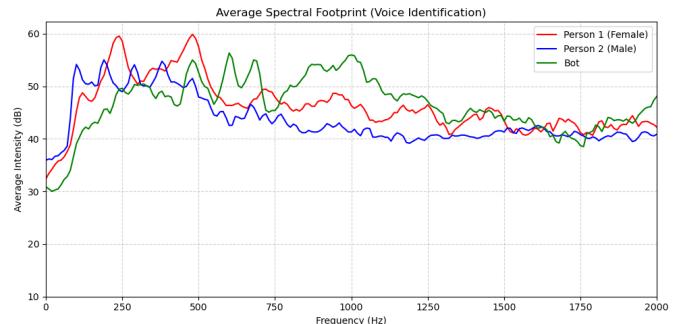


FIG. 17: Average intensity vs. frequency for voice identification.

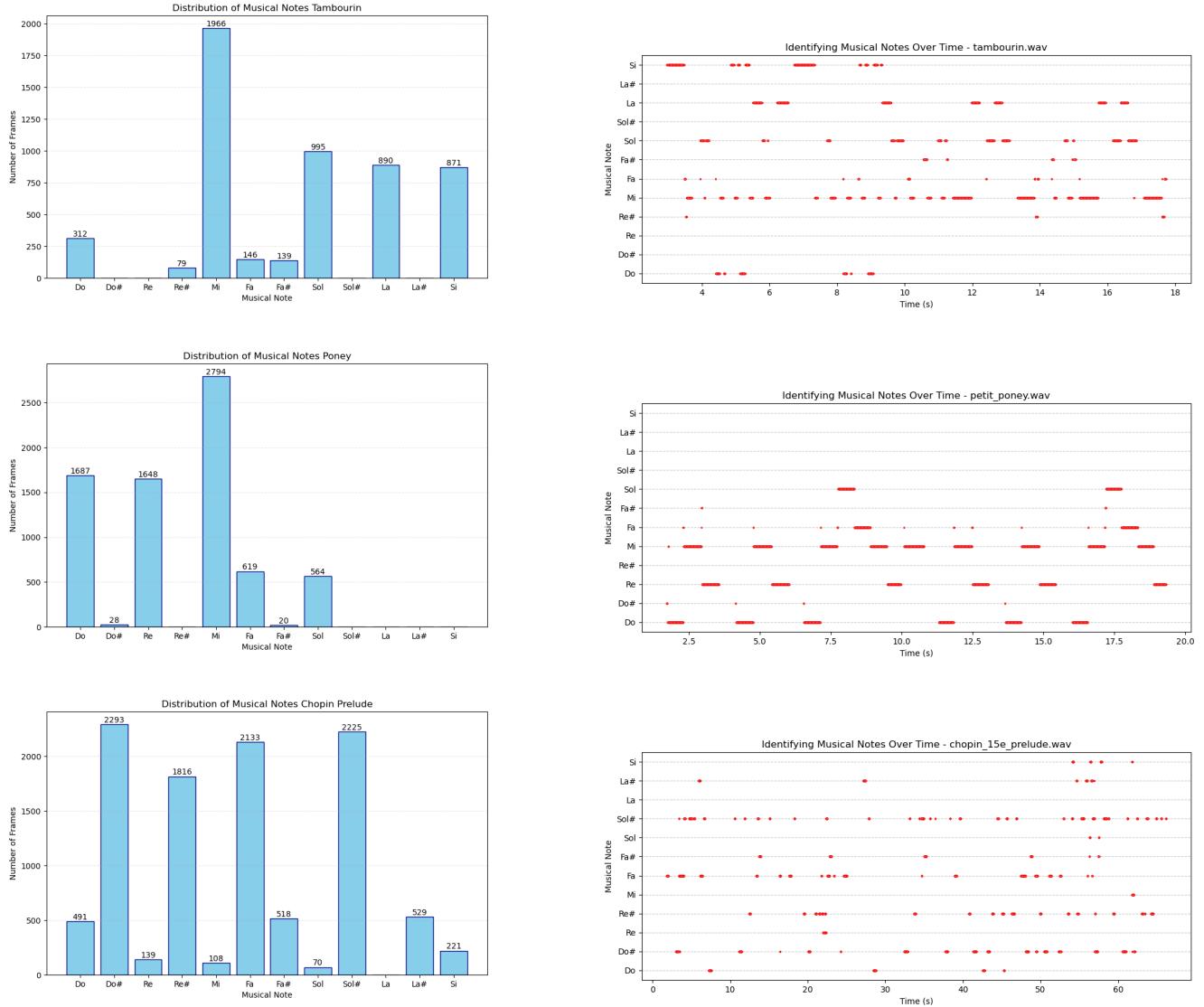


FIG. 15: Musical analysis: Histograms of note frequency distribution (left) and note sequence over time (right) for the three pieces.

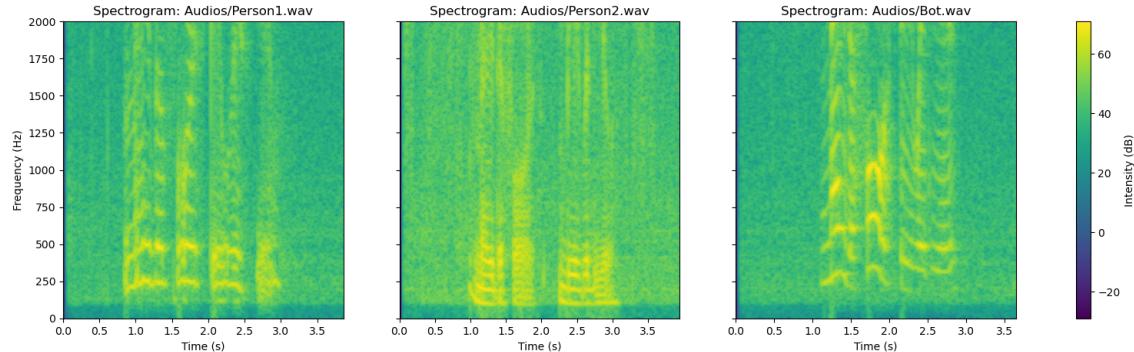


FIG. 16: Comparative spectrograms of the same sentence recorded by different speakers (from left to right: Person 1 (Female), Person 2 (Male), and Synthetic Voice (Bot)). The vertical axis represents frequency in Hz and the horizontal axis represents time in seconds.

F. Voice Transformation (Pitch Shifting)

Finally, a program was developed to modify the vocal timbre by scaling the harmonic spacing in the complex spectral domain. By compressing or expanding the distance between harmonics, the voice can be perceived as deeper or higher-pitched, respectively.

Figure 18 illustrates this effect: the original spectrogram of Person 2 (with closely spaced harmonics) is transformed into a version with wider spacing, successfully shifting the pitch towards a higher register.

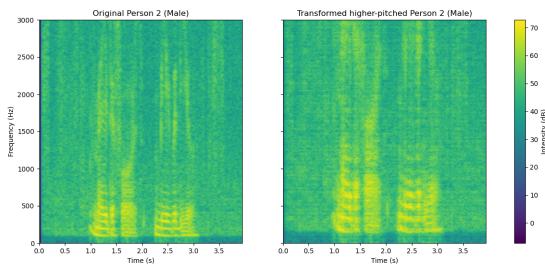


FIG. 18: Original vs. Transformed spectrogram demonstrating pitch shifting for Person 2 (male voice).

All audio recordings and implementation codes are available in the project repository [1].

VIII. CONCLUSIONS AND DISCUSSION

This laboratory demonstrated the robust capabilities of computational tools in the characterisation and manipulation of digital signals. Through the implementation of the Fast Fourier Transform (FFT) and Windowed Fourier Transform (WFT), we established a clear link between theoretical phenomena—such as aliasing, spectral leakage, the uncertainty principle, and their practical implications in music and speech analysis. The use of complex spectral manipulation for pitch shifting proved that preserving phase information is critical for high-fidelity audio reconstruction, successfully transforming vocal timbres.

Furthermore, the comparative analysis of filtering techniques highlighted the trade-offs between different kernel geometries; while moving-average filters provided aggressive noise suppression, they significantly degraded high-frequency signals compared to the smoother roll-off of Gaussian and RLC filters. The identification of musical notes and vocal footprints underscores the efficiency of spectral methods in extracting deterministic patterns from non-stationary data. Overall, these sessions solidified our understanding of digital signal processing as a fundamental bridge between physical acoustics and computational skills.

REFERENCES

- [1] M. A. Herrera Arias, “SMS-ComputerLab.” <https://github.com/ManuelHerreraA/SMS-ComputerLab>, 2025.