“Exploration of Timbre by Analysis and Synthesis”

Using Python, Ableton and ChatGPT

Dr. Egor Polyakov

HOCHSCHULE
FÜR MUSIK UND THEATER
»FELIX MENDELSSOHN
BARTHOLDY«
LEIPZIG

Historical and theoretical background

What is "Exploration of Timbre by Analysis and Synthesis"?

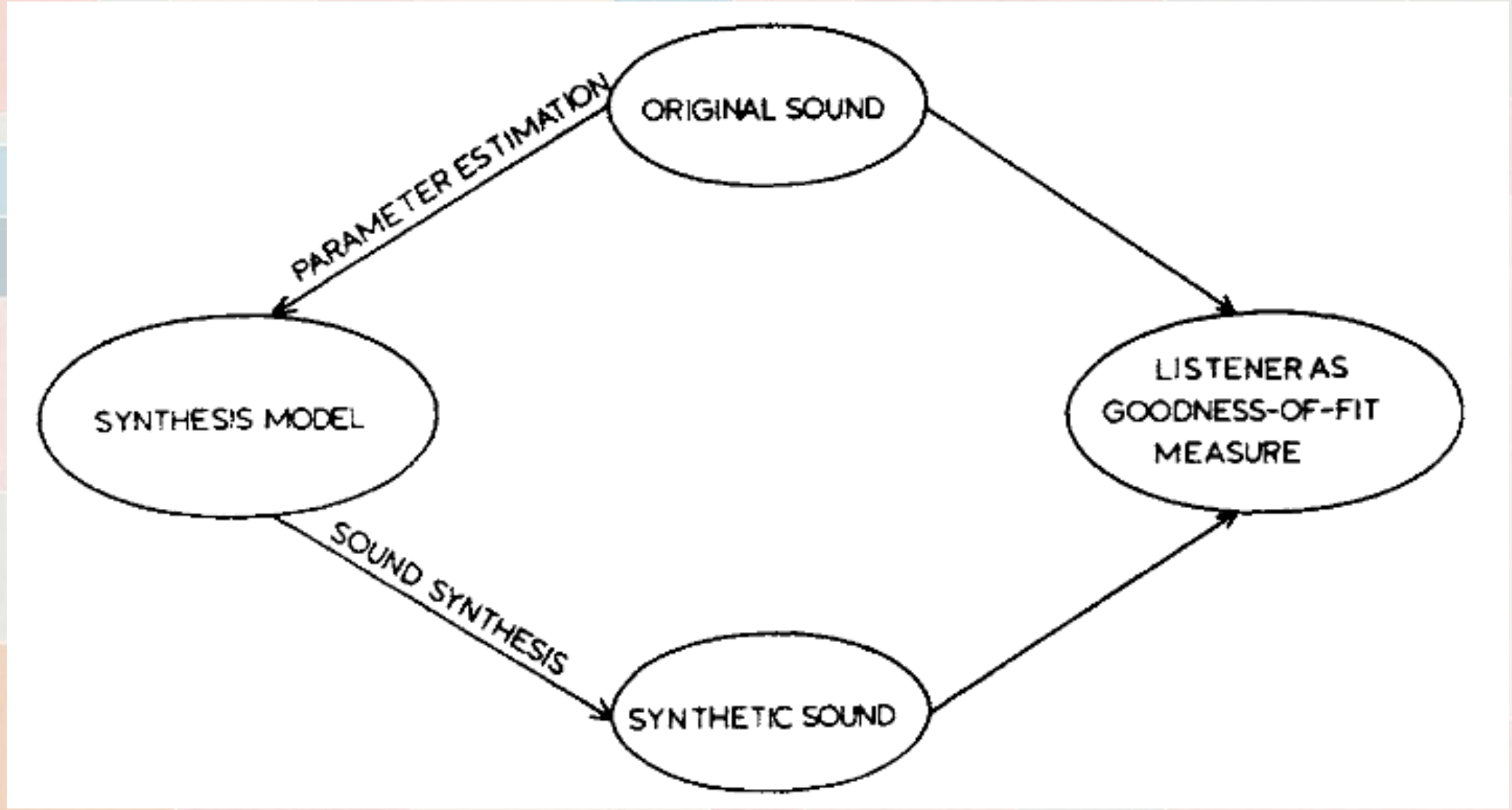
- A theoretical concept developed by Jean Claude Risset in the 1960s.
- Originated from his work at Bell Labs with Max Matthews.

Main Concept

- Utilizes human sonic perception to evaluate the timbral quality of resynthesis.

- Achieves a deeper understanding by comparing:
    - Original sound.
    - Its analytic model.
    - The synthesized version according to the model.

- Helps to define the limits of what can be analyzed, modeled, and synthesized.
- Reveals intricate details of the original sound through this comparative approach.



Conceptual framework of the analysis-synthesis process by Risset & Wessel (1999).

Conceptual Differences from Modern Workflows

No Real-Time Analysis or Synthesis

- Unlike modern workflows, the original concept did not support real-time processing.

Spectral Analysis

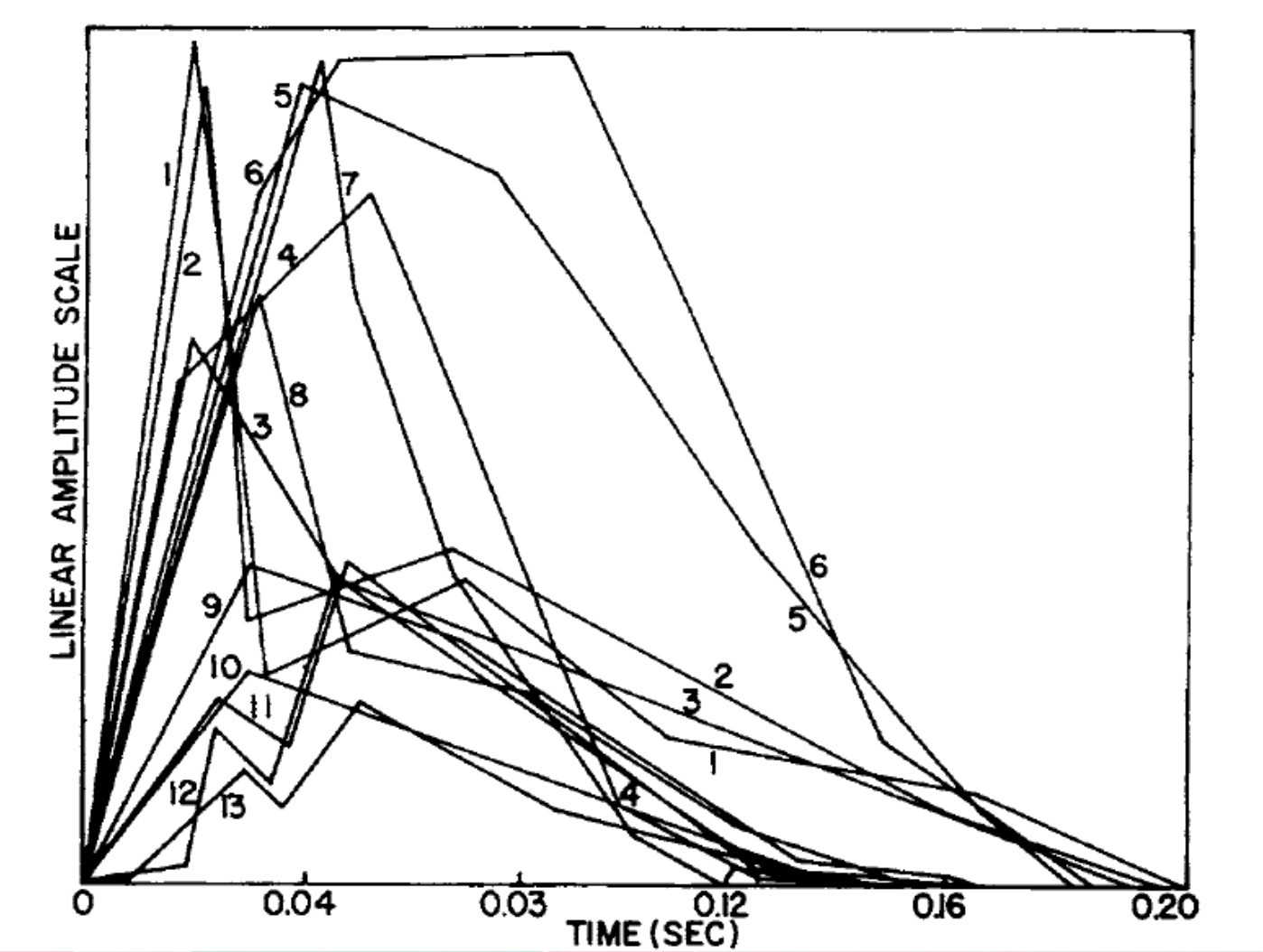
- Conducted via time-variant harmonic analysis.
- The FFT algorithm by Cooley and Tukey, published in 1965, was not implemented until much later.

Software Utilized

- (Additive) Synthesis was performed using the Music IV software run on the Bell Labs IBM 7094.

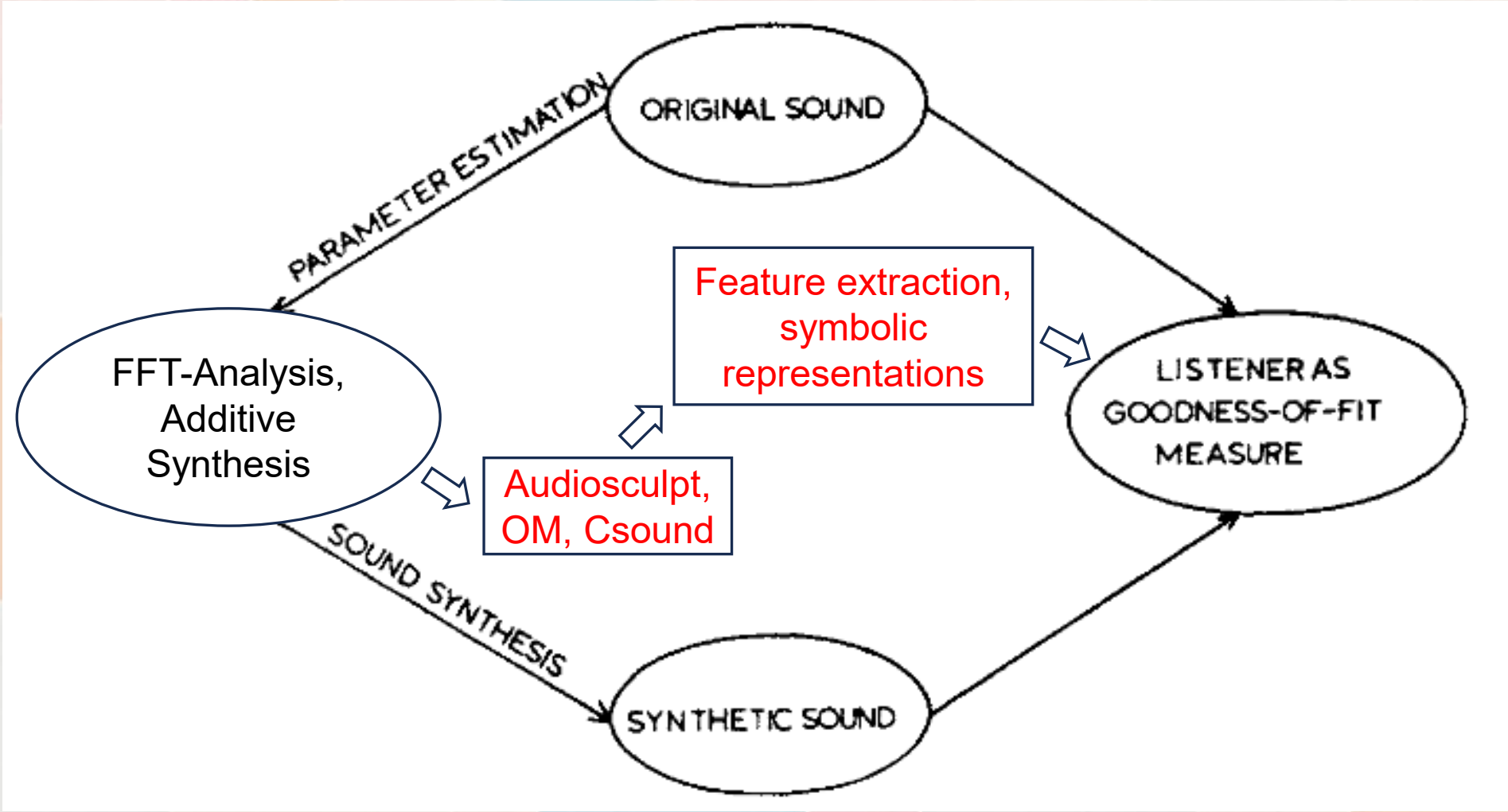
A deep understanding of the entire analysis/synthesis process was necessary for optimal workflow.

Rissset – Mutations (1969)



Example of Line-segment functions that approximate the evolution in time of 13 harmonics of a D4 trumpet tone lasting 0.2 sec (Risset & Matthews 1969).

“Exploration of Timbre by Analysis and Synthesis” today?



“Exploration of Timbre” within Modern Workflows

Real-Time Analysis or Synthesis
- Real-time analysis and resynthesis are now possible, e.g., within Max.

Spectral Analysis
- Various analyses based on FFT, such as partial tracking and chord sequence analysis.
- Import/Export of spectral analysis values via SDIF.

Symbolic Representation
- Open Music can be used for symbolic music representations of spectra.

Synthesis
- Sophisticated synthesis methods are available within Csound.

Various feature representations can enhance understanding of the entire analysis/synthesis process as well as the initial properties of the analyzed sound.

“Exploration of Timbre” without Audiosculpt?

Introduction of 64-bit OSX 10.15 in 2019

- Audiosculpt can no longer run on newer machines.
- Licensing issues arise even on old 32-bit OSX machines.

Alternatives?

Spear

- Buggy, SDIF not supported, not updated since 2019.

Partiels

- Successor of Audiosculpt by IRCAM, initially released in 2021 with limited functionality.
- Still actively in development, Available for free with Linux support.

Why not go Python and recreate the “original” Audiosculpt/OM setup?

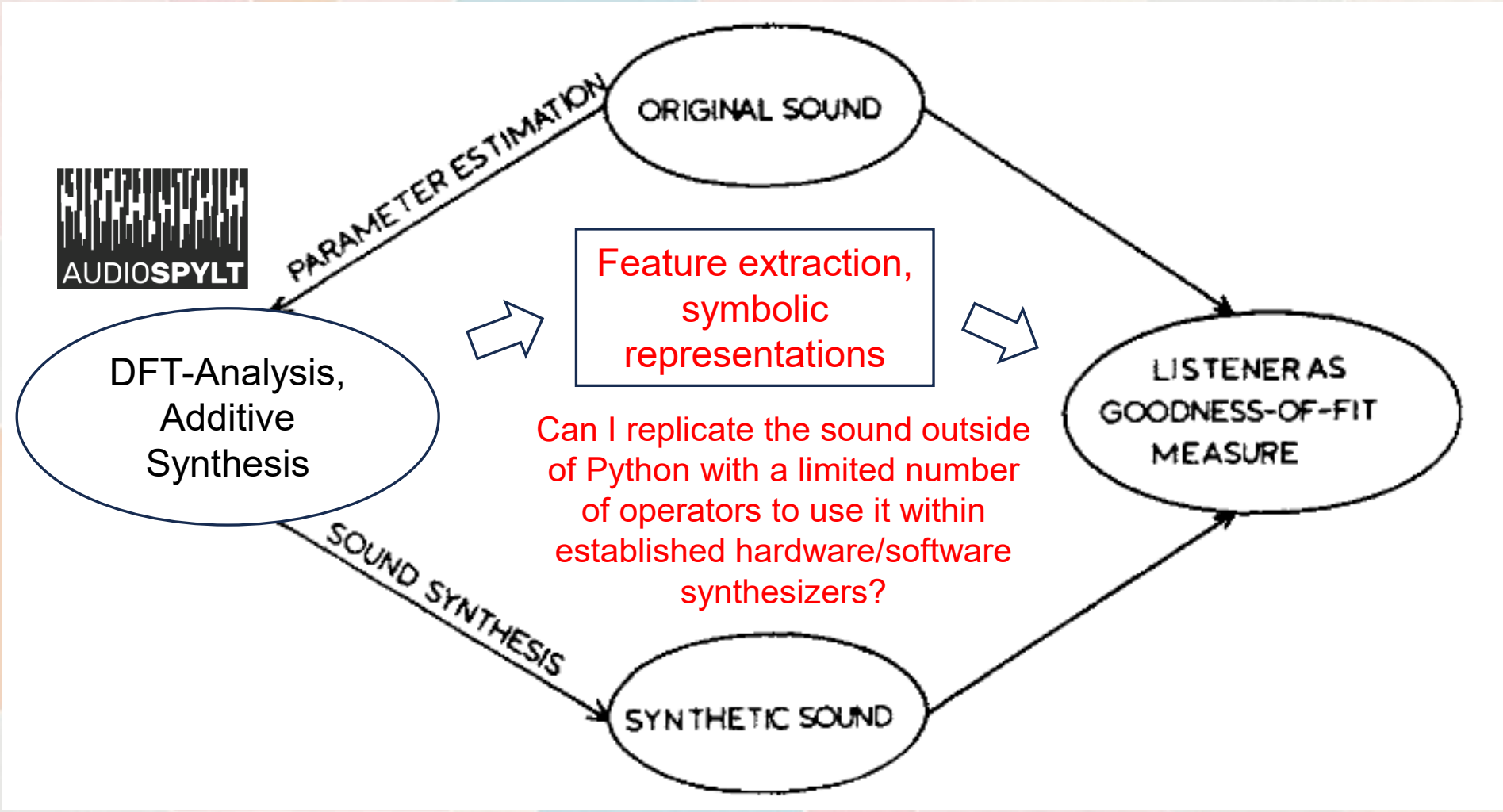
Introducing Audiospylt

- **Py**thon-based toolbox tailored for sound analysis, re/synthesis, and diverse visual & symbolic sound representations.
- Operates within the Jupyter Notebooks environment.

Audio in Python

- Generating audio is equivalent to plotting a waveform.
- Sample rate/bit depth functions as the resolution of your canvas.
- Utilizes Pandas DataFrames for saving spectral information (frequency/amplitude tables).
- SDIF support is coming!

	freq_start	freq_stop	time_start	time_stop	amp_min	amp_max
0	71.428571	71.428571	0.03	0.1	0.494724	0.494724
1	128.571429	128.571429	0.03	0.1	0.477712	0.477712
2	157.142857	157.142857	0.03	0.1	0.465799	0.465799
3	185.714286	185.714286	0.03	0.1	0.511994	0.511994
4	242.857143	242.857143	0.03	0.1	0.567231	0.567231
5	271.428571	271.428571	0.03	0.1	0.604748	0.604748


AUDIOSPYLT
PARAMETER ESTIMATION
ORIGINAL SOUND
Feature extraction, symbolic representations
DFT-Analysis, Additive Synthesis
LISTENER AS GOODNESS-OF-FIT MEASURE
Can I replicate the sound outside of Python with a limited number of operators to use it within established hardware/software synthesizers?
SOUND SYNTHESIS
SYNTHETIC SOUND

Synthesizing Sound Outside of the Python Environment

Comparison of Synthesis Methods

- Additive synthesis in Audiospylt uses an (almost) unlimited number of operators.
- For practical purposes, we need to restrict our output to a certain setting.

=> Additive Synthesis/FM Synthesis with limited number of operators (4).

Finding Optimal Settings for the Operators

- Manual search (16 vCPU, CUDA/Torch only marginally faster):

```
Optimizing:   0%                                  114/16199612306357045 76 [00:20<5061090894787:45:32,  88.91it/s]
```

```
# Define frequency and amplitude ranges
frequency_range = np.arange(50, 5005, 5)
amplitude_range = np.arange(0.1, 1.0, 0.1)
```

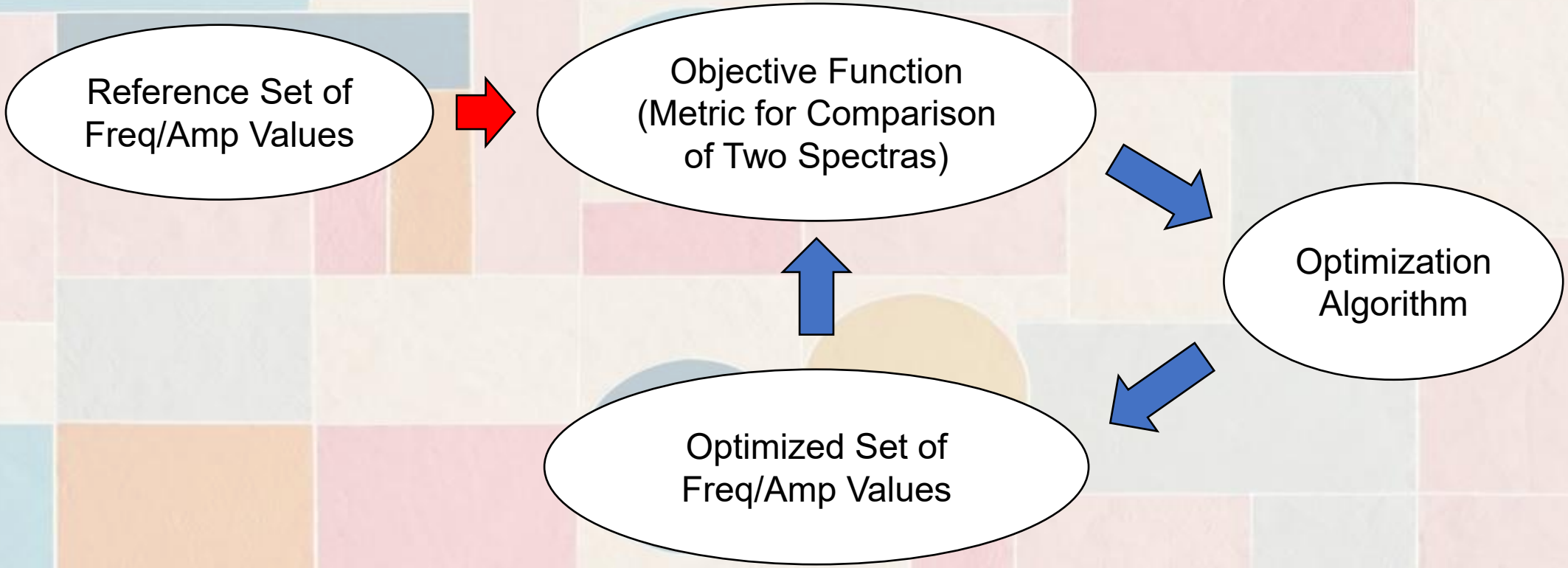
- **Algorithmic search**

=> Global Optimization Algorithms.

- **Objective Function**

=> A mathematical function that quantifies the difference between the current candidate solution and the target solution.

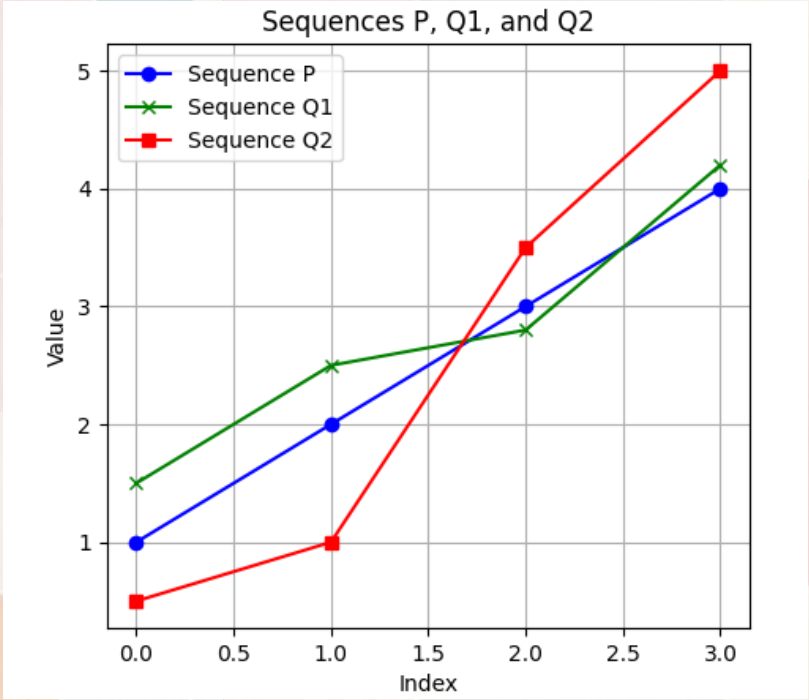
ML Modell for Matching Spectra

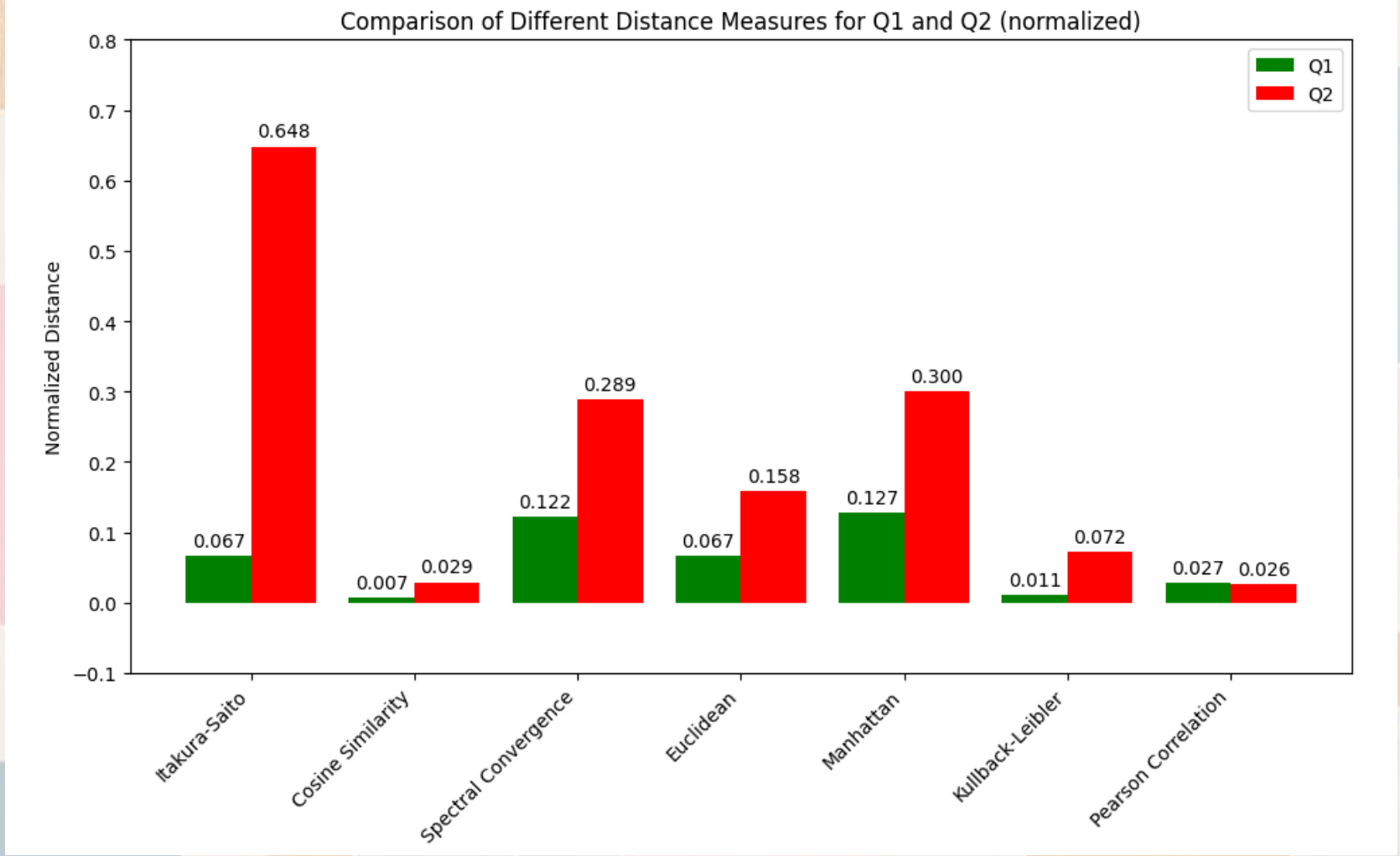


Objective Functions: Itakura-Saito distance, Cosine Similarity, Spectral Convergence distance, Euclidean distance, Manhattan Distance, Kullback-Leibler divergence, Pearson correlation coefficient, MFCC distance

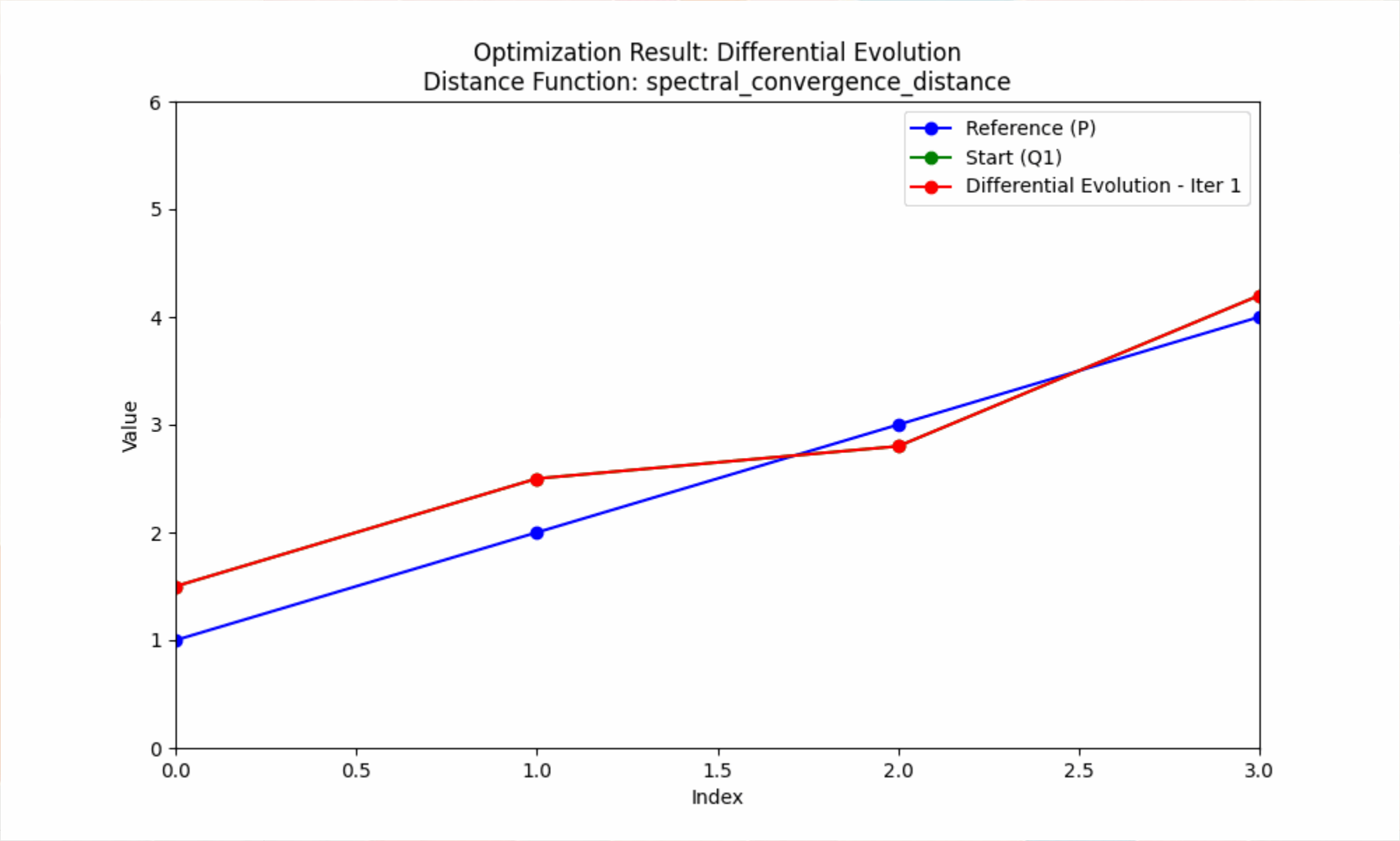
Optimization Algorithms: Basin-hopping, Differential evolution, Dual Annealing

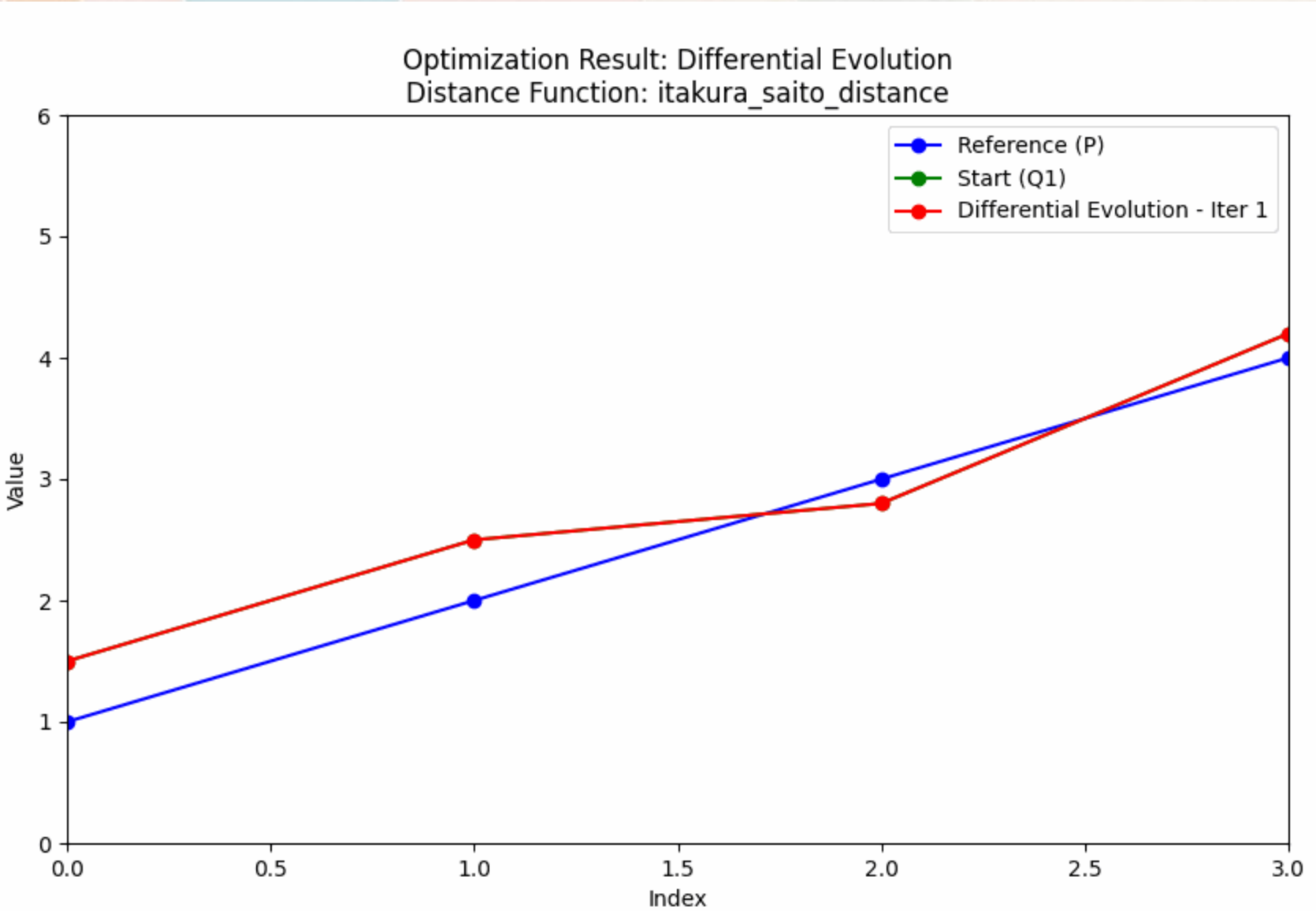
Objective Functions





Optimization Algorithm Learning




Optimization Result: Differential Evolution
Distance Function: itakura_saito_distance
Reference (P)
Start (Q1)
Differential Evolution - Iter 1
Value
Index
0
1
2
3
4
5
6
0.0
0.5
1.0
1.5
2.0
2.5
3.0

Final Setup and Evaluation

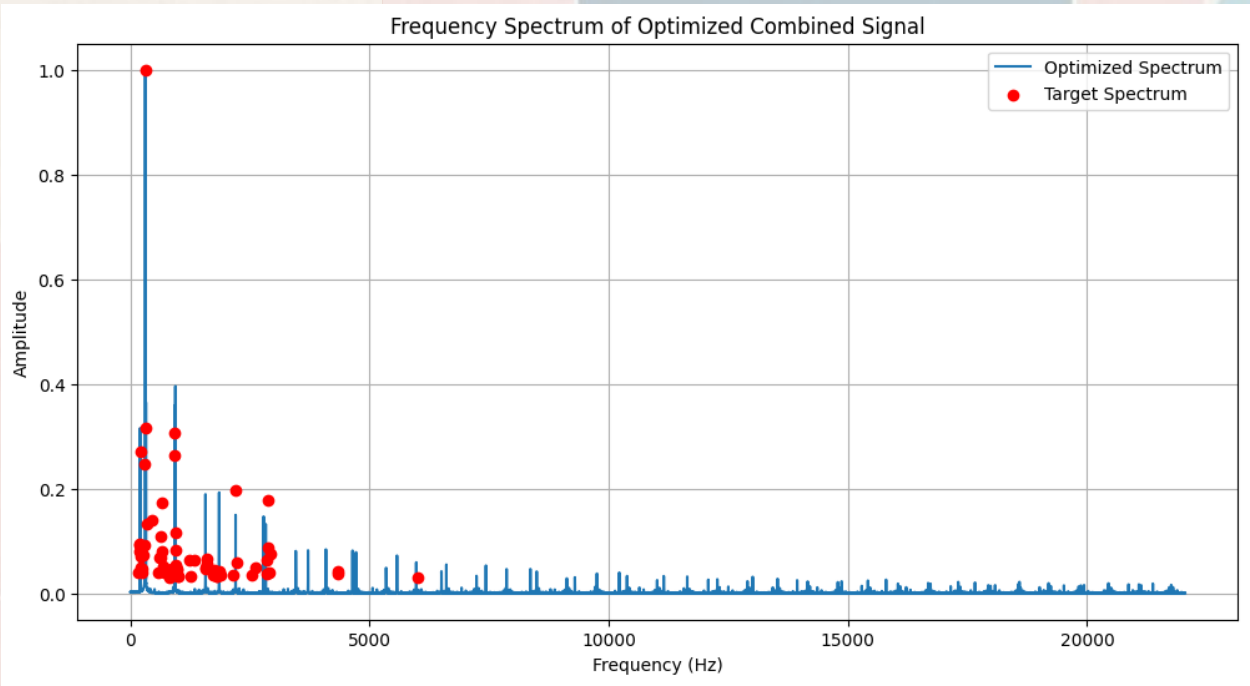
- **AM Synthesis:** Implemented as OP1 + OP2 + OP3 + OP4.
- **FM Synthesis:** Implemented as OP4 => OP3 => OP2 => OP1.

- Optimization is **VERY SLOW** (up to 30 min on 16 vCPU).
- Performance depends on the selected objective function and optimization algorithms.
- General evaluation of the best-fitting objective function and optimization algorithms is difficult.
- Performance varies due to randomness and variance in reference spectra.

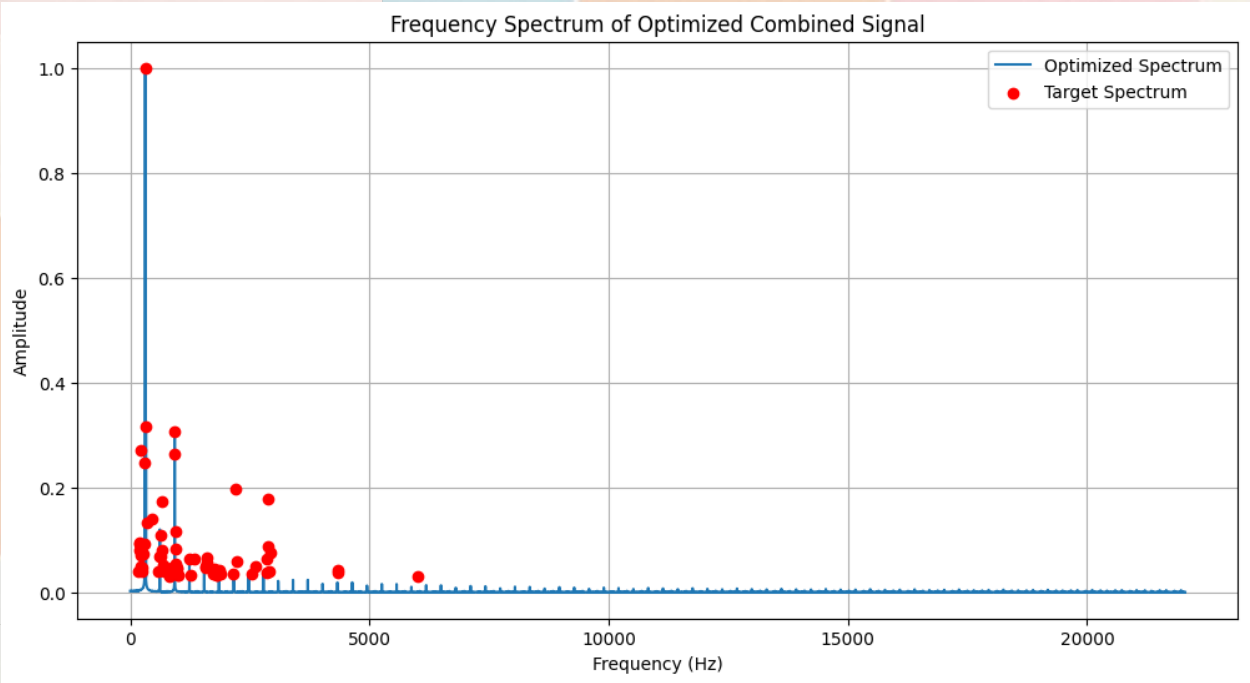
Optimization Algorithms and Objective Functions

- **Differential Evolution:** Most stable and best-performing optimization algorithm
- Itakura-Saito and Kullback-Leibler Divergence most stable and best-performing objective functions for AM.
- Cosine Similarity and MFCC Distance best performing for FM.

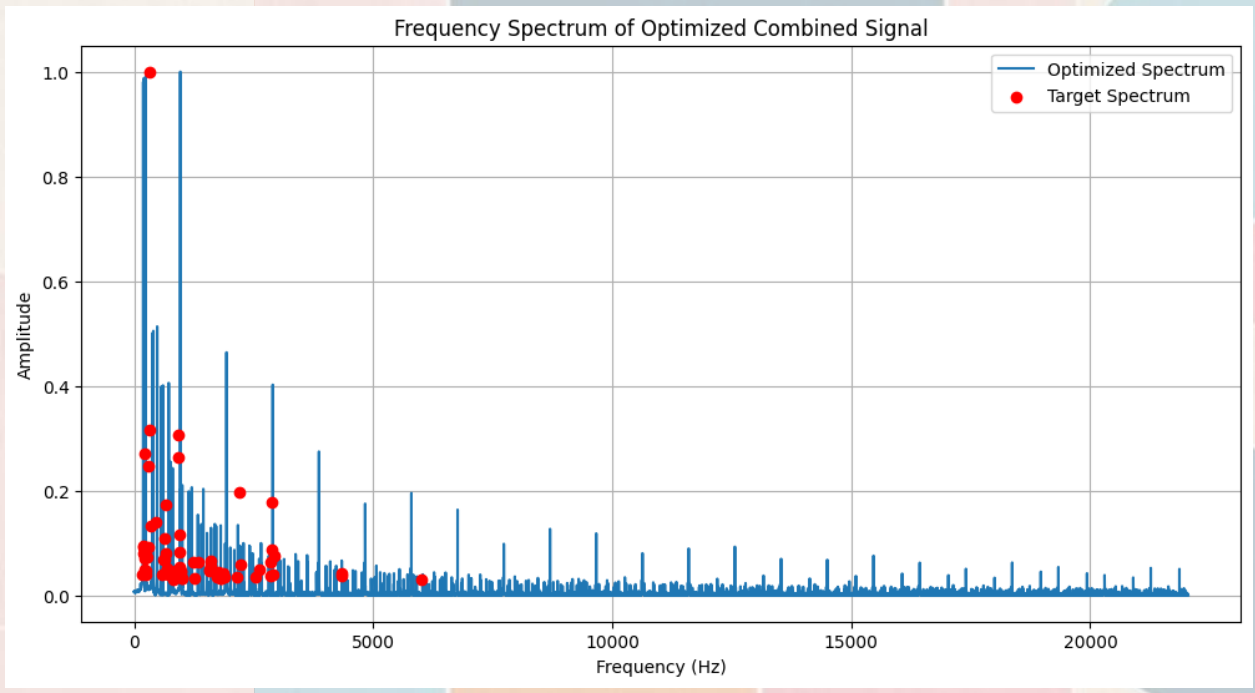
- **Basin-Hopping:** Poor performance but retained for artistic purposes



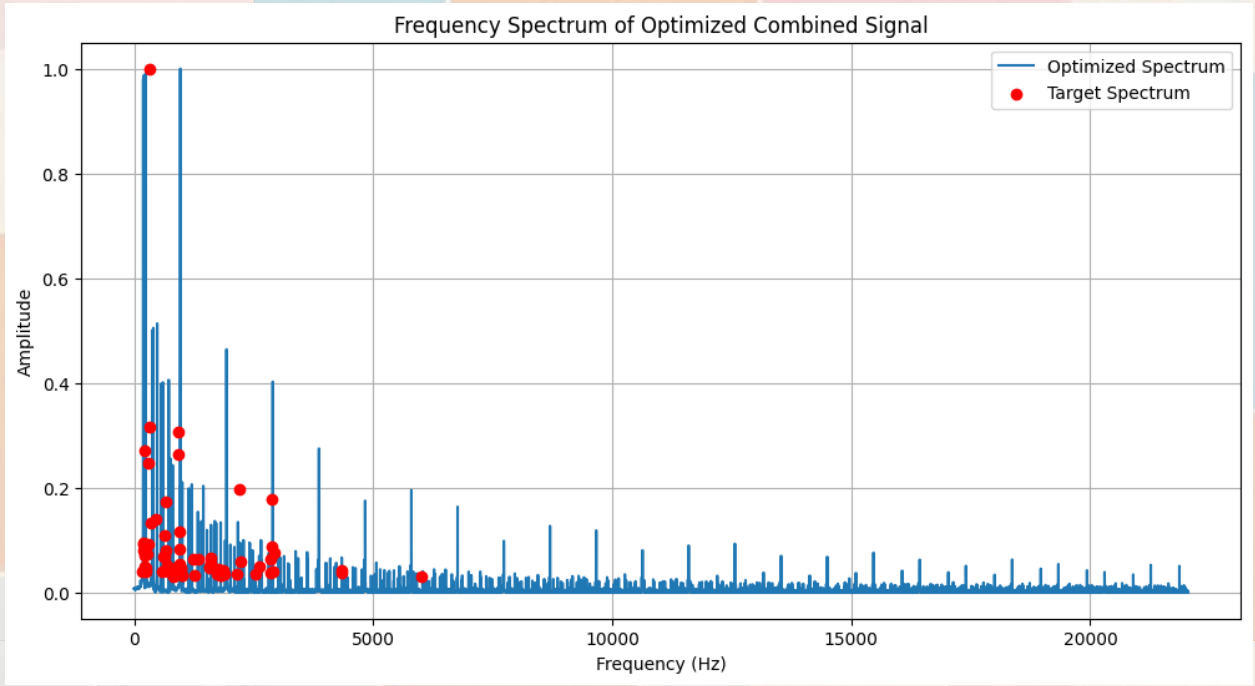
Itakura-Saito Distance with Differential Evolution (AM)



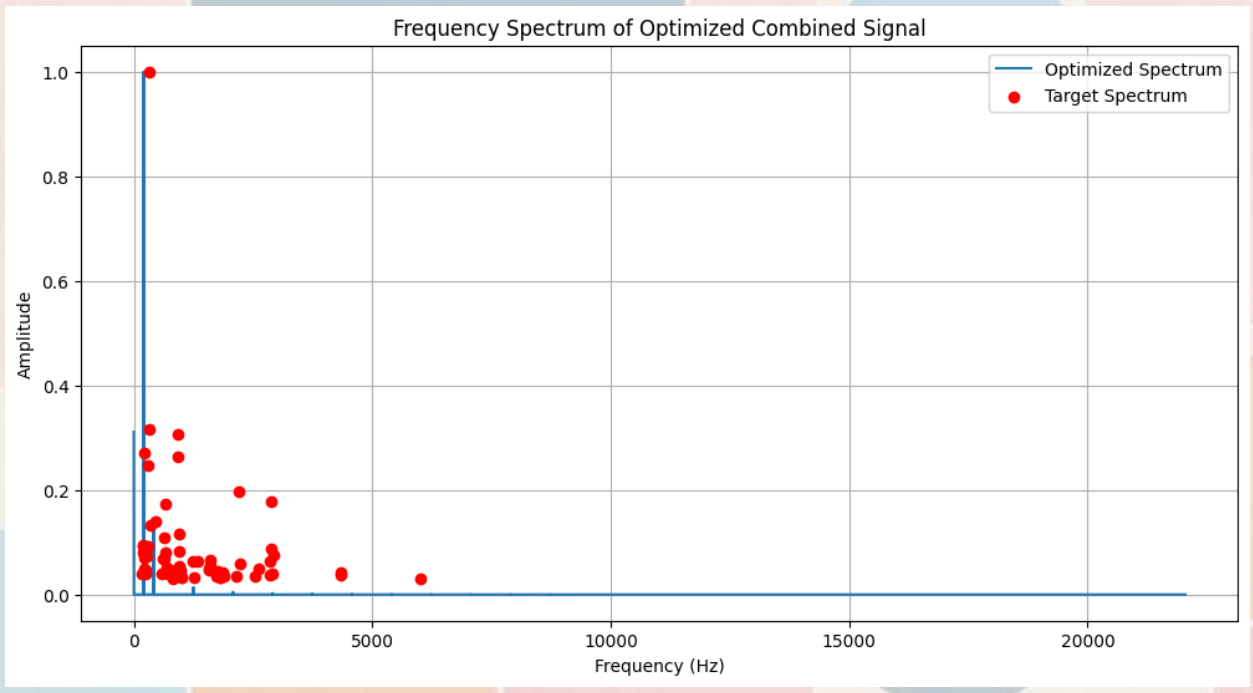
Cosine Similarity with Differential Evolution (AM)



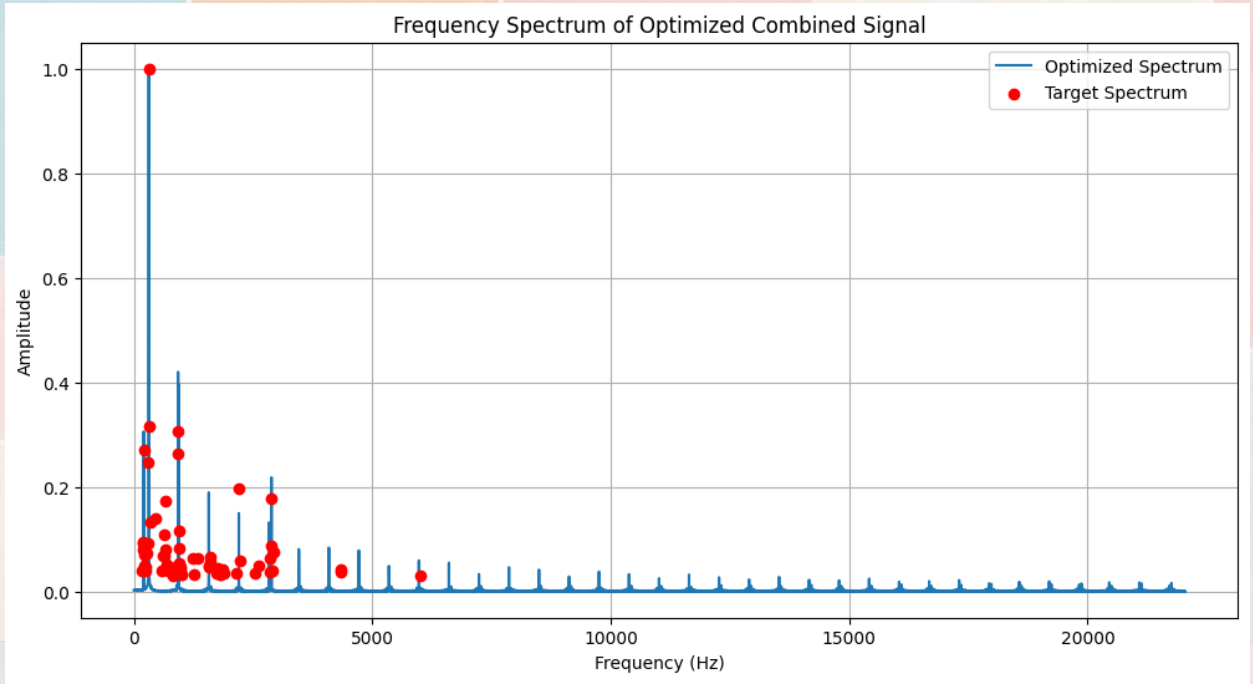
Spectral Convergence Distance with Differential Evolution (AM)



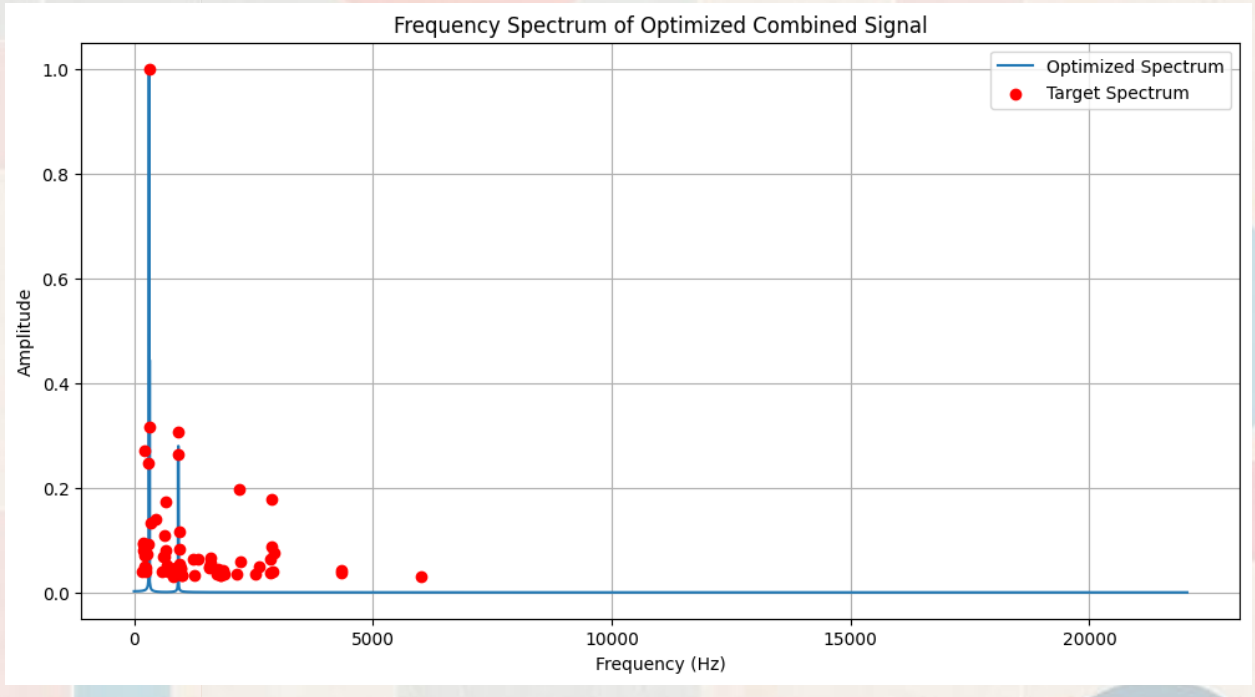
Euclidean Distance with Differential Evolution (AM)



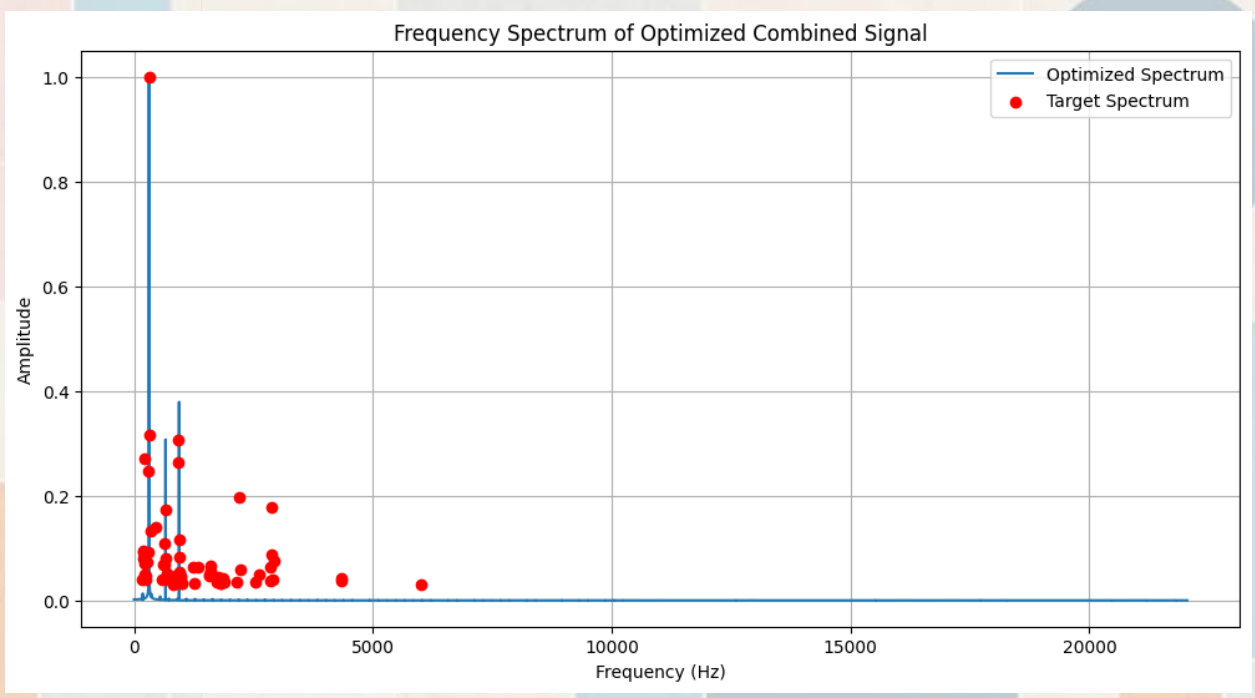
Manhattan Distance with Differential Evolution (AM)



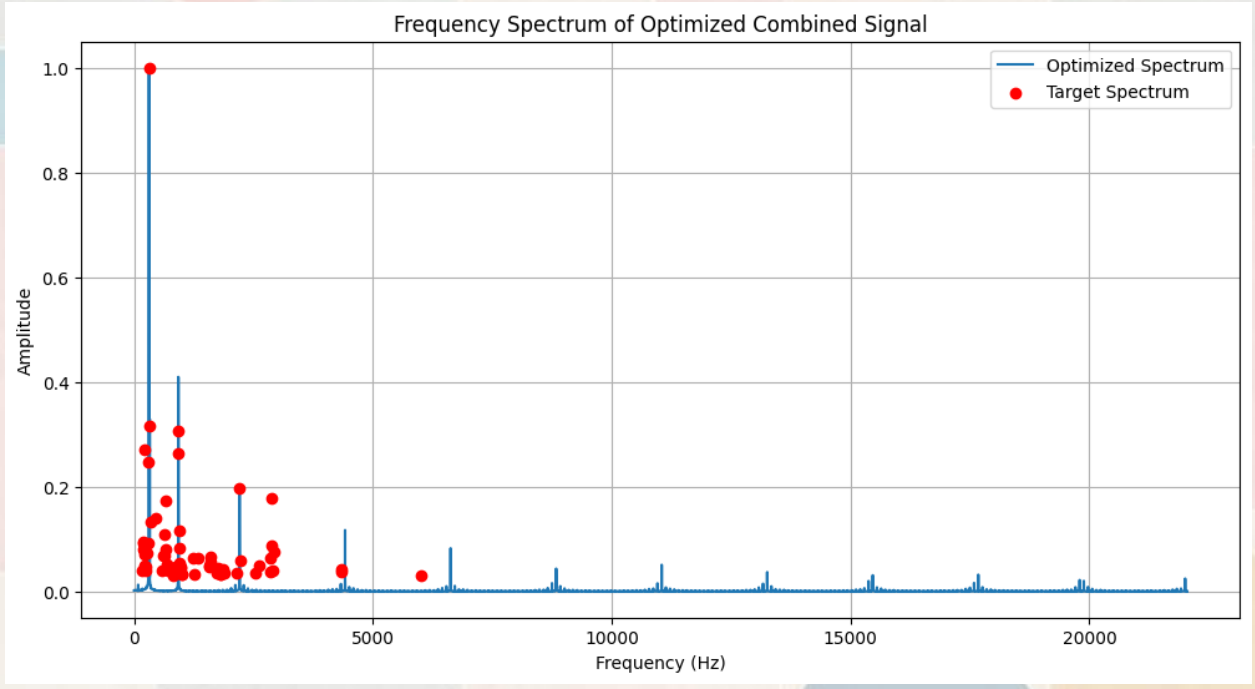
Kullback-Leibler Divergence with Differential Evolution (AM)



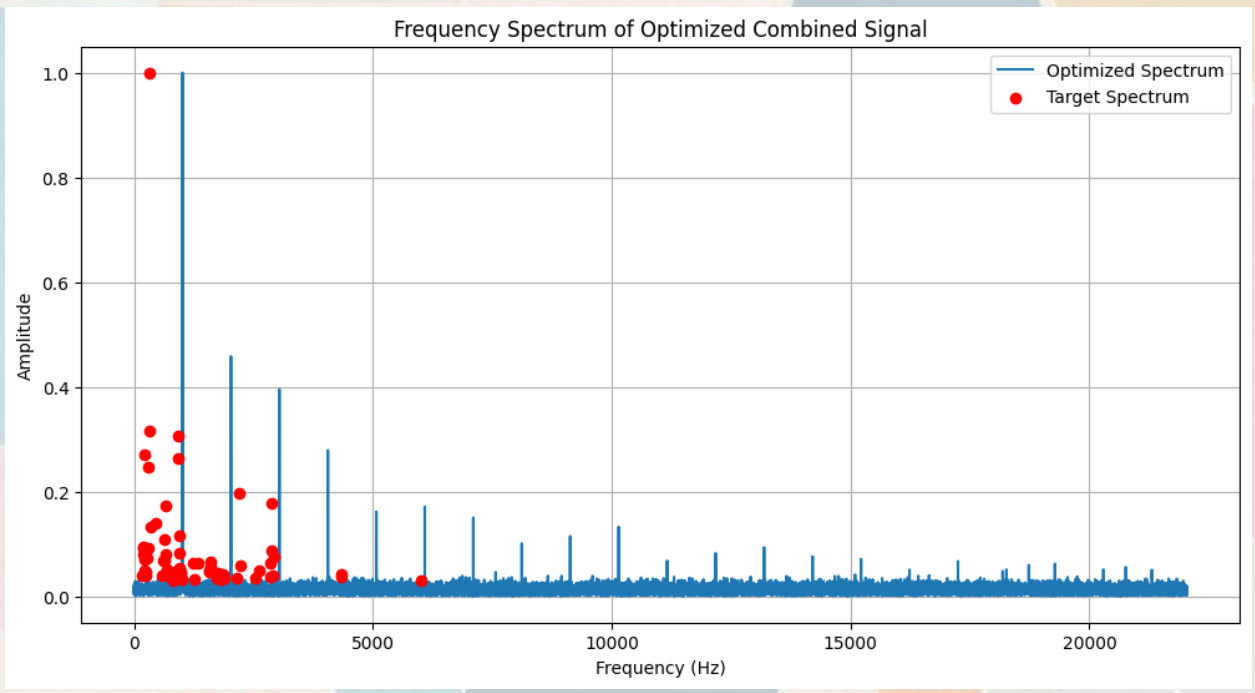
Pearson Correlation Coefficient with Differential Evolution (AM)



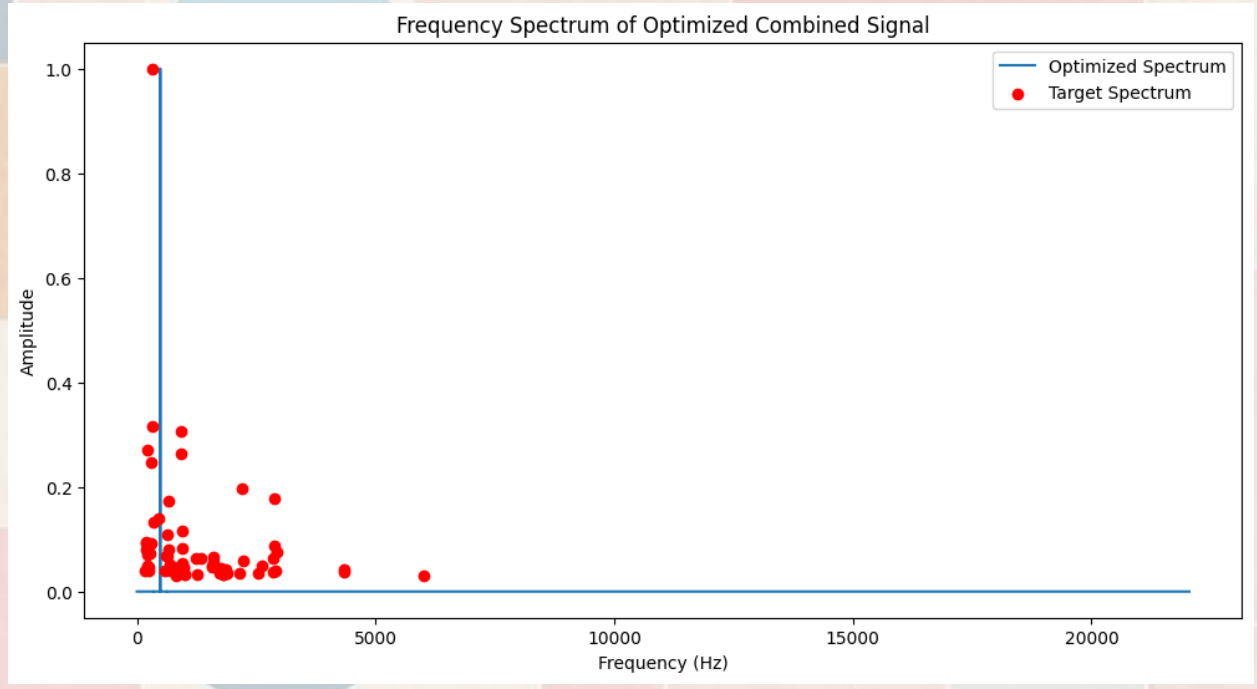
MFCC Distance with Differential Evolution (AM)



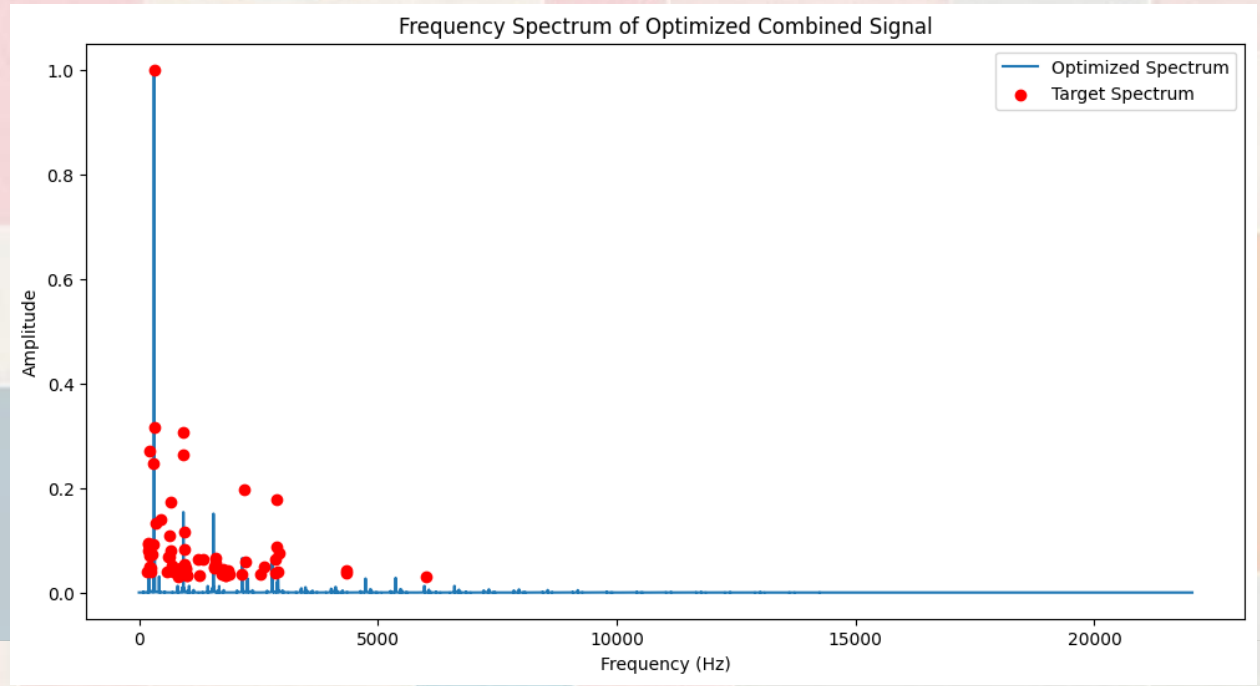
Itakura-Saito Distance
with Dual Annealing (AM)



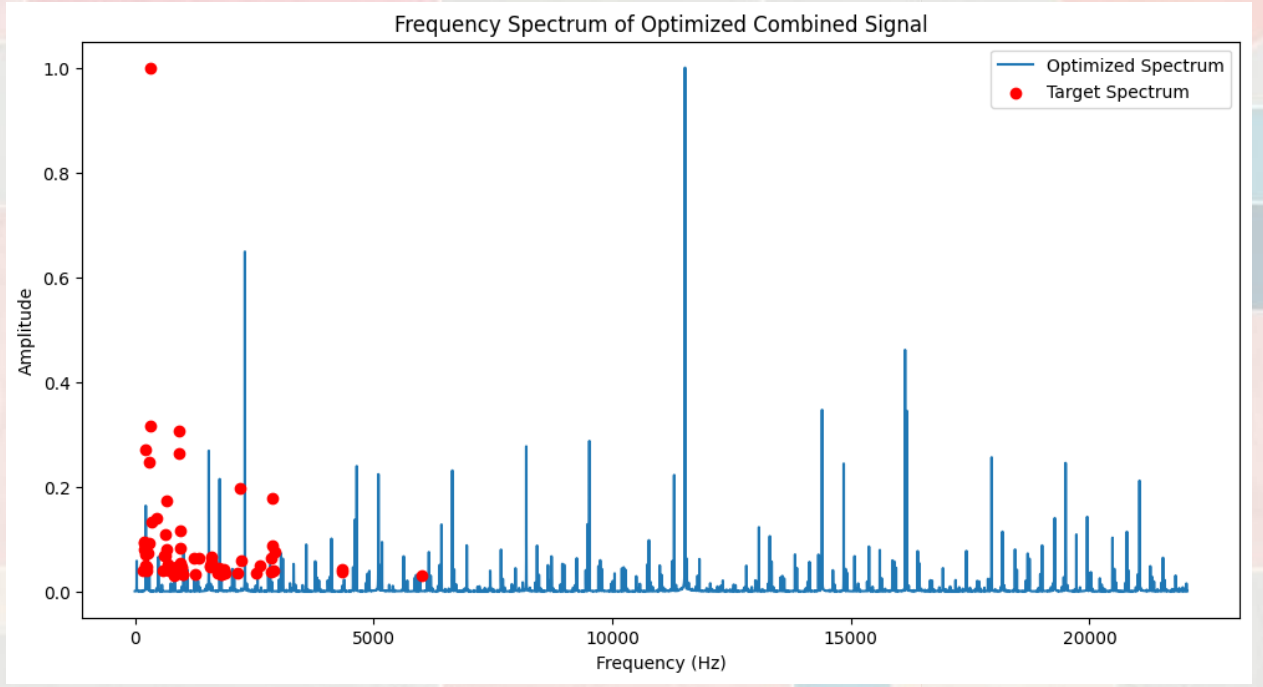
Itakura-Saito Distance
with Basin-hopping (AM)



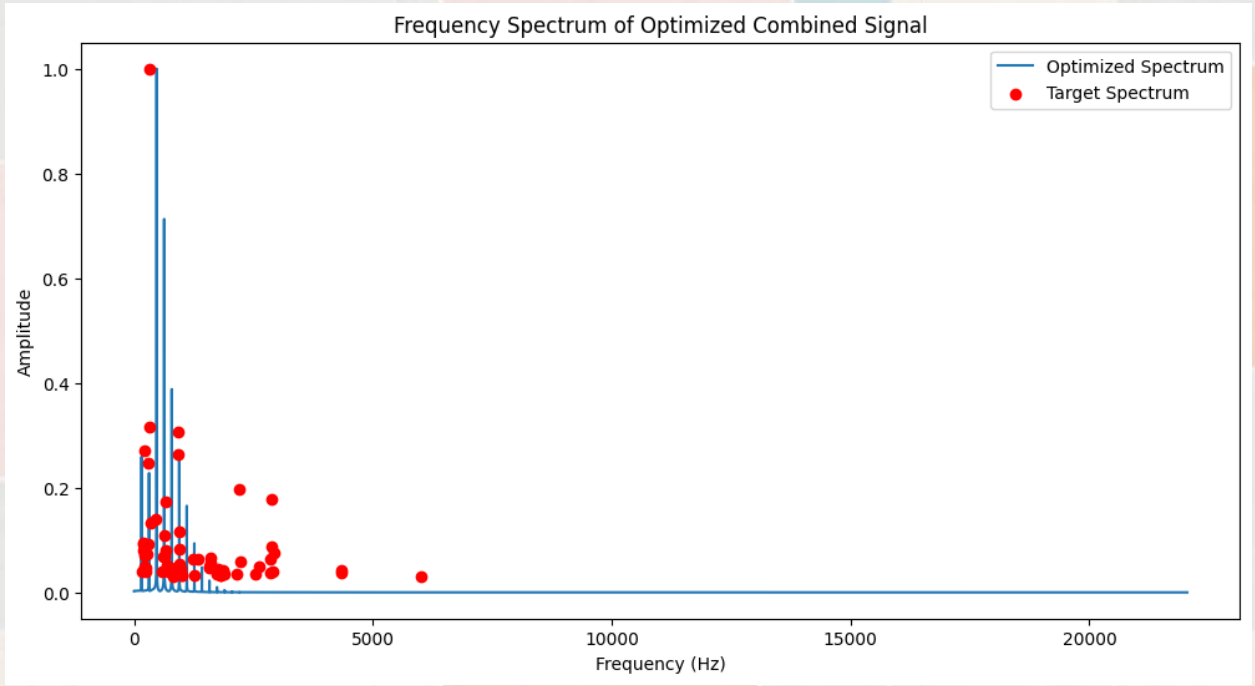
Itakura-Saito Distance with Differential Evolution (FM)



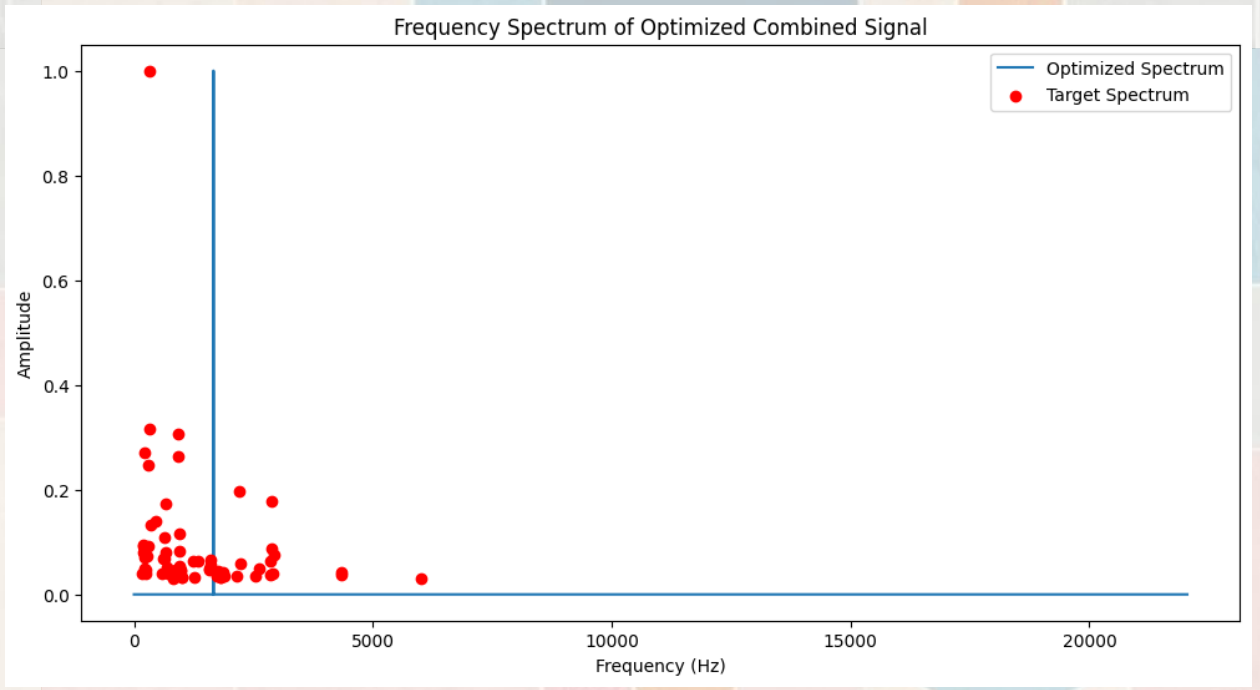
Cosine Similarity with Differential Evolution (FM)



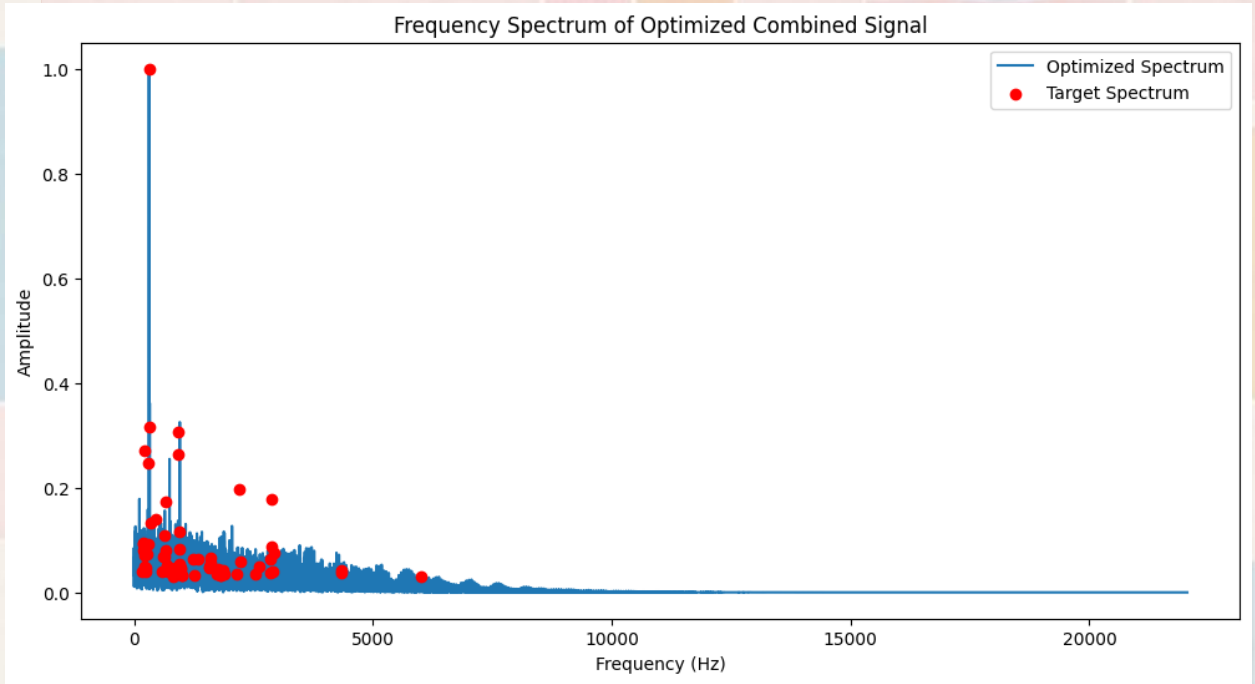
Spectral Convergence Distance with Differential Evolution (FM)



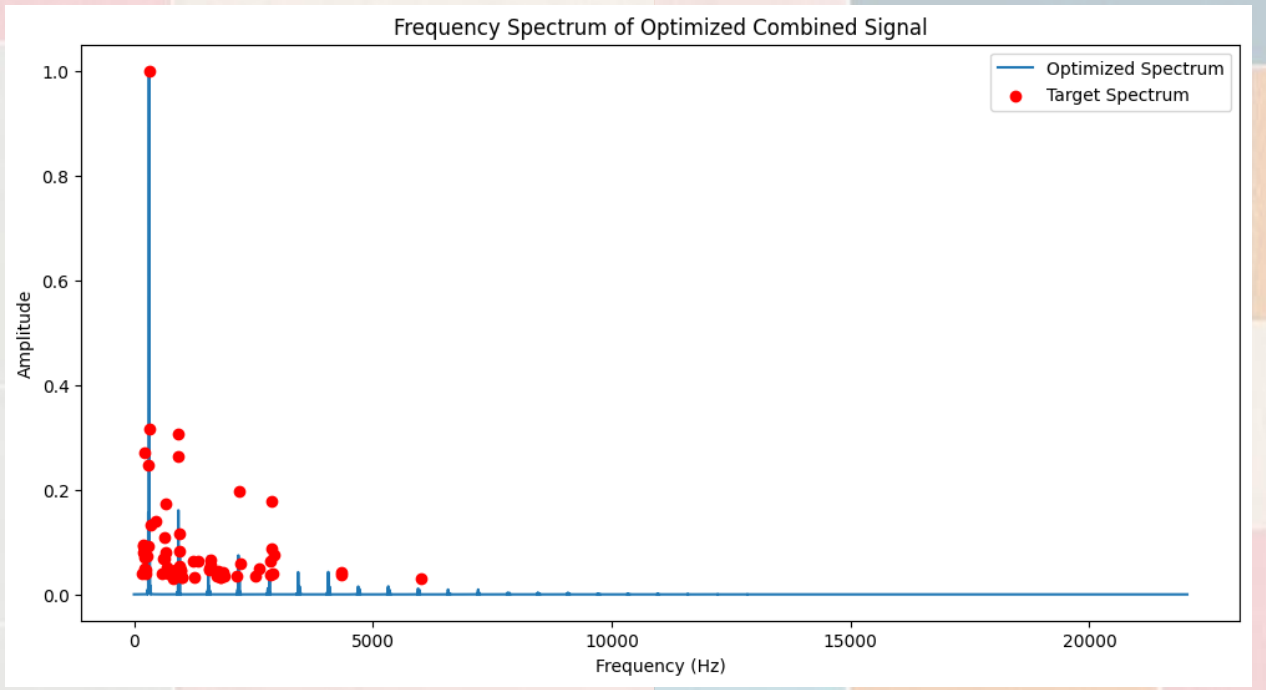
Euclidean Distance with Differential Evolution (FM)



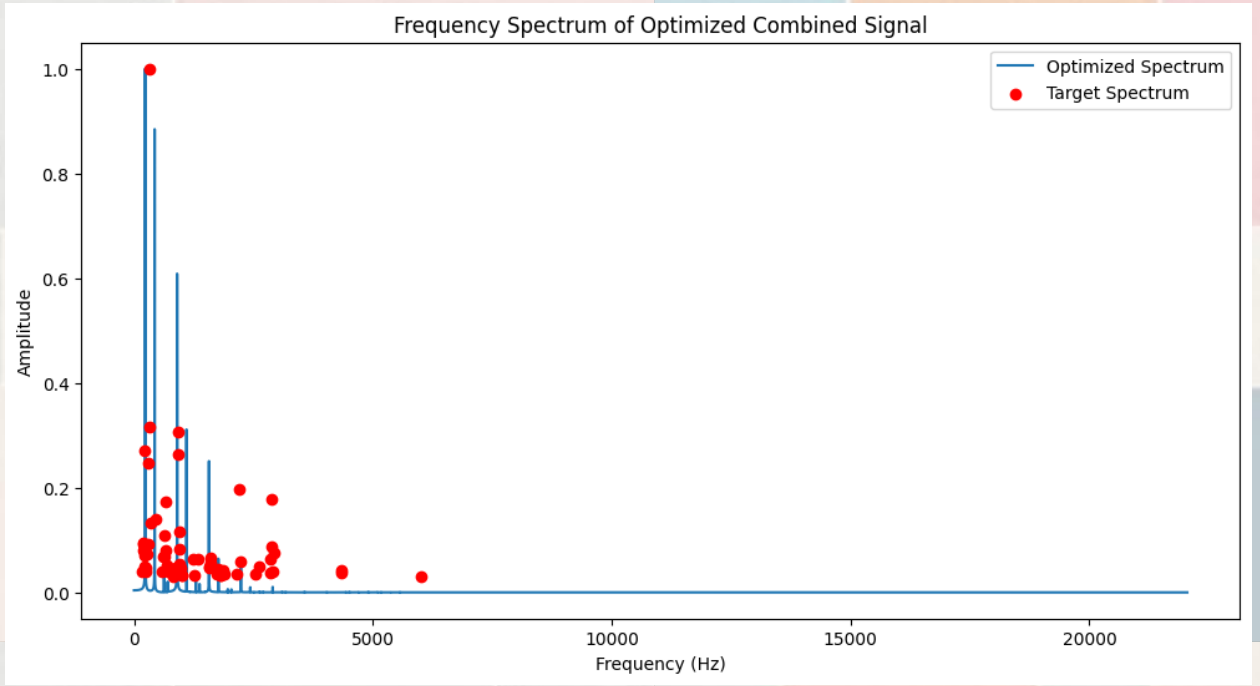
Manhattan Distance with Differential Evolution (FM)



Kullback-Leibler Divergence with Differential Evolution (FM)



Pearson Correlation Coefficient with Differential Evolution (FM)



MFCC Distance with Differential Evolution (FM)

Recorded Cello Sample

Original Audio

Cut (DFT Frame)

Origianal Audiospylt Resynthesis

4 Operator FM Resynthesis (Cosine Similarity)

4 Operator FM Resynthesis (Kullback-Leibler)

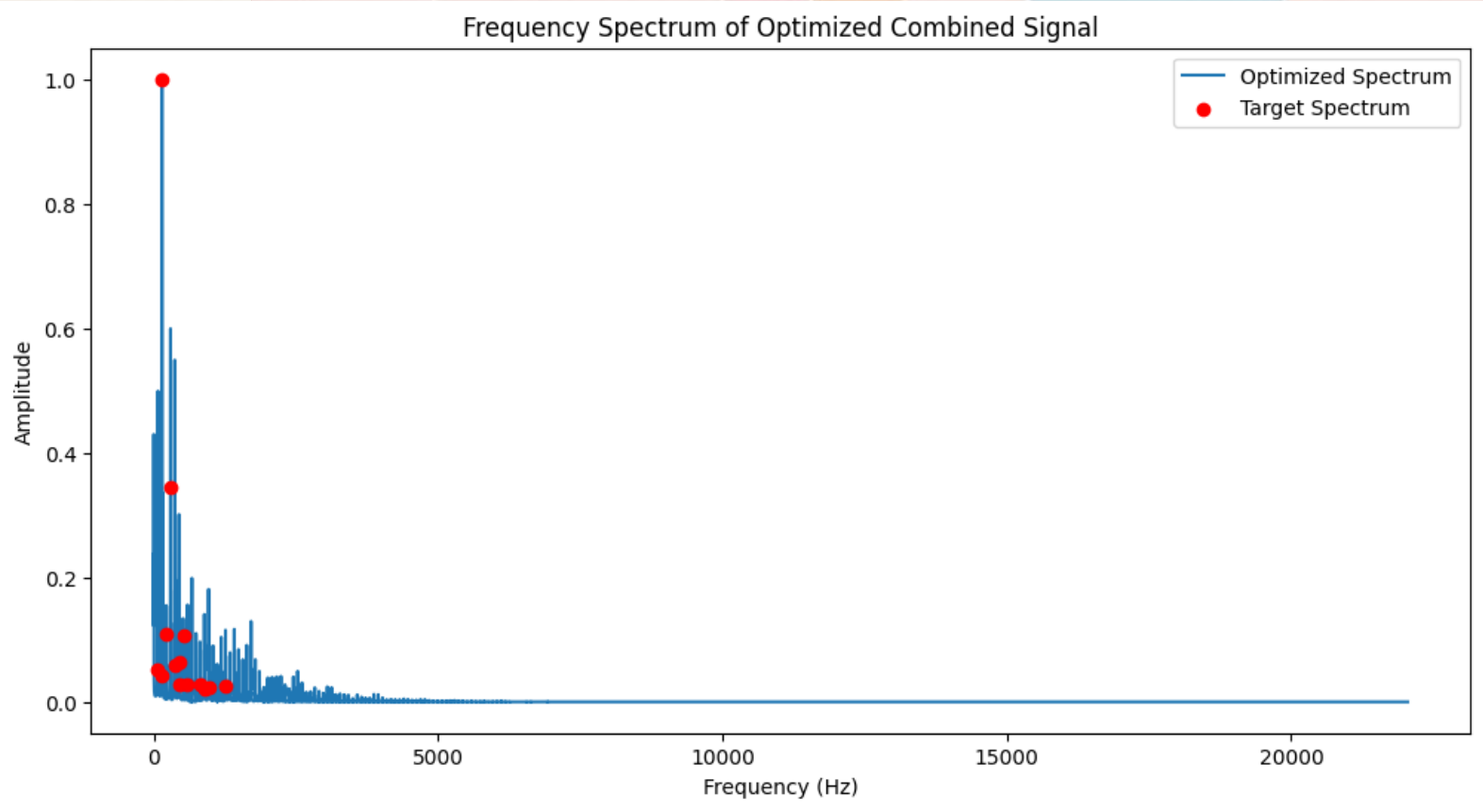
4 Operator AM Resynthesis (Itakura-Saito)

How to Implement the Calculated Values into DAW/Plugins?

Implementation in Ableton

- Ableton's Operator synth consists of 4 operators.
- Replicating Sound in "Fixed" Mode => Use the frequency/amplitude table directly.
- Using MIDI as Input => Remap the frequency/amplitude table using "Coarse" and "Fine" settings => **another optimization problem ☺.**
- No dedicated approach needed due to the simple search method.
- Low resolution of "Fine" settings introduces quantization in the frequency table.
- Works well for AM synthesis, but (was) problematic for FM because of undocumented Index scaling => thankfully somebody on reddit made a comparison between Dexed Arturia DX7V and Operator that lists exact index values as multiplier.

- Creating a Preset from calculated values => ChatGPT is here to help!



	Modulator	Frequency (Hz)	Amplitude
0	1	147.124873	8.669343
1	2	74.849028	3.855502
2	3	520.808977	0.265076
3	4	820.190109	1.141220

	Modulator	Frequency (Hz)	Amplitude	Amplitude (dB)	Waveform	Coarse Tuning	Fine Tuning	Suggested Frequency (Hz)	MIDI Value	Note Name	Base Note
0	Modulator 1	147.124873	8.669343	-2.823901	sine_wave	1.0	0	147.124873	50.034452	D3	Yes
1	Modulator 2	74.849028	3.855502	-9.862007	sine_wave	0.5	17	74.812998	38.326288	D2	No
2	Modulator 3	520.808977	0.265076	-33.116232	sine_wave	2.0	770	520.822052	71.919444	B4	No
3	Modulator 4	820.190109	1.141220	-20.436234	sine_wave	5.0	115	820.221170	79.782113	G5	No

Analyzing Preset Files with ChatGPT-4

- Able to handle complex files, such as Operator presets, which consist of approximately 3300 lines in a gzip compressed XML file.
- Successfully identifies crucial tags for Operator manipulation such as Coarse, Fine, Volume, and Waveform settings.

- Despite Ableton/Operator being closed-source with no available documentation, ChatGPT accurately locates necessary tags.
- Some initial guesses may be incorrect, but reasoning with the model can guide it correctly.

- Analyzing presets works flawlessly for well-documented open-source plugins like Dexed or Surge, as well as several well-known hardware synthesizers like the Yamaha DX7—likely because their openly available documentation was directly used during the training of the model.

- Overall, the use of ChatGPT can significantly shorten the time for analysis of preset structure and adaptation of plugins.

Similar Products and Approaches

Commercial:

- Sonic Charge Synplant 2 with Genopatch
- Micro Music with Vital synthesizer

Open Source:

- **Spiegelib:** https://github.com/spiegelib
- **preset-gen-vae:** https://github.com/gwendal-lv/preset-gen-vae
- **DDX7:** https://github.com/fcaspe/ddx7
- **IRCAM RAVE:** https://github.com/fcaspe/RAVE

To Dos:

- More waveforms, general wave-table support
- Advanced FM implementation => Full Dexed support
- Full Surge support
- GUI and usability improvements (current implementation requires at least basic Python understanding)
- Multi-frame FFT support
- Integration of dx7pytorch for CUDA-based calculations (should be MUCH faster compared to current version)

THANK YOU!!!

egor.polyakov@hmt-leipzig.de
https://github.com/klirr2007