THIN – Sistemi di intonazione musicale / Tuning systems

Version: PHI • Release date: 2025-09-06 • Author: LUCA BIMBI • License: MIT

This manual is bilingual. Italian comes first, English follows below.

ITALIANO

Introduzione

THIN è un programma a riga di comando per generare e confrontare sistemi di intonazione. Consente di: - creare tabelle `cpstun` (GEN -2) in un file `.csd` per Csound; - esportare tabelle del sistema in formato testo (`_system.txt`) e Excel (`_system.xlsx`); - generare tabelle di confronto (`_compare.txt`/`_compare.xlsx`) rispetto a TET (12/24/48), serie armonica e subarmonica; - opzionalmente esportare file AnaMark `.tun`; - opzionalmente eseguire analisi audio (F0 e formanti) tramite `librosa` e allineare i risultati alle tabelle di confronto.

THIN è parte del progetto SIM e deriva concetti organizzativi dal documento SIM-2NV (manual-2nv.md), ma le opzioni e il comportamento riportati qui sono specifici di THIN (thin.py).

Requisiti e installazione

- Python 3.8+
- Moduli standard: argparse, sys, math, re, os, shutil, fractions, typing, threading, time
- Opzioni:
  - Excel export (`*.xlsx`): `openpyxl`
    - installazione: `pip install openpyxl`
  - Analisi audio: `librosa` (consigliato anche `scipy` per un miglior peak picking)
    - installazione: `pip install librosa scipy`

Uso locale: - Tenere `thin.py` in una cartella accessibile; su Unix-like si può renderlo eseguibile: `chmod +x thin.py`.

Uso rapido

Formato generale: - Python: `python3 thin.py [opzioni] OUTPUT_BASE` - Unix-like: `./thin.py [opzioni] OUTPUT_BASE`

Dove `OUTPUT_BASE` è il prefisso dei file generati (senza estensione), per esempio `out`.

Esempio minimo (12-TET su ottava con A4=440 Hz, nota base C4):

```
python3 thin.py out
```

Genera: - `out.csd` (tabella cpstun GEN -2); - `out_system.txt` e, se disponibile `openpyxl`, anche `out_system.xlsx`; - `out_compare.txt` e, se disponibile `openpyxl`, anche `out_compare.xlsx`.

Opzioni utili: - `--export-tun` per esportare anche `out.tun` (AnaMark TUN); - `--basenote A4 --diapason 442` per cambiare la nota di riferimento e il diapason; - `--span 3` per ripetere la serie su 3 intervalli; - `--audio-file voce.wav --analysis lpc` per analisi audio con indicatore di avanzamento.

Parametri della riga di comando (CLI)

Nota: i default sono tra parentesi.

Tabelle riepilogative dei parametri CLI

Base

Parametro	Argomenti	Tipo input	Obbligatorio	Default	Descrizione
`--lang`	`{it,en}`	scelta	No	`it`	Lingua dell'interfaccia
`-v, --version`	-	flag	No	-	Stampa la versione del programma
`--diapason`	`<Hz>`	float (Hz)	No	`440.0`	Diapason A4 in Hertz
`--basekey`	`<MIDI>`	int (0..127)	No	`60`	Nota MIDI di partenza per la tabella
`--basenote`	`<nome|Hz>`	nota (es. A4, F#2, Bb3) oppure float (Hz)	No	`C4`	Nota/frequenza di riferimento; microtoni: `+`=+50c, `-`=-50c, `!`=+25c, `.`=-25c

Sistemi di intonazione (sceglierne uno; opzionali)

Parametro	Argomenti	Tipo input	Obbligatorio	Default	Descrizione
`--et`	`INDEX INTERVAL`	INDEX: int (>0); INTERVAL: int/frazione/float (cents o frazione→cents)	No	`12 1200`	Temperamento equabile sull'intervallo specificato
`--geometric`	`GEN STEPS INTERVAL`	GEN: int/frazione/float (>0); STEPS: int (>0); INTERVAL: rapporto (float/frazione) oppure cents (int o suffisso `c`)	No	-	Progressione geometrica con riduzione nell'intervallo indicato salvo `--no-reduce`
`--natural`	`A_MAX B_MAX`	int, int (>=0)	No	-	Sistema naturale 4:5:6 con riduzione all'ottava salvo `--no-reduce`
`--danielou`	`a,b,c`	tre interi (negativi consentiti; usare virgolette)	No	-	Aggiunge rapporti del sistema Danielou; opzione ripetibile
`--danielou-all`	-	flag	No	-	Genera la griglia completa Danielou

Opzioni aggiuntive

Parametro	Argomenti	Tipo input	Obbligatorio	Default	Descrizione
`--no-reduce`	-	flag	No	-	Non ridurre all'ottava/intervallo
`--span`, `--ambitus`	`N`	int (>=1)	No	`1`	Ripete la serie su N intervalli
`--interval-zero`	-	flag	No	-	Forza `interval=0` nella tabella cpstun (usa la serie non ripetuta)
`--export-tun`	-	flag	No	-	Esporta file `.tun` (AnaMark TUN)
`--tun-integer`	-	flag	No	-	`.tun`: arrotonda i cents all'intero più vicino

Confronto

Parametro	Argomenti	Tipo input	Obbligatorio	Default	Descrizione
`--compare-fund`	`[<nome|Hz>]`	nota o float (Hz)	No	`basenote`	Fondamentale per confronto; senza argomento usa la basenote
`--compare-tet`	`{12,24,48}`	scelta	No	`12`	Divisioni del TET per confronto
`--compare-tet-align`	`{same,nearest}`	scelta	No	`same`	Allineamento del TET
`--subharm-fund`	`<nome|Hz>`	nota o float (Hz)	No	`A5`	Fondamentale subarmonica
`--midi-truncate`	-	flag	No	-	Tronca la serie al range MIDI 0..127

Analisi audio (librosa)

Parametro	Argomenti	Tipo input	Obbligatorio	Default	Descrizione
`--audio-file`	`<wav>`	path file	No	-	File WAV da analizzare (richiede `librosa`)
`--analysis`	`{lpc,specenv}`	scelta	No	`lpc`	Metodo di stima formanti (e F0)
`--frame-size`	`<int>`	int	No	`1024`	Dimensione frame
`--hop-size`	`<int>`	int	No	`512`	Hop size
`--lpc-order`	`<int>`	int	No	`12`	Ordine LPC
`--window`	`{hamming,hanning}`	scelta	No	`hamming`	Tipo finestra

Output finale

Parametro	Argomenti	Tipo input	Obbligatorio	Default	Descrizione
`OUTPUT_BASE`	-	stringa posizionale	No	`out`	Nome base dei file generati

Comportamento e file generati

- Banner iniziale: il programma stampa sempre Nome, Versione, Data, Autore, Licenza.
- Help: è paginato entro 80 righe. Prompt `--More--` (Invio=continua, q=esci).
- Riepilogo: prima delle tabelle viene mostrato un riepilogo dei parametri selezionati (in lingua IT/EN).
- Tabelle `System`: colonne Step, MIDI, Ratio, Hz; ordinamento per Hz crescente.
- Tabelle `Compare`: per ciascun passo mostra colonne Custom_Hz, Harmonic_Hz, DeltaHz_Harm, Subharm_Hz, DeltaHz_Sub, TET_Hz, TET_Note, DeltaHz_TET e, se presenti, AudioF0_Hz, AudioFormant_Hz, Formant_RelAmp.
- Serie armonica: max 10 kHz; serie subarmonica: min 16 Hz; l'allineamento avviene rispetto alle frequenze del sistema custom.
- Excel: richiede `openpyxl`; se non presente, l'export `.xlsx` viene saltato (messaggio a schermo localizzato).
- Analisi audio: durante l'analisi appare un indicatore di avanzamento; in caso di fallimento, le tabelle sono comunque generate senza le colonne audio.

Esempi

- ET 12 su ottava, confronto 12-TET, output base `out`:

```
./thin.py --et 12 1200 out
```

- Geometrico: generatore 3/2, 12 passi, intervallo 2/1; confronto 48-TET:

```
./thin.py --geometric 3/2 12 2/1 --compare-tet 48 out_geo
```

- Danielou (griglia completa), diapason 442 Hz, esporta .tun:

```
./thin.py --danielou-all --diapason 442 --export-tun out_dan
```

- Analisi audio su file voce, metodo LPC, base C+4:

```
./thin.py --basenote C+4 --audio-file voce.wav --analysis lpc out_voice
```

Note e limitazioni

- L'analisi F0/formanti è euristica e dipende molto dal materiale audio; per risultati stabili su voce, preferire LPC.
- La conversione frazione→cents per `--et` limita il denominatore (fino a 10000) per evitare frazioni troppo fini.
- La tabella cpstun inserisce voci in ordine crescente e aggiorna automaticamente il numero di funzione `f N`.
- Le note TET mostrate in colonna sono calcolate per labeling 12-TET rispetto al diapason.

Licenza e crediti

Ringraziamenti: `openpyxl` per export Excel; `librosa` e (opz.) `scipy` per analisi.

ENGLISH

Introduction

THIN is a command-line tool to generate and compare tuning systems. It can: - create `cpstun` (GEN -2) tables into a `.csd` file for Csound; - export system tables to text (`_system.txt`) and Excel (`_system.xlsx`); - build comparison tables (`_compare.txt`/`_compare.xlsx`) against TET (12/24/48), the harmonic and the subharmonic series; - optionally export AnaMark `.tun` files; - optionally run audio analysis (F0 and formants) via `librosa` and align results within comparison tables.

It belongs to the SIM project and adopts ideas from SIM-2NV (manual-2nv.md), but the options and behavior described here are specific to THIN (thin.py).

Requirements and installation

- Python 3.8+
- Standard modules: argparse, sys, math, re, os, shutil, fractions, typing, threading, time
- Optional:
  - Excel export (`*.xlsx`): `openpyxl`
    - install: `pip install openpyxl`
  - Audio analysis: `librosa` (and preferably `scipy`)
    - install: `pip install librosa scipy`

Local use: - Keep `thin.py` in a convenient folder; on Unix-like systems you can make it executable: `chmod +x thin.py`.

Quick start

General form: - Python: `python3 thin.py [options] OUTPUT_BASE` - Unix-like: `./thin.py [options] OUTPUT_BASE`

`OUTPUT_BASE` is the prefix used for generated files, e.g., `out`.

Minimal example (12-TET on octave, A4=440 Hz, basenote C4):

```
python3 thin.py out
```

This produces: - `out.csd` (cpstun GEN -2 table); - `out_system.txt` and, if `openpyxl` is available, also `out_system.xlsx`; - `out_compare.txt` and, if `openpyxl` is available, also `out_compare.xlsx`.

Useful options: - `--export-tun` to also export `out.tun` (AnaMark TUN); - `--basenote A4 --diapason 442` to set reference note and diapason; - `--span 3` to repeat the series over 3 intervals; - `--audio-file voice.wav --analysis lpc` to run audio analysis with a progress indicator.

Command line parameters (CLI)

Defaults in parentheses.

Summary tables of CLI parameters

Basics

Parameter	Arguments	Input type	Required	Default	Description
`--lang`	`{it,en}`	choice	No	`it`	Interface language
`-v, --version`	-	flag	No	-	Print program version
`--diapason`	`<Hz>`	float (Hz)	No	`440.0`	A4 reference in Hertz
`--basekey`	`<MIDI>`	int (0..127)	No	`60`	Starting MIDI note for the table
`--basenote`	`<name|Hz>`	note (e.g., A4, F#2, Bb3) or float (Hz)	No	`C4`	Reference note/frequency; microtones: `+`=+50c, `-`=-50c, `!`=+25c, `.`=-25c

Tuning systems (pick one; optional)

Parameter	Arguments	Input type	Required	Default	Description
`--et`	`INDEX INTERVAL`	INDEX: int (>0); INTERVAL: int/fraction/float (cents or fraction→cents)	No	`12 1200`	Equal temperament on the specified interval
`--geometric`	`GEN STEPS INTERVAL`	GEN: int/fraction/float (>0); STEPS: int (>0); INTERVAL: ratio (float/fraction) or cents (int or `c` suffix)	No	-	Geometric progression reduced within the given interval unless `--no-reduce`
`--natural`	`A_MAX B_MAX`	int, int (>=0)	No	-	Natural system 4:5:6 with octave reduction unless `--no-reduce`
`--danielou`	`a,b,c`	three integers (negatives allowed; use quotes)	No	-	Adds Danielou system ratios; repeatable option
`--danielou-all`	-	flag	No	-	Generate the full Danielou grid

Additional options

Parameter	Arguments	Input type	Required	Default	Description
`--no-reduce`	-	flag	No	-	Do not reduce to octave/interval
`--span`, `--ambitus`	`N`	int (>=1)	No	`1`	Repeat the series across N intervals
`--interval-zero`	-	flag	No	-	Force `interval=0` in cpstun (uses non-repeated series)
`--export-tun`	-	flag	No	-	Export `.tun` (AnaMark TUN) file
`--tun-integer`	-	flag	No	-	`.tun`: round cents to nearest integer

Comparison

Parameter	Arguments	Input type	Required	Default	Description
`--compare-fund`	`[<name|Hz>]`	note or float (Hz)	No	`basenote`	Fundamental for comparison; if omitted, basenote is used
`--compare-tet`	`{12,24,48}`	choice	No	`12`	TET divisions for comparison
`--compare-tet-align`	`{same,nearest}`	choice	No	`same`	TET alignment
`--subharm-fund`	`<name|Hz>`	note or float (Hz)	No	`A5`	Subharmonic fundamental
`--midi-truncate`	-	flag	No	-	Truncate series to MIDI 0..127

Audio analysis (librosa)

Parameter	Arguments	Input type	Required	Default	Description
`--audio-file`	`<wav>`	file path	No	-	WAV file to analyze (requires `librosa`)

Parameter	Arguments	Input type	Required	Default	Description
`--analysis`	`{lpc,specenv}`	choice	No	`lpc`	Formant (and F0) estimation method
`--frame-size`	`<int>`	int	No	`1024`	Frame size
`--hop-size`	`<int>`	int	No	`512`	Hop size
`--lpc-order`	`<int>`	int	No	`12`	LPC order
`--window`	`{hamming,hanning}`	choice	No	`hamming`	Window type

Final output

Parameter	Arguments	Input type	Required	Default	Description
`OUTPUT_BASE`	-	positional string	No	`out`	Base name for generated files

Behavior and generated files

- Startup banner: the program always prints Name, Version, Date, Author, License.
- Help: paginated within 80 rows. Prompt `--More--` (Enter=continue, q=quit).
- Summary: before the tables, a localized run summary is printed.
- System tables: Step, MIDI, Ratio, Hz; sorted by increasing Hz.
- Comparison tables: per step show Custom_Hz, Harmonic_Hz, DeltaHz_Harm, Subharm_Hz, DeltaHz_Sub, TET_Hz, TET_Note, DeltaHz_TET and, if available, AudioF0_Hz, AudioFormant_Hz, Formant_RelAmp.
- Harmonics: cutoff at 10 kHz; subharmonics: cutoff at 16 Hz; alignment is performed against the custom system frequencies.
- Excel requires `openpyxl`; if missing, `.xlsx` export is skipped with a localized message.
- Audio analysis: while running, a progress indicator is shown; if analysis fails, tables are generated without audio columns.

Examples

- 12-TET on octave, comparison with 12-TET, base output `out`:

```
./thin.py --et 12 1200 out
```

- Geometric: generator 3/2, 12 steps, interval 2/1; comparison with 48-TET:

```
./thin.py --geometric 3/2 12 2/1 --compare-tet 48 out_geo
```

- Danielou (full grid), diapason 442 Hz, also export .tun:

```
./thin.py --danielou-all --diapason 442 --export-tun out_dan
```

- Audio analysis on a voice file, LPC method, basenote C+4:

```
./thin.py --basenote C+4 --audio-file voice.wav --analysis lpc out_voice
```

Notes and limitations

- F0/formant analysis is heuristic and content-dependent; for voice, LPC often gives more stable results.
- Fraction→cents for `--et` limits denominator to avoid extremely fine fractions.
- The cpstun table is written with ratios in increasing order and `f N` numbering is auto-incremented.
- TET note labels are computed as 12-TET pitch names relative to the diapason.

License and credits

Credits: `openpyxl` for Excel export; `librosa` and (optional) `scipy` for analysis.