SIM - Sistemi intonazione musicale

Documentazione Completa v1.0

Autore: LUCA BIMBI
Data: 29 Agosto 2025
Versione: 1.0

Indice

Introduzione

`SIM - Sistemi intonazione musicale` è un generatore avanzato di parametri e rapporti per la costruzione di tabelle di intonazione musicale. Il programma è particolarmente utile per compositori e ricercatori che lavorano con sistemi di intonazione non standard in Csound attraverso l'opcode `cpstun`.

Scopo Principale

Il software genera: - Tabelle di accordatura per Csound (formato GEN -2 non normalizzato) - File TUN per software compatibili (AnaMark .TUN) - Tabelle comparative con 12-TET e serie armonica - Esportazioni in formato testo ed Excel

Sistemi Supportati

1. **Temperamento Equabile (ET)** - Divisione personalizzabile di un intervallo
2. **Sistema Geometrico** - Progressione geometrica basata su generatore
3. **Intonazione Naturale (4:5:6)** - Sistema basato su rapporti armonici
4. **Sistema Danielou** - Intonazione microtonale secondo Danielou (1907)

Requisiti e Installazione

Requisiti Minimi

- **Python:** 3.10 o superiore (richiesto per l'uso di `|` nelle annotazioni di tipo)
- **Moduli Standard Python:**
  - `argparse`
  - `sys`
  - `math`
  - `fractions`
  - `re`
  - `os`
  - `typing`

  Nota: Per Python < 3.10, è necessario modificare le annotazioni di tipo sostituendo `float | Fraction` con `Union[float, Fraction]` e importando `Union` da `typing`.

Dipendenze Opzionali

- **openpyxl:** Per esportazione in formato Excel (.xlsx)

  ```
  pip install openpyxl
  ```

Installazione

1. Salvare il file `sim.py` in una directory locale
2. Rendere il file eseguibile (su sistemi Unix-like):

   ```
   chmod +x sim.py
   ```

3. Installare le dipendenze opzionali se necessario

Caratteristiche Principali

1. Generazione Tabelle Csound

Crea automaticamente tabelle `cpstun` in formato GEN -2 con: - Numero di gradi (numgrades) - Intervallo di ripetizione (interval) - Frequenza base (basefreq) - Nota MIDI base (basekey) - Rapporti di frequenza (ratios)

2. Conversione Note

- Supporta nomi di nota (es. "A4", "F#2", "Bb3")
- Conversione automatica in frequenze Hz
- Diapason personalizzabile (default: 440 Hz)

3. Riduzione all'Ottava

- Opzione per ridurre i rapporti nell'intervallo [1, 2)
- Flag `--no-reduce` per mantenere rapporti non ridotti

4. Esportazione Multipla

- File .csd (Csound)
- File .tun (AnaMark TUN)
- Tabelle comparative .txt e .xlsx
- Tabelle di sistema con Step, MIDI, Ratio, Hz

5. Analisi Comparativa

Confronto automatico con: - 12-TET (temperamento equabile a 12 toni) - Serie armonica - Calcolo delle differenze in Hz - Evidenziazione delle prossimità (< 17 Hz)

Opzioni di confronto: - `--compare-fund`: imposta la fondamentale per la serie armonica e per l'ancoraggio della griglia 12TET (default: `--basenote`). - `--compare-tet-align {same, nearest}`: - `same`: il 12TET parte dalla fondamentale di confronto e avanza di semitono in semitono (i = step). - `nearest`: per ogni valore Custom viene scelto il grado 12TET più vicino nella griglia ancorata alla fondamentale di confronto.

Parametri della Riga di Comando

Parametri Generali

Parametro	Tipo	Default	Descrizione
`--basenote`	nota/Hz	440	Nota di riferimento (es. "A4") o frequenza in Hz
`--basekey`	int	60	Nota MIDI base per tabella cpstun (60 = C4)
`--diapason`	float	440	Diapason di riferimento in Hz
`-v, --version`	-	-	Visualizza versione del programma
`--export-tun`	flag	-	Esporta file .tun con 128 note MIDI
`--no-reduce`	flag	-	Non riduce i rapporti all'ottava

Parametro	Tipo	Default	Descrizione
`--compare-fund`	nota/Hz	basenote	Fondamentale per confronto serie armonica e ancoraggio 12TET
`--compare-tet-align`	enum	"same"	Allineamento 12TET: `same` dalla fondamentale o `nearest` nota più vicina
`output_file`	string	richiesto	Nome base per i file di output

Parametri Sistema

Temperamento Equabile

`-et INDEX CENTS`

- `INDEX`: Indice della radice (es. 12 per 12-TET)
- `CENTS`: Ampiezza in cents o frazione (es. 1200 o 3/2)

Sistema Geometrico

`--geometric GENERATORE PASSI`

- `GENERATORE`: Rapporto generatore (intero, frazione o float)
- `PASSI`: Numero di passi (> 0)

Sistema Naturale

`--natural A_MAX B_MAX`

- `A_MAX`: Esponente massimo per (3/2)^a
- `B_MAX`: Esponente massimo per (5/4)^b

Sistema Danielou

```
--danielou          # Sottoinsieme predefinito
--danielou-all      # Griglia completa (fino a 53 rapporti)
```

Sistemi di Accordatura

1. Temperamento Equabile (ET)

Formula: `r = (2^(cents/1200))^(1/index)`

Divide un intervallo (solitamente l'ottava) in parti uguali su scala logaritmica.

Esempio 12-TET:

```
python3 sim.py -et 12 1200 output_12tet
```

2. Sistema Geometrico

Formula: `ratio[i] = generatore^i`

Genera una progressione geometrica basata su un rapporto generatore.

Caratteristiche: - Riduzione all'ottava attiva di default - Supporta generatori razionali (es. 3/2 per quinte) - Opzione `--no-reduce` per mantenere progressione crescente

Esempio Ciclo delle Quinte:

```
python3 sim.py --geometric 3/2 12 output_quinte
```

3. Sistema Naturale (4:5:6)

Formula: `ratio = ((3/2)^a) * ((5/4)^b)`

Dove: - `a -> [-A_MAX, A_MAX]` - `b -> [-B_MAX, B_MAX]`

Genera rapporti basati su terze maggiori (5/4) e quinte giuste (3/2).

Esempio:

```
python3 sim.py --natural 3 3 output_natural
```

4. Sistema Danielou

Formula: `ratio = ((6/5)^a) * ((3/2)^b) * (2^c)`

Dove `c` è usato per la riduzione all'ottava.

Due modalità:

1. **Sottoinsieme (default):**
   - Tonica (1/1)
   - Asse delle quinte (a=0, b=-5..5)
   - Tre terze minori armoniche successive
   - Tre seste maggiori armoniche successive
2. **Griglia completa (`--danielou-all`):**
   - `a -> [-3, 3]`
   - `b -> [-5, 5]`
   - Fino a 53 rapporti dopo riduzione

Esempio:

```
# Sottoinsieme
python3 sim.py --danielou output_danielou

# Griglia completa
python3 sim.py --danielou-all output_danielou_full
```

File di Output

1. File Csound (.csd)

Formato: `<output_file>.csd`

Contiene tabelle cpstun in formato GEN -2:

```
;            numgrades basefreq rapporti-di-intonazione .......
;               interval basekey
; tabella cpstun generata automaticamente | basekey=60 basefrequency=440.000000Hz
f 1 0 16 -2  12 2.0 440.0 60  1.0 1.059463 1.122462 ...
```

Struttura dati: - `numgrades`: numero di rapporti - `interval`: 2.0 per ottava, 0.0 se non definibile - `basefreq`: frequenza base in Hz - `basekey`: nota MIDI di riferimento - `ratios`: lista dei rapporti di frequenza

2. File TUN (.tun)

Formato: `<output_file>.tun`

File AnaMark TUN con mappatura di tutte le 128 note MIDI:

```
[Tuning]
FormatVersion=1
Name=Generated by SIM - Sistemi intonazione musicale 1.0

[Exact Tuning]
; basekey=60 basefrequency=440.000000Hz
Note 0=8.1757989156 Hz
Note 1=8.6619572180 Hz
...
Note 127=12543.8539514 Hz
```

3. Tabelle di Sistema

File testo: `<output_file>_system.txt`

```
Step    MIDI    Ratio           Hz
0       60      1.0000000000    440.000000
1       61      1.0594630944    466.163762
...
```

Nota: le colonne sono allineate a larghezza fissa (padding con spazi) per migliorare la leggibilità.

File Excel: `<output_file>_system.xlsx` (se openpyxl installato) - Formattazione con header in grassetto - Sfondo grigio per intestazioni

4. Tabelle Comparative

File testo: `<output_file>_compare.txt`

```
Step    MIDI    Ratio           Custom_Hz    Harmonic_Hz    DeltaHz_Harm    TET_Hz    DeltaH
0       60      1.0000000000    440.000000   440.000000     0.000000        440.000   0.0000
...
```

Nota: anche per il confronto le colonne sono allineate a larghezza fissa (padding con spazi) e quando Custom e Harmonic sono molto vicini viene aggiunto il simbolo “ ”.

File Excel: `<output_file>_compare.xlsx` (se openpyxl installato) - Colori distintivi: - Rosso chiaro: Custom - Verde chiaro: Harmonic - Blu chiaro: TET - Grassetto per differenze < 1 Hz tra Custom e Harmonic

---

Esempi di Utilizzo

Esempio 1: 12-TET Standard

```
python3 sim.py \
    --basekey 60 \
    --basenote A4 \
    --diapason 440 \
    -et 12 1200 \
    output_12tet
```

Esempio 2: Scala Pitagorica (12 quinte)

```
python3 sim.py \
    --basekey 60 \
    --basenote 261.63 \
    --geometric 3/2 12 \
    output_pythagorean
```

Esempio 3: Sistema Naturale Esteso

```
python3 sim.py \
    --basekey 48 \
    --basenote C3 \
    --natural 5 5 \
```

```
    --export-tun \
    output_natural_extended
```

Esempio 4: Danielou con Export Completo

```
python3 sim.py \
    --basekey 60 \
    --basenote 440 \
    --danielou-all \
    --export-tun \
    output_danielou_complete
```

Esempio 5: Progressione Geometrica Non Ridotta

```
python3 sim.py \
    --basekey 60 \
    --basenote 440 \
    --geometric 4/3 8 \
    --no-reduce \
    output_fourths_unreduced
```

Dettagli Tecnici

Conversione Note-MIDI

Il programma converte automaticamente i nomi delle note in valori MIDI:

Nota	MIDI	Frequenza (A4=440)
C4	60	261.63 Hz
A4	69	440.00 Hz
C5	72	523.25 Hz

Formato supportato: - Note naturali: C, D, E, F, G, A, B - Alterazioni: # (diesis), b (bemolle) - Ottave: 0-9

Conversione Cents-Rapporti

Da rapporto a cents:

```
cents = log(ratio) * 1200 / log(2)
```

Da cents a rapporto:

```
ratio = 2^(cents/1200)
```

Riduzione all'Ottava

Algoritmo per ridurre un rapporto nell'intervallo [1, 2):

```
while ratio >= 2.0:
    ratio /= 2.0
while ratio < 1.0:
    ratio *= 2.0
```

Gestione Frazioni

Il programma supporta l'input di frazioni razionali: - Input: `3/2`, `7/4`, `16/9` - Conversione automatica in float per calcoli - Precisione mantenuta dove possibile

---

Formule Matematiche

Temperamento Equabile

Per un temperamento a `n` divisioni di un intervallo di `c` cents:

$$r_k = 2^{\frac{k \cdot c}{n \cdot 1200}}$$

dove `k` è l'indice del grado (0 k < n).

Sistema Geometrico

Per un generatore `g` e `n` passi:

$$r_k = g^k$$

Con riduzione all'ottava:

$$r'_k = g^k \cdot 2^{-\lfloor \log_2(g^k) \rfloor}$$

Sistema Naturale

$$r_{a,b} = \left(\frac{3}{2}\right)^a \cdot \left(\frac{5}{4}\right)^b$$

dove: - a -> [-A_max, A_max] - b -> [-B_max, B_max]

Sistema Danielou

$$r_{a,b} = \left(\frac{6}{5}\right)^a \cdot \left(\frac{3}{2}\right)^b \cdot 2^c$$

dove `c` è scelto per mantenere r nell'intervallo [1, 2).

Note e Limitazioni

Limitazioni Correnti

1. **Python 3.10+:** Richiesto per le annotazioni di tipo con `|` (union types)
2. **Range MIDI:** 0-127 (standard MIDI)
3. **Precisione:** Float a 64 bit per i calcoli
4. **Tolleranza duplicati:** 1e-9 per confronto rapporti
5. **File Excel:** Richiede installazione separata di openpyxl

Considerazioni per l'Uso

1. **File esistenti:** Il programma appende nuove tabelle ai file .csd esistenti
2. **Normalizzazione:** I rapporti sono sempre ordinati in modo crescente
3. **Deduplicazione:** Rapporti molto vicini ($< 1e\text{-}9$) vengono considerati identici

Compatibilità

- **Python:** 3.10+ richiesto (3.6+ con modifiche al codice)
- **Csound:** Compatibile con tutte le versioni che supportano cpstun
- **AnaMark:** File .tun compatibili con AnaMark 3.0+
- **Excel:** Richiede Excel 2007+ per file .xlsx

Compatibilità con Python < 3.10 Per utilizzare il programma con versioni di Python precedenti alla 3.10, è necessario modificare le annotazioni di tipo:

```python
# Python 3.10+
from fractions import Fraction

def reduce_to_octave(value: Fraction | float):
    pass

# Python 3.6-3.9
from typing import Union

def reduce_to_octave(value: Union[Fraction, float]):
    pass
```

Le funzioni da modificare sono: - `normalize_ratios()` - `pow_fraction()` - `reduce_to_octave()` - `build_natural_ratios()` - `build_danielou_ratios()` - `ratio_et()`

Best Practices

1. **Backup:** Sempre fare backup dei file .csd prima di appendere nuove tabelle
2. **Naming:** Usare nomi descrittivi per i file di output
3. **Documentazione:** Annotare i parametri usati per ogni generazione

---

Risoluzione Problemi

Errori Comuni

"TypeError: unsupported operand type(s) for |" - Python < 3.10 rilevato - Aggiornare a Python 3.10+ o modificare le annotazioni di tipo (vedi sezione Compatibilità)

"Nome nota non valido" - Verificare formato: nota + alterazione + ottava (es. "C#4")

"File non trovato" - Verificare percorso e permessi di scrittura

"openpyxl non installato" - Installare con: `pip install openpyxl`

Debug

Per debug dettagliato, modificare il codice aggiungendo:

```
import logging
logging.basicConfig(level=logging.DEBUG)
```

Changelog

Versione 1.0 (29 Agosto 2025)

- Release iniziale
- Supporto per ET, geometrico, naturale, Danielou
- Export multi-formato
- Tabelle comparative
- Documentazione completa
- **Requisiti:** Python 3.10+ per le annotazioni di tipo con `|`

Licenza e Crediti

Autore: LUCA BIMBI
Email: [contatto autore]
Anno: 2025

Riferimenti

1. **Csound Reference Manual** - Documentazione opcode cpstun
2. **Csound. Guida al sound design in 20 lezioni** corso su Csound
3. **I numeri della musica**, Walter Branchi, pan edizioni

Appendice: Quick Reference

Comandi Rapidi

```
# 12-TET standard
./sim.py -et 12 1200 out

# Scala pitagorica
./sim.py --geometric 3/2 12 out

# Sistema naturale 4:5:6
./sim.py --natural 3 3 out

# Danielou completo con TUN
./sim.py --danielou-all --export-tun out

# 19-TET
./sim.py -et 19 1200 out

# Serie armonica (primi 16)
./sim.py --geometric 1 16 --no-reduce out
```

Tabella Rapporti Comuni

Sistema	Rapporto	Cents	Intervallo
Ottava	2/1	1200	P8
Quinta giusta	3/2	701.96	P5
Quarta giusta	4/3	498.04	P4

Sistema	Rapporto	Cents	Intervallo
Terza maggiore	5/4	386.31	M3
Terza minore	6/5	315.64	m3
Tono maggiore	9/8	203.91	M2
Tono minore	10/9	182.40	m2
Semitono diatonico	16/15	111.73	m2
Comma sintonico	81/80	21.51	-

Fine documentazione SIM - Sistemi intonazione musicale v1.0