# Labo 6: Wavetable-synthese en het Karplus-Strong algoritme

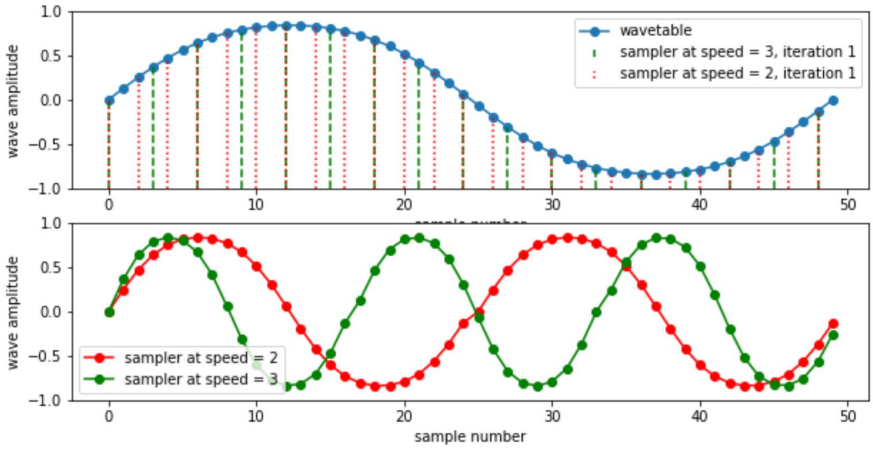
Gianni Allebosch, Martin Dimitrievski

## 1 Wavetable-synthese

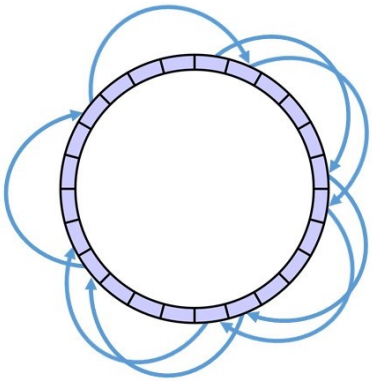
Wavetable-synthese is een geluidsynthesetechniek die wordt gebruikt om periodieke golfvormen te maken. Deze techniek werd ontwikkeld in de late jaren 1970 en wordt vaak gebruikt om muzikale tonen te produceren in synthesizers.

Wavetable-synthese is gebaseerd op het volgende idee: een wavetable is een willekeurige golfvorm met één cyclus voorgesteld als een rij van punten. Een periodieke reproductie kan worden geproduceerd door een sampler met een bepaalde snelheid over de wavetable te itereren en telkens het dichtsbijzijnde punt te nemen. Zie Figuur 1 voor een voorbeeld waarbij twee nieuwe periodieke golfvormen werden geproduceerd door iteratief samples te nemen uit de wavetable met snelheid=2 en snelheid=3. Op deze manier kunnen makkelijk verschillende nieuwe golfvormen van verschillende frequenties worden afgeleid uit de originele wavetable door de wavetable te herhalen en periodiek te samplen.

De wavetable kan ook opgevat worden als een circulaire buffer die met een bepaalde spronggrootte of snelheid wordt doorlopen zoals in Figuur 2.




**Figuur 1.** Nieuwe periodieke golfvormen geproduceerd door de wavetable te samplen met verschillende snelheden.



**Figuur 2.** De wavetable als een circulaire buffer die wordt doorlopen met stapgrootte 5.

## 2 Karplus-Strong algoritme

### 2.1 Basis algoritme

De wavetable-synthesetechniek is heel eenvoudig maar nogal muzikaal saai omdat het puur periodieke tonen produceert. Traditionele muziekinstrumenten produceren geluiden die met de tijd variëren. Deze variatie kan via het Karplus-Strong[1] algoritme op computers worden bereikt. Het algoritme is gebaseerd op wavetable-synthese maar produceert zijn rijke, natuurlijke golfvormen door de wavetable te wijzigen tijdens het samplen.

Figuur 3 illustreert het Karplus-Strong algoritme voor het produceren van golfvormen die klinken als het aanslaan van een gitaarsnaar. Mathematisch is dit te schrijven als

$$Y_t = \frac{1}{2}(Y_{t-p} + Y_{t-p-1}) \tag{1}$$

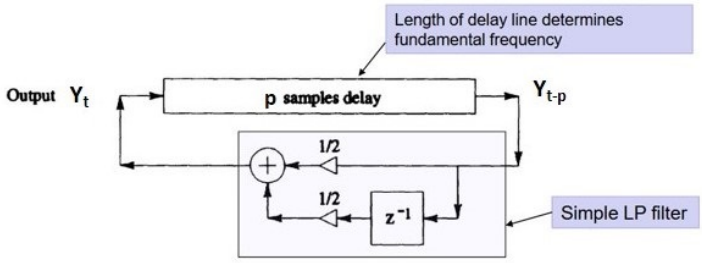
waarbij $Y_t$ de t-de sample van de golfvorm is en p de vertraging. De frequentie van de bekomen golfvorm komt overeen met een periode van $p + \frac{1}{2}$ samples:

$$f = \frac{f_s}{p + \frac{1}{2}} \tag{2}$$

De vertragingsbuffer met p+1 samples is overeenkomstig met de wavetable uit wavetable-synthese. Om realistische snaargeluiden te produceren is het wenselijk om de noot te beginnen met veel hoge harmonischen. Dit kan bereikt worden door de vertagingsbuffer initieel te vullen met een random 1,-1 patroon:

$$Y_t = \begin{cases} 1 & \text{probabiliteit } \frac{1}{2} \\ -1 & \text{probabiliteit } \frac{1}{2} \end{cases} \text{ voor } -p \leq t \leq 0 \tag{3}$$

[1] Zie *karplus_strong.pdf* voor meer informatie over het Karplus-Strong algoritme.




**Figuur 3.** Schematisch overzicht van het Karplus-Strong algoritme voor aangeslagen gitaarsnaren.

### 2.2 Uitrekken van het verval

Er zijn verschillende manieren om de noten langer te laten klinken. Een eerste methode is om een langere vertraginsbuffer te gebruiken en deze te vullen met meerdere kopieën van dezelfde golfvorm.

Een andere, sterkere methode om het verval van de noot uit te rekken is door het Karplus-Strong algoritme aan te vullen met een uitrekkingsfactor $S$:

$$Y_t = \begin{cases} Y_{t-p} & \text{probabiliteit} 1 - \frac{1}{S} \\ \frac{1}{2}(Y_{t-p} + Y_{t-p-1}) & \text{probabiliteit} \frac{1}{S} \end{cases} \quad \text{met } S \geq 1 \tag{4}$$

De vervaltijd van elke toon wordt ongeveer vermenigvuldigd met $S$ in vergelijking met de vervaltijd voor dezelfde toon in het basisalgoritme. De toonhoogte van het geluid wordt echter ook beïnvloed door $S$, aangezien de periode nu ongeveer $p + \frac{1}{2S}$ is in plaats van $p + \frac{1}{2}$. De optimale keuze voor $S$ hangt dus af van de samplefrequentie $f_s$, $p$ en het gewenste effect. De vervaltijd $D$ van de golfvorm is proportioneel met $p^3$:

$$D \approx \frac{(2p+1)^3}{4\pi^2 n^2 f_s} \approx \frac{2p^3}{\pi^2 n^2 f_s} \sim p^3/n^2 \tag{5}$$

met $n$ de $n^{de}$ harmonische van de basisfrequentie van de golfvorm. Door $S$ evenredig te kiezen met $\frac{1}{p}$ of $\frac{1}{p^2}$, wordt de vervaltijd $D$ evenredig met $p^2$ of $p$. Een vaak gebruikte formule voor de uitrekkingsfactor die voor een goede tuning zorgt is dan ook:

$$S \approx \frac{C}{p} \approx Cf \tag{6}$$

met C een constante factor.

### 2.3 Opdrachten

Met behulp van het Karplus-Strong algoritme kunnen we geluiden synthetiseren die sterk lijken op het tokkelen van een gitaar. In volgende opdrachten zullen we stap voor stap melodieën creëren op een gesynthetiseerde gitaar.

1. Schrijf een functie die het Karplus-Strong algoritme toepast om een golfvorm te produceren met frequentie 110 Hz en 3 s loopduur bij $f_s = 8000\,Hz$. De samplefrequentie duidt het aantal samples per seconde aan. Speel deze noot af en plot ze uit op een tijdas. Toon ook de inhoud van de initiële vertragingsbuffer. **Vraag:** ***Wat is het gevolg van het feit dat*** $p$ ***een geheel getal moet zijn?)***
   **TIP:** Sla de golfvorm op als een .wav-file m.b.v. de ***write*** functie van ***scipy.io.wavfile*** en speel ze af m.b.v. onderstaande hulpcode.

```
import simpleaudio as sa
def Audio(sample,rate=fs):
    # Ensure that highest value is in 16-bit range
    audio = sample / np.max(np.abs(sample))
    audio = audio * (2 ** 15 - 1)
    # Convert to 16-bit data
    audio = audio.astype(np.int16)
    # Start playback
    play_obj = sa.play_buffer(audio, 1, 2, rate)
    # Wait for playback to finish before exiting
    play_obj.wait_done()
```

2. Schrijf een functie die de frequentie van een gegeven noot in een gegeven octaaf berekent. De functie moet minimaal een bereik hebben van noot C in octaaf 1 t.e.m. noot B in octaaf 6. De toonladder gaat als volgt: C, C#, D, D#, E, F, F#, G, G#,A, A#, B. Gebruik slechts één hardgecodeerde basisfrequentie voor de laagste noot en bereken daaruit de frequenties van de andere noten in het bereik. De frequentie van C1 is 32.703 Hz. Tussen elke noot wordt er een sprong gemaakt met factor $2^{\frac{1}{12}}$. **Vraag:** ***Wat is de frequentie voor volgende noten: B4, F5, G#6 en D7?***
3. Synthetiseer de noot A vanaf octaaf 1 t.e.m. octaaf 6 via het Karplus-Strong algoritme, speel ze elk 2 s lang en toon ze afzonderlijk op een tijdas. Toon ook voor elke noot het frequentiespectrum. **Vraag:** ***Welk effect heeft de frequentie op het verloop van de geluidsgolf? Wat zie je nog op het frequentiespectrum?***
4. Schrijf een functie die de tweede methode toepast om het verval van de noot uit te rekken met een gegeven uitrekkingsfactor $S$. Pas dit toe op de noot A vanaf octaaf 1 t.e.m. octaaf 6 met looptijd 2 s waarbij de uitrekkingsfactor $S = 2^{octaaf-1}$. Speel elke noot 2 s lang af. Beeld opnieuw elke noot afzonderlijk af op de tijdsas. **Vraag:** ***Met welke verhouding komt deze uitrekkingsfactor overeen?***
   **LET OP:** De uitrekkingsfactor heeft invloed op de frequentie en dus ook op de $p$ van de geluidsgolf.
5. Maak twee golfvormen die respectievelijk de C-majeur en de natuurlijke C-mineur toonladder voorstellen in octaaf 3. Gebruik hierbij een $S$-factor die evenredig is met de frequentie van de noot en omgekeerd evenredig met de frequentie van de laagste noot in de toonladder ($S \sim \frac{f}{f_{min}}$). Elke noot moet hierbij 0.33 seconde lang zijn. Sla deze golfvormen op en speel ze af.

6. De *RingTone Text Transfer Language* (RTTTL) was vroeger het primaire formaat waarin beltonen werd gedistributeerd voor Nokia-gsm's. We kunnen dit ook gebruiken om eenvoudige melodieën te synthetizeren. Lees volgende RTTTL-voorbeelden[2] in en synthetiseer hun melodieën in het bereik [C2,B5], sla ze op en speel ze af. Gebruik $S = \frac{f}{f_{C2}}$ voor elke noot. Herken je deze melodieën?

**melodie 1**: •

- basis octaaf = 2, basis loopduur = 4, bpm = 125
- rtttl = 4e3, 8p, 8e3, 8.g3, 8e3, 32p, 8d3, 32p, 2c3, 4b, 4p, 4e3, 8p, 8e3, 8g3, 32p, 8e3, 32p, 8.d3, 8c3, 32p, 8d3, 32p, 8.c3, 4b, 4p, 4e3, 8p, 8e3, 8.g3, 8e3, 32p, 8d3, 32p, 2c3, 4b, 4p, 4e3, 8p, 8e3, 8g3, 32p, 8e3, 32p, 8.d3, 8c3, 32p, 8d3, 32p, 8.c3, 4b

**melodie 2**: •

- basis octaaf = 2, basis loopduur = 16, bpm = 125
- rtttl = 2p, 8p, 8b, 8e.3, g3, 8f#3, 4e3, 8b3, 4a.3, 4f#.3, 8e.3, g3, 8f#3, 4d3, 8f3, 2b, 8p, 8b, 8e.3, g3, 8f#3, 4e3, 8b3, 4d4, 8c#4, 4c4, 8g#3, 8c.4, b3, 8a#3, 4f#3, 8g3, 2e3, 8p, 8g3, 4b3, 8g3, 4b3, 8g3, 4c4, 8b3, 4a#3, 8f#3, 8g.3, b3, 8a#3, 4a#, 8b, 2b3, 8p

**Enkele tips:**

- Het RTTTL formaat: [<loopduur>] <toon> [<speciale verlengduur>] [<octaaf>] <scheidingsteken>.
- De loopduur van de noot is uitgedrukt als de deler van de volledige nootduur, e.g. 4 stelt een kwartnoot voor.
- De *'p'* noot in het RTTTL formaat stelt een pauze voor. Dit is een noot met frequentie = 0 Hz.
- Als de noot in het RTTTL formaat wordt gevolgd door een '.', dan wordt de noot verlengd met factor 1.5.
- De basis loopduur is de default loopduur van de noot indien deze niet is meegegeven in de RTTTL-noot.
- Een noot is normaal 4 beats lang. Gebruik de bpm, de basisduur, de loopduur en speciale verlengduur om de totale duur van de noot in seconden te berekenen.
- Voeg een fade-in effect toe aan elke noot met $T_{fade-in} = \frac{1}{8}$ beat. Je kan een fade-in effect creëen door een envelop toe te voegen aan de gesynthetiseerde golfvorm. Voeg een exponetiële fade-in toe door de golfvorm te vermenigvuldigen met $(1 - exp(\frac{-5t}{T_{fade-in}})$ waarbij $t$ de tijdsas van de gesynthetiseerde golfvorm voorstelt en $T_{fade-in}$ de looptijd van het fade-in effect.
- Je kan onderstaande code gebruiken om de loopduur, de toon, de octaaf en speciale verlengduur van een noot in het RTTTL formaat in te lezen.

[2] Meer voorbeelden kun je vinden op: `https://ringtonesgalore.co.uk/`

```
import re

note = "16f#6"
duration_pattern = re.compile("^[0-9]{1,2}")
pitch_pattern = re.compile("[a-zA-Z]#*")
octave_pattern = re.compile("[0-9]$")
special_pattern = re.compile("[.]")

duration = duration_pattern.findall(note)
pitch = pitch_pattern.findall(note)[0].lower()
octave = octave_pattern.findall(note)
special_duration = special_pattern.findall(note)
```

## 3 Indienen

Indienen van het labo gebeurt via Opdrachten op Ufora. De code dien je in als 1 pythonscript genaamd **labo6.py** waarin alle opdrachten sequentieel worden uitgevoerd. Voeg een **README.txt** bestand toe waarin je programma beschreven is en waarin de vragen beantwoord worden. Alle gevraagde outputafbeeldingen en geluidsbestanden geef je mee in een .zip-file genaamd **output.zip**.