SINGING VOICE SYNTHESIS USING DIFFERENTIABLE LPC AND GLOTTAL-FLOW-INSPIRED WAVETABLES

Chin-Yun Yu   **György Fazekas**

Centre for Digital Music, Queen Mary University of London, UK

chin-yun.yu@qmul.ac.uk, george.fazekas@qmul.ac.uk

ABSTRACT

This paper introduces GlOttal-flow LPC Filter (GOLF), a novel method for singing voice synthesis (SVS) that exploits the physical characteristics of the human voice using differentiable digital signal processing. GOLF employs a glottal model as the harmonic source and IIR filters to simulate the vocal tract, resulting in an interpretable and efficient approach. We show it is competitive with state-of-the-art singing voice vocoders, requiring fewer synthesis parameters and less memory to train, and runs an order of magnitude faster for inference. Additionally, we demonstrate that GOLF can model the phase components of the human voice, which has immense potential for rendering and analysing singing voice in a differentiable manner. Our results highlight the effectiveness of incorporating the physical properties of the human voice mechanism into SVS and underscore the advantages of signal-processing-based approaches, which offer greater interpretability and efficiency in synthesis.

1. INTRODUCTION

Singing voice synthesis (SVS) has attracted substantial interest as a research topic over the last decades, and a variety of techniques have been developed. Early successful SVS systems were usually based on sample concatenation [1–4], while parametric systems have become much more prevalent. The actual synthesis process in parametric systems is carried out by a *vocoder* controlled by synthesis parameters generated from a separate acoustic model given some musical context factors (i.e. note number, duration, phoneme, etc.). Early systems of this kind use a linear source-filter model as vocoder [5, 6]. Deep Neural Networks (DNNs) have subsequently become the dominant approach for state-of-the-art vocoders [7–13]. However, mel-spectrograms are often chosen as input features to these models, which are less interpretable than traditional vocoder parameters (e.g. f0, aperiodicity ratios). Also, a significant amount of data is needed to cover various vocal expressions to achieve generalisation.

In contrast, Differentiable Digital Signal Processing (DDSP) models [14–16] incorporate existing signal processing operations into neural networks as an inductive bias, making them more interpretable and generalisable. DDSP additive synthesis has been proposed for SVS by Alonso et al. [17]. Wu et al. [18] improved this further by using subtractive synthesis and sawtooth as the harmonic source. Nercessian et al. [19] proposed a differentiable version of the WORLD vocoder [20] for doing end-to-end singing voice conversion. Yoshimura et al. [21] used Taylor expansion to approximate the mel-log spectrum approximation filter's (MLSA) exponential function and embedded it into an SVS system. However, most of their architectures only assume the target signal is a monophonic instrument, which can potentially lead to solutions that do not reflect some properties of voice. In their design, the harmonic sources are fixed to a specific shape (e.g. sawtooth, pulse train), and the filters are symmetric in the time domain, except Yoshimura et al. [21] which use a minimum-phase MLSA filter. Incorporating constraints specific to the human voice on the harmonic source and the filters could lead to a more interpretable and compact SVS vocoder.

In this work, we propose GlOttal-flow LPC Filter (GOLF), an SVS module informed by the physical properties of the human voice. We build upon the Harmonic-plus-Noise architecture of DDSP [14] and the subtractive synthesis of SawSing [18], but replace the harmonic source with a glottal model and use IIR filters. We developed a differentiable IIR implementation in PyTorch [22] for training efficiency. We then used this module as a neural vocoder and compared its performance with other DDSP-based vocoders. Specifically, a simple and lightweight NN encoder converts the mel-spectrogram into synthesis parameters, and the synthesiser decodes the signal from it. We paired different synthesisers with the same encoder and trained them jointly.

Our contributions are twofold. First, GOLF has significantly fewer synthesis parameters but is still competitive with state-of-the-art SVS vocoders. Second, GOLF requires less than 40% of memory to train and runs ten times faster than its alternatives for inference. Moreover, we indirectly show that GOLF could model the phase components of the human voice by aligning the synthesised waveforms to the ground truth and calculating the differences. This characteristic has excellent potential for analysing singing voice in a differentiable manner. Decomposing the

human voice into the glottal source and vocal tract could also enable us to adjust the singing style in different ways, such as altering the amount of vocal effort with varying shapes of the glottal pulse.

2. BACKGROUND

We first introduce the relevant notation. $\mathbf{x}_i$ denotes the ith column vector and $x_{i,j}$ denotes the entry at the ith row and the jth column of the matrix $\mathbf{X}$. Concatenating two matrices along the column dimension is denoted by $[;]$. x_i denotes the ith entry of the vector $\mathbf{x}$ or a time sequence x indexed by i. $X(z)$ denotes the response of x_n in the z-domain. Unless stated otherwise, we use n as the time index and k as the frame index. Angular frequencies and periods are normalised to the interval $[0, 1]$. We use one-based indexing for elements with finite dimensions.

2.1 Glottal Source-Filter Model

In the source-filter model, we have the following simplified voice production model:

$$S(z) = (G(z) + N(z)) H(z)L(z), \qquad (1)$$

where $G(z)$ represents the periodic vibration from the vocal folds, $N(z)$ represents random components of the glottal source, $H(z)$ represents the vocal-tract filter, and $L(z)$ represents the radiation at the lips [23]. Since this formulation is linear, the radiation filter $L(z)$ and the glottal pulse $G(z)$ can be merged into a single source $G'(z)$ called *the radiated glottal pulse*. If we assume $L(z)$ is a first-order differentiator $1 - z^{-1}$ [24], then $G'(z)$ is the derivative of the glottal pulse, which can be described by the LF model [25], a four-parameter model of glottal flow. $H(z)$ is usually a Linear Predictive Coding (LPC) filter.

2.2 Linear Predictive Coding

LPC assumes that the current speech sample s_n can be predicted from a finite number of previous M samples s_{n-1} to s_{n-M} by a linear combination with residual errors e_n:

$$s_n = e_n - \sum_{i=1}^{M} a_i s_{n-i}, \qquad (2)$$

where a_i are the linear prediction coefficients. This is the same as filtering the residuals, equivalent to the glottal source in our case, with an Mth-order all-pole filter, a filter that has an infinite impulse response (IIR). We can use the LPC filter to represent the response of the vocal tract if the vocal tract is approximated by a series of cylindrical tubes with varying diameters [26], providing a physical interpretation.

Using LPC for neural audio synthesis is not new [8, 27, 28], and works have been conducted to incorporate IIR filters and train them jointly with deep learning models [29–35]. The difficulty of training IIR in deep learning framework (e.g. PyTorch) using Eqn (2) is that its computation is recursive, i.e. the output at each step depends on

the previous results, and to make the calculation differentiable, separated tensors are allocated in each step. This generates a significant number of memory allocations and overheads for creating tensors, thus leading to performance issues, especially for long sequences. One way to mitigate this is to allocate shared continuous memory before computation. However, in-place modification is not differentiable in these frameworks. Some studies sidestep the recursion by approximating IIR in the frequency domain using Discrete Fourier Transform (DFT) [27, 30, 32–35], but the accuracy of this approximation depends on the DFT resolution. Moreover, the IIRs used in practice are usually low-order; in this case, it is faster to compute them directly, especially on long sequences.

3. PROPOSED MODEL

Usually, $N(z)$ in Eqn (1) is treated as amplitude-modulated Gaussian noise [23, 24]. Our early experiments found this formulation to be challenging to optimise. As an alternative, we move the noise components $N(z)$ outside the glottal source and filter it with time-varying filter $C(z)$, resulting in

$$S(z) = G'(z)H(z) + N(z)C(z). \qquad (3)$$

This resembles the classic *Harmonic-plus-Noise* model [36] and was used in previous DDSP-based SVS [17, 18]. Alonso et al. [17] modelled $G'(z)H(z)$ jointly using additive harmonic oscillators and time-varying Finite Impulse Responses (FIRs) as $C(z)$; Wu et al. [18] introduced a sawtooth oscillator as $G'(z)$ and zero-phase time-varying FIRs as $H(z)$. In this work, we use a glottal flow model to synthesise harmonic sources and time-varying IIRs as filters.

3.1 Glottal Flow Wavetables

We adopted the transformed-LF model [37] for generating glottal pulses. This model re-parameterises the LF model [25] using just one parameter R_d, which has been found to correspond to the perceived vocal effort well and covers a wide range of different glottal flow shapes. We sampled K values of $\log(R_d)$ with equal spacing inside $[\log(0.3), \log(2.7)]$ according to the value range suggested by [23]. We calculate the flow derivative function $g'(t; R_d)$ in continuous time t for each sampled R_d and then sampled L points in one period to get its discrete version. The details for calculating $g'(t; R_d)$ were given in [38]. By stacking these sampled glottal flows, we built wavetables $\mathbf{D} \in \mathbb{R}^{K \times L}$, with each row containing one period of a sampled glottal pulse (see Fig. 1). The rows are sorted based on R_d.

The model generates glottal pulses g'_n by linearly interpolating the two $\mathbf{D}$ axes. The encoder network first predicts instantaneous frequency $f_n \in [0, 0.5]$ and the fractional index $\tau_n \in [0, 1]$ for R_d. We then use the instantaneous phase $\phi_n = \sum_{i=1}^{n} f_i$ to interpolate the waveform
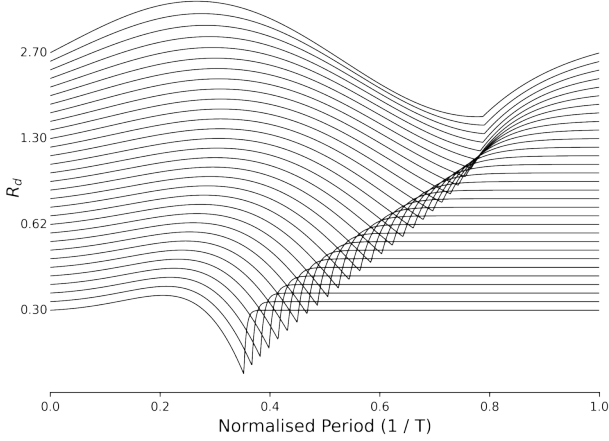
Figure 1. An example of the wavetables we used, corresponding to matrix $\mathbf{D}$ with $K = 31$.

as:

$$
\begin{aligned}
g'_n = (1 - p) &\left((1 - q)\hat{d}_{\lfloor k \rfloor, \lfloor l \rfloor} + q\hat{d}_{\lfloor k \rfloor, \lceil l \rceil} \right) \\
&+ p \left((1 - q)\hat{d}_{\lceil k \rceil, \lfloor l \rfloor} + q\hat{d}_{\lceil k \rceil, \lceil l \rceil} \right), \quad (4)
\end{aligned}
$$

where $l = (\phi_n \mod 1)L + 1$, $k = \tau_n(K - 1) + 1$, $p = k - \lfloor k \rfloor$, $q = l - \lfloor l \rfloor$, and $\hat{\mathbf{D}} = [\mathbf{D}; \mathbf{d}_1] \in \mathbb{R}^{K \times (L+1)}$. The wavetables $\mathbf{D}$ are fixed in our case, contrary to [16], and we only pick one wavetable at a time, not a weighted sum.

3.2 Frame-Wise LPC Synthesis

Time-varying LPC synthesis is usually done by linearly interpolating the LPC coefficients to the audio resolution and filtering sample by sample. This is not parallelisable and slows down the training process. As an alternative, we approximate LPC synthesis by treating each frame independently and using overlap-add:

$$
s_n = \sum_k LPC(g'_n \gamma_n u_{n-kT}; \mathbf{a}_k) w_{n-kT}, \quad (5)
$$

where $LPC(e_n; \mathbf{a})$ represents Eqn (2), $\mathbf{a}_k \in \mathbb{R}^M$ are the filter coefficients at the kth frame, u_n and w_n are the windowing functions, $\gamma_n \in \mathbb{R}^+$ is the gain, and T is the hop size. u_n is fixed to the square window. In this way, the computation can be parallelised. We found that the voice quality differences between overlap-add LPC and sample-by-sample LPC are barely noticeable if we use a sufficiently small hop size. We empirically found that a 200 Hz frame rate is sufficient.

3.3 LPC Coefficients Parameterisation

For the LPC filter to be stable, all of its poles must lie inside the unit circle on the complex plane. Stability can be guaranteed using robust representations, such as reflection coefficients [28]. The representation we chose in this work is cascaded 2nd-order IIR filters, and we solve the stability issue by ensuring all the 2nd-order filters are stable. We use the *coefficient representation* from [33] to parameterise

the ith IIR filter's coefficients $1 + \eta_{i,1}z^{-1} + \eta_{i,2}z^{-2}$ from the encoder's outputs and cascade them together to form an Mth-order LPC filter:

$$
\begin{aligned}
&(1 + \eta_{1,1}z^{-1} + \eta_{1,2}z^{-2})(1 + \eta_{2,1}z^{-1} + \eta_{2,2}z^{-2}) \\
&\cdots (1 + \eta_{\frac{M}{2},1}z^{-1} + \eta_{\frac{M}{2},2}z^{-2}) \quad (6) \\
&= 1 + a_1 z^{-1} + a_2 z^{-2} + \cdots + a_M z^{-M} = A(z).
\end{aligned}
$$

3.4 Unvoiced Gating

The instantaneous frequency f_n predicted by the encoder is always non-zero and keeps the oscillator working. Without constraint, the model would utilise these harmonics in the unvoiced region creating buzzing artefacts [18]. We propose to mitigate this problem by jointly training the model to predict the voiced/unvoiced probabilities as $v_n \in [0, 1]$ and feeding the gated frequency $\hat{f}_n = v_n f_n$ to the oscillator instead.

4. OPTIMISATION

Training deep learning models is usually accomplished by backpropagating the gradients evaluated at a chosen loss function $\mathcal{L}$ throughout the whole computational graph back to the parameters. Partially inspired by Bhattacharya et al. [29], we derived the closed form of *backpropagation through time* to utilise efficient IIR implementation to solve the problems we mentioned in Section 2.2 while keeping the filter differentiable. Here, $\mathbf{e} \in \mathbb{R}^N$ is the input, $\mathbf{a} \in \mathbb{R}^M$ is the filter coefficients, and $\mathbf{s} \in \mathbb{R}^N$ is the output. Assuming we know $\frac{\partial \mathcal{L}}{\partial \mathbf{s}}$, we can get the derivatives $\frac{\partial \mathcal{L}}{\partial \mathbf{e}}$ and $\frac{\partial \mathcal{L}}{\partial \mathbf{a}}$, using chain rules $\frac{\partial \mathcal{L}}{\partial \mathbf{s}} \frac{\partial \mathbf{s}}{\partial \mathbf{e}}$ and $\frac{\partial \mathcal{L}}{\partial \mathbf{s}} \frac{\partial \mathbf{s}}{\partial \mathbf{a}}$.

4.1 Backpropagation Through the Coefficients

Taking the derivatives of Eqn (2) with respect to a_i we get:

$$
\frac{\partial s_n}{\partial a_i} = -s_{n-i} - \sum_{k=1}^M a_k \frac{\partial s_{n-k}}{\partial a_i}, \quad (7)
$$

which equals $LPC(-s_{n-i}; \mathbf{a})$. $s_n|_{n \leq 0}$ does not depend on a_i so the initial conditions $\frac{\partial s_n}{\partial a_i}|_{n \leq 0}$ are zeros. We can get $\frac{\partial s_n}{\partial \mathbf{a}}$ with one pass of filtering because $\frac{\partial s_n}{\partial a_j}$ is $\frac{\partial s_n}{\partial a_i}$ shifted by an offset $j - i$. Lastly, we calculate $\frac{\partial \mathcal{L}}{\partial a_i}$ as $\sum_{n=1}^N \frac{\partial \mathcal{L}}{\partial s_n} \frac{\partial s_n}{\partial a_i}$.

4.2 Backpropagation Through the Input

To get the derivatives for input e_n, we first re-write Eqn (2) as the following convolutional form:

$$
s_n = \sum_{m=1}^n e_m h_{n-m}, \quad (8)
$$

where $h_n = \mathcal{Z}^{-1}\{H(z)\}$, $H(z) = \frac{1}{A(z)}$. From Eqn (8) we see that $\frac{\partial s_n}{\partial e_m} = h_{n-m}$. The derivative of loss $\mathcal{L}$ with respect to e_m depends on all future samples s_n, which is:

$$
\frac{\partial \mathcal{L}}{\partial e_m} = \sum_{n=m}^N \frac{\partial \mathcal{L}}{\partial s_n} \frac{\partial s_n}{\partial e_m} = \sum_{n=m}^N \frac{\partial \mathcal{L}}{\partial s_n} h_{n-m}. \quad (9)
$$

By swapping the variables n, m and considering the equivalence of Eqn (2) and Eqn (8), Eqn (9) can be simplified to

$$\frac{\partial \mathcal{L}}{\partial e_n} = \sum_{m=n}^{N} \frac{\partial \mathcal{L}}{\partial s_m} h_{m-n}$$
$$= \frac{\partial \mathcal{L}}{\partial s_n} - \sum_{i=1}^{M} a_i \frac{\partial \mathcal{L}}{\partial e_{n+i}}. \quad (10)$$

Eqn (10) shows that we can get the derivatives $\frac{\partial \mathcal{L}}{\partial e_n}$ by just filtering $\frac{\partial \mathcal{L}}{\partial s_n}$ with the same filter, but running in backwards. The initial conditions $\frac{\partial \mathcal{L}}{\partial e_n}|_{n>N}$ are naturally zeros.

In conclusion, backpropagation through an IIR filter consists of two passes of the same filter and one matrix multiplication [1]. We implemented the IIR in C++ and CUDA with multi-threading to filter multiple sequences simultaneously [2]. The differentiable IIR is done by registering the above backward computation in PyTorch, and we submitted the implementation to TorchAudio [39] as part of the `torchaudio.functional.lfilter`.

5. EXPERIMENTAL SETUP

5.1 Dataset

We test GOLF as a neural vocoder on the MPop600 dataset [40], a high-quality Mandarin singing voice dataset featuring nearly 600 singing recordings with aligned lyrics sung by four singers. We used the audio recordings from the `f1` (female) and `m1` (male) singers. For each singer, we selected the first three recordings as the test set, the following 27 recordings as the validation set, and used the rest as training data (around three hours in total). All the recordings were downsampled to 24 kHz. The vocoder feature we choose is the log mel-spectrogram. We computed the feature with a window size of 1024 and 80 mel-frequency bins and set the hop size T to 120. We normalised the feature to between zero and one and sliced the training data into two seconds excerpts with 1.5 seconds overlap.

5.2 Model Details

We adopted the encoder from SawSing but replaced the transformer layers with three layers of Bi-LSTM for favourable implementation, resulting in around 0.7M parameters in total. A final linear layer predicts the synthesis parameters $\{f_k, v_k, \gamma_k, \beta_k, \mathbf{a}_k, \mathbf{b}_k\}$. The first four parameters are linearly upsampled to $\{f_n, v_n, \gamma_n, \beta_n\}$. β_n and $\mathbf{b}_k$ are the Gaussian noise's gain and filter coefficients. We added an average pooling layer with a size of 10 and two convolution layers after the encoder to predict the R_d fractional index τ_o at a lower rate and then linearly upsampled to τ_n. This step avoids possible modulation effects caused by switching the wavetables too quickly. A

---

[1] The computation of the IIR does not need to fulfil the implementation requirements set by the automatic differentiation framework, thus can be highly optimised.

[2] Although the single-core performance of a GPU is usually inferior to a CPU, and we can only use at most one thread for each IIR, the GPU has a much higher number of cores, which is beneficial for training on a large number of sequences at once.

system diagram of GOLF is shown in Fig. 2. We set $K = 100, L = 2048$, Hanning window for w_n, and $M = 22$ for both LPC filters. We used the same hop size T and a window size of 480 for frame-wise LPC. We normalised all wavetables to have equal energy and aligned them with the negative peak.
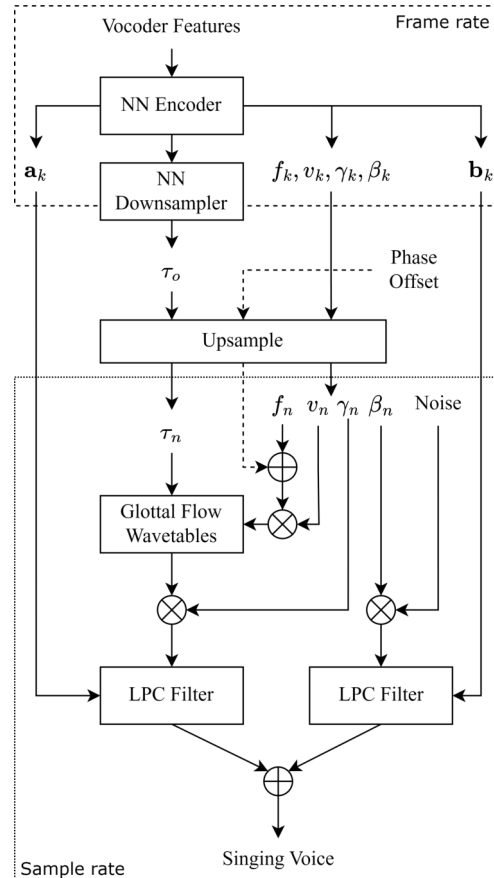

Figure 2. Overview of the GOLF synthesis process. *phase offset* is only introduced at test time, where the details are given in Section 6.1.

We compare GOLF with three DDSP-based baselines using the same NN encoder to predict their synthesis parameters. The first two are the original DDSP [14] and SawSing [18]. We set their noise filter length to 80 and harmonic filter length to 256 for SawSing. The third model is PUlse-train LPC Filter (PULF), which is similar to GOLF but replaces the glottal flow wavetables with a band-limited pulse train [19] using additive synthesis, while the LPC order for the harmonic source is increased to 26 to accommodate the glottal pulse response. The number of oscillating sinusoids was set to over 150 for all the baselines. We did not compare GOLF with Nercessian et al. [19] and Yoshimura et al. [21] because these architectures are based closely on the source-filter model, and use additional postnets to enhance the voice, which makes it harder to compare directly with GOLF.

5.3 Training Configurations

We trained separate models for each singer, resulting in 8 models. The loss function is the summation of the

multi-resolution STFT loss (MSSTFT) and f0 loss from SawSing with FFT sizes set to $\{512, 1024, 2048\}$, plus a binary cross entropy loss on voiced/unvoiced prediction. We stopped the gradients from the harmonic source to the f0s and voiced decisions to stabilise the training. We used Adam [41] for running all optimisations. For DDSP and SawSing, the batch size and learning rate were set to 32 and 0.0005; for GOLF and PULF, the numbers were 64 and 0.0001. We used the ground truth f0s (extracted by WORLD [20]) for the harmonic oscillator of PULF during training due to stability issues. We trained all the models for 800k steps to reach sufficient convergence and picked the checkpoint with the lowest validation loss as the final model [3].

6. EVALUATIONS

6.1 Objective Evaluation

The objective metrics we choose are the MSSTFT, the mean absolute error (MAE) in f0, and the Fréchet audio distance (FAD) [42] on the predicted singing of the test set. Table 1 shows that DDSP has the lowest MSSTFT and f0 errors, while SawSing reaches the lowest FAD. GOLF and PULF show comparable results in f0 errors to other baselines. We report the memory usage when training these models and their real-time factor (RTF), both on GPU and CPU, in Table 2. The amount of memory required to train GOLF is around 35% of others, and it runs extremely fast, especially on the CPU.

Singers	Models	MSSTFT	MAE-f0 (cent)	FAD
f1	DDSP	**3.09**	**74.47**±1.19	0.50±0.02
	SawSing	3.12	78.91±1.18	**0.38**±0.02
	GOLF	3.21	77.06±0.88	0.62±0.02
	PULF	3.27	76.90±1.11	0.75±0.04
m1	DDSP	**3.12**	**52.95**±1.03	0.57±0.02
	SawSing	3.13	56.46±1.04	**0.48**±0.02
	GOLF	3.26	54.09±0.30	0.67±0.01
	PULF	3.35	54.60±0.73	1.11±0.04

Table 1. Evaluation results on the test set. We omit the standard deviation if it is smaller than 0.01.

As an additional metric we use the L2 loss between the predicted and the ground truth waveform. The intuition behind this is that GOLF and PULF are the only two models introducing non-linear phase response because of IIR filtering. The filters in DDSP and SawSing are all zero-phase, and the initial phases of the sinusoidal oscillators are fixed to zeros. We emphasise that this test is not targeting human perception but the phase reconstruction ability of the models. Humans cannot perceive the absolute frequency phase, but accurate reconstruction could be important in sound matching and mixing use cases. We evaluate the loss on one of the test samples from m1 we used in the subjective evaluation. We created a new parameter

---

[3] The trained checkpoints, source codes, and audio samples are available at https://github.com/iamycy/golf.

called *phase offset* sampled at 20 Hz. We linearly upsampled *phase offset* and added it to the instantaneous phase ϕ_n, introducing a slowly varying phase shift. We optimised this parameter by minimising the predicted waveform's L2 loss to the ground truth using Adam with a learning rate of 0.001 and 1000 steps. We wrapped the differences between the points of *phase offset* during optimisation to [-0.5, 0.5]. We ran this optimisation five times for each model. Each time the *phase offset* was initialised randomly. We report the minimum and maximum final losses from these trials. Table 2 shows the lowest losses GOLF and PULF can reach are significantly smaller than the others, with GOLF having the smallest among all.

Models	Memory	RTF		Waveform L2	
		GPU	CPU	Min	Max
DDSP	7.3	0.015	0.237	71.83	88.77
SawSing	7.3	0.015	0.240	75.72	93.16
GOLF	**2.6**	**0.009**	**0.023**	**21.98**	**64.82**
PULF	7.5	0.015	0.248	44.08	70.59

Table 2. The required number of VRAM (GB) for training with a batch size of 32, real-time factor (RTF), and the minimum/maximum L2 loss on waveform using one of the test samples. The benchmark was conducted on an Ubuntu 20.04 LTS machine with an i5-4790k processor and an NVIDIA GeForce RTX 3070 GPU.

6.2 Subjective Evaluation

We conducted an online listening test using Go Listen [43]. We picked one short clip from each test set recording, resulting in 6 clips with duration ranging from 6 to 11 seconds. The test is thus divided into six examples, each consisting of one ground truth clip and four synthesised results from different models, and their loudness was normalised to −16dB LUFS. The order of the examples and the stimulus were randomised for each subject. Each subject was requested to rate the quality of these stimuli on a score from 0 to 100. We collected responses from 33 anonymous participants. We dropped one of the participants who did not indicate using headphones. We normalise scores to fall between 1 to 5 and report the Mean Opinion Score (MOS) in Fig. 4. DDSP has the highest opinion scores overall, and a Wilcoxon signed-rank test shows that it is not statistically significantly different from the f1 ground truth (p = 0.168). We applied the one-side Wilcoxon test on GOLF and PULF to compare them with SawSing, and the results show that GOLF significantly outperforms SawSing ($p < 0.0001$), and PULF performs better than SawSing on m1 ($p < 0.022$).

7. DISCUSSIONS

Given the evaluation results and the number of synthesis parameters in GOLF is roughly six times smaller than DDSP and SawSing, it is clear that GOLF's synthesis parameters are a more compact representation. Comparing
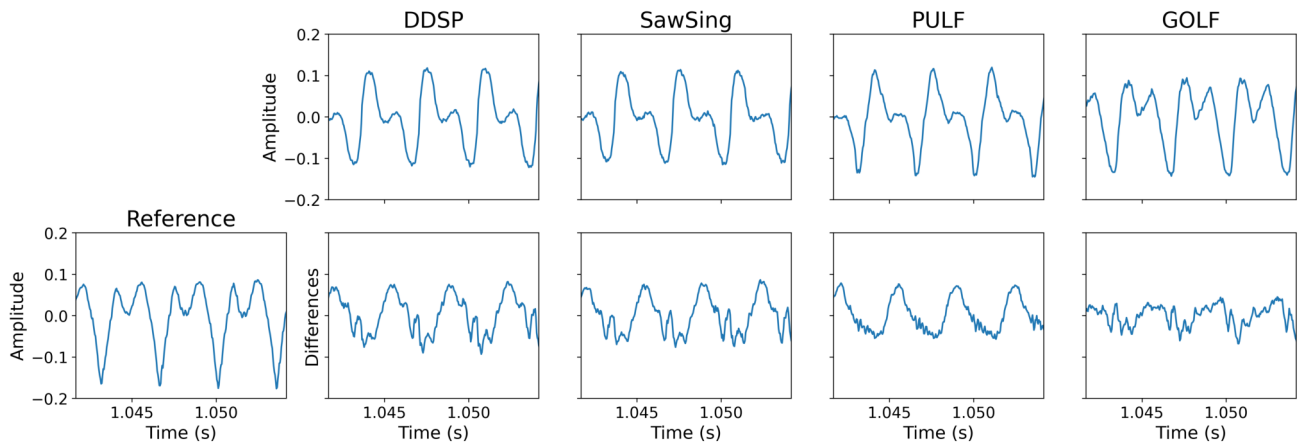
Figure 3. The predicted waveforms of a short segment from one of the `m1` test samples. The differences were computed by subtracting the predicted signal from the reference.
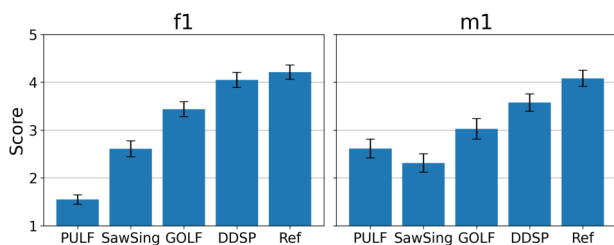

Figure 4. The MOS results of the vocoders trained on different singers with 95% confidence interval.

the differences between GOLF and PULF in Table 2, we can see that the performance gain is due to the use of wavetables. Other baselines synthesise band-limited harmonic sources with many sinusoids oscillating simultaneously, thus increasing the computational cost. PULF's MOS score is much worse on `f1`, with noticeable artefacts in the unvoiced region and random components of the voice. After investigation, we found the noise gains β_n predicted by PULF fluctuating at high speed, producing a modulation effect. This behaviour is also found in PULF trained on `m1` and GOLF, even on the harmonic gains γ_n. Still, the amount of fluctuation is small and barely noticeable in the test samples. Given the available results, we could only conclude that this effect relates to the type of harmonic source and the range of f0, i.e., female singers have higher f0. This amplitude modulation effect cannot be observed in spectrograms and thus is not captured by the training loss we used. It could be an intrinsic drawback of using frame-wise LPC approximation, but more experiments and comparisons with sample-wise LPC are needed. In addition, SawSing produced low scores for both singers because of the buzzing artefacts in the unvoiced region. Although unvoiced gating (Sec. 3.4) reduces this problem to a large degree, human ears are susceptible to this effect. This could be an inherent problem in using a sawtooth as the harmonic source.

The L2 loss shown in Table 2 demonstrates that GOLF matches phase-related characteristics more accurately than

other models. Fig. 3 shows GOLF produces the most similar waveform to the ground truth. Other baselines' waveforms are similar because they use the same additive synthesiser. It is possible to reduce their L2 loss by optimising the initial phases of the oscillators, but this cannot account for time-varying source shapes. Low L2 loss is a positive effect of the deterministic phase responses embedded in GOLF. This opens up many possibilities, such as decomposing and analysing the voice in a differentiable manner and training the vocoder using the time domain loss function. The latter could be a possible way to reduce the fluctuation problem discussed in the previous paragraph. The waveform matching of GOLF can be improved further by using a more flexible glottal source model, adding FIR and all-pass filters to account for the voice's mixed-phase components and the recording environment's acoustic response.

Lastly, we note that cascaded IIR filters provide an orderless representation (i.e. the cascading order does not affect the outputs). This results in the *responsibility problem* [44, 45] for the last layer of the encoder, which might be one of the reasons why GOLF and PULF are less stable to train than other baselines. Developing architectures that can handle orderless representation or switch to other robust representations are possible ways to address this.

8. CONCLUSIONS

We present a lightweight singing voice vocoder called GOLF, which uses wavetables with different glottal flows as entries to model the time-varying harmonic components and differentiable LPC filters for filtering both the harmonics and random elements. We show that GOLF requires less memory to train and runs an order of magnitude faster on the CPU than other DDSP-based vocoders, but still attains competitive voice quality in subjective and objective evaluations. Furthermore, we empirically show that the predicted waveforms from GOLF represent the voice's phase response more faithfully, which could allow us to use GOLF to decompose and analyse human voice.

9. ACKNOWLEDGEMENTS

10. REFERENCES

[1] M. W. Macon, L. Jensen-Link, J. Oliverio, M. A. Clements, and E. B. George, "A singing voice synthesis system based on sinusoidal modeling," in *International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, vol. 1.   IEEE, 1997, pp. 435–438.

[2] J. Bonada, Ò. Celma Herrada, À. Loscos, J. Ortolà, X. Serra, Y. Yoshioka, H. Kayama, Y. Hisaminato, and H. Kenmochi, "Singing voice synthesis combining excitation plus resonance and sinusoidal plus residual models," in *International Computer Music Conference (ICMC)*, Havana, Cuba, 2001.

[3] J. Bonada and X. Serra, "Synthesis of the singing voice by performance sampling and spectral models," *IEEE Signal Processing Magazine*, vol. 24, no. 2, pp. 67–79, 2007.

[4] J. Bonada, M. Umbert Morist, and M. Blaauw, "Expressive singing synthesis based on unit selection for the singing synthesis challenge 2016," in *INTERSPEECH*.   International Speech Communication Association (ISCA), 2016.

[5] K. Saino, H. Zen, Y. Nankaku, A. Lee, and K. Tokuda, "An HMM-based singing voice synthesis system," in *9th International Conference on Spoken Language Processing*, 2006.

[6] Y. Hono, S. Murata, K. Nakamura, K. Hashimoto, K. Oura, Y. Nankaku, and K. Tokuda, "Recent development of the DNN-based singing voice synthesis system—Sinsy," in *Asia-Pacific Signal and Information Processing Association Annual Summit and Conference (APSIPA ASC)*.   IEEE, 2018, pp. 1003–1009.

[7] J. Shen, R. Pang, R. J. Weiss, M. Schuster, N. Jaitly, Z. Yang, Z. Chen, Y. Zhang, Y. Wang, R. Skerrv-Ryan *et al.*, "Natural TTS synthesis by conditioning WaveNet on mel spectrogram predictions," in *International Conference on Acoustics, Speech and Signal Processing (ICASSP)*.   IEEE, 2018, pp. 4779–4783.

[8] J.-M. Valin and J. Skoglund, "LPCNet: Improving neural speech synthesis through linear prediction," in *International Conference on Acoustics, Speech and Signal Processing (ICASSP)*.   IEEE, 2019, pp. 5891–5895.

[9] R. Yoneyama, Y.-C. Wu, and T. Toda, "Unified source-filter GAN with harmonic-plus-noise source excitation generation," in *INTERSPEECH*.   International Speech Communication Association (ISCA), 2022.

[10] R. Prenger, R. Valle, and B. Catanzaro, "WaveGlow: A flow-based generative network for speech synthesis," in *International Conference on Acoustics, Speech and Signal Processing (ICASSP)*.   IEEE, 2019, pp. 3617–3621.

[11] J. Liu, C. Li, Y. Ren, F. Chen, and Z. Zhao, "Diff-Singer: Singing voice synthesis via shallow diffusion mechanism," in *AAAI Conference on Artificial Intelligence*, vol. 36, no. 10, 2022, pp. 11 020–11 028.

[12] Y.-P. Cho, F.-R. Yang, Y.-C. Chang, C.-T. Cheng, X.-H. Wang, and Y.-W. Liu, "A survey on recent deep learning-driven singing voice synthesis systems," in *2021 IEEE International Conference on Artificial Intelligence and Virtual Reality (AIVR)*.   IEEE, 2021, pp. 319–323.

[13] N. Takahashi, M. Kumar, Singh, and Y. Mitsufuji, "Hierarchical diffusion models for singing voice neural vocoder," *arXiv preprint arXiv:2210.07508*, 2022.

[14] J. Engel, L. H. Hantrakul, C. Gu, and A. Roberts, "DDSP: Differentiable digital signal processing," in *International Conference on Learning Representations*, 2020.

[15] B. Hayes, C. Saitis, and G. Fazekas, "Neural wave-shaping synthesis," in *Proc. International Society for Music Information Retrieval*, 2021.

[16] S. Shan, L. Hantrakul, J. Chen, M. Avent, and D. Trevelyan, "Differentiable wavetable synthesis," in *International Conference on Acoustics, Speech and Signal Processing (ICASSP)*.   IEEE, 2022, pp. 4598–4602.

[17] J. Alonso and C. Erkut, "Latent space explorations of singing voice synthesis using DDSP," *arXiv preprint arXiv:2103.07197*, 2021.

[18] D.-Y. Wu, W.-Y. Hsiao, F.-R. Yang, O. Friedman, W. Jackson, S. Bruzenak, Y.-W. Liu, and Y.-H. Yang, "DDSP-based singing vocoders: A new subtractive-based synthesizer and a comprehensive evaluation," in *Proc. International Society for Music Information Retrieval*, 2022.

[19] S. Nercessian, "Differentiable WORLD synthesizer-based neural vocoder with application to end-to-end audio style transfer," *arXiv preprint arXiv:2208.07282*, 2022.

[20] M. Morise, F. Yokomori, and K. Ozawa, "WORLD: a vocoder-based high-quality speech synthesis system for real-time applications," *IEICE TRANSACTIONS on Information and Systems*, vol. 99, no. 7, pp. 1877–1884, 2016.

[21] T. Yoshimura, S. Takaki, K. Nakamura, K. Oura, Y. Hono, K. Hashimoto, Y. Nankaku, and K. Tokuda, "Embedding a differentiable mel-cepstral synthesis filter to a neural speech synthesis system," *arXiv preprint arXiv:2211.11222*, 2022.

[22] S. Li, Y. Zhao, R. Varma, O. Salpekar, P. Noordhuis, T. Li, A. Paszke, J. Smith, B. Vaughan, P. Damania *et al.*, "PyTorch distributed: Experiences on accelerating data parallel training," *Proceedings of the VLDB Endowment*, vol. 13, no. 12, 2020.

[23] G. Degottex, "Glottal source and vocal-tract separation," Ph.D. dissertation, Université Pierre et Marie Curie-Paris VI, 2010.

[24] H.-L. Lu and J. O. Smith III, "Glottal source modeling for singing voice synthesis." in *International Computer Music Conference (ICMC)*, 2000.

[25] G. Fant, J. Liljencrants, Q.-g. Lin *et al.*, "A four-parameter model of glottal flow," *STL-QPSR*, vol. 4, no. 1985, pp. 1–13, 1985.

[26] J. D. Markel and A. H. Gray, *Linear Prediction of Speech*, ser. Communication and Cybernetics. Berlin, Heidelberg: Springer, 1976, vol. 12.

[27] S. Oh, H. Lim, K. Byun, M.-J. Hwang, E. Song, and H.-G. Kang, "ExcitGlow: Improving a WaveGlow-based neural vocoder with linear prediction analysis," in *Asia-Pacific Signal and Information Processing Association Annual Summit and Conference (APSIPA ASC)*. IEEE, 2020, pp. 831–836.

[28] K. Subramani, J.-M. Valin, U. Isik, P. Smaragdis, and A. Krishnaswamy, "End-to-end LPCNet: A neural vocoder with fully-differentiable LPC estimation," *arXiv preprint arXiv:2202.11301*, 2022.

[29] P. Bhattacharya, P. Nowak, and U. Zölzer, "Optimization of cascaded parametric peak and shelving filters with backpropagation algorithm," in *International Conference on Digital Audio Effects*, 2020, pp. 101–108.

[30] S. Nercessian, "Neural parametric equalizer matching using differentiable biquads," in *International Conference on Digital Audio Effects*, 2020, pp. 265–272.

[31] B. Kuznetsov, J. D. Parker, and F. Esqueda, "Differentiable IIR filters for machine learning applications," in *International Conference on Digital Audio Effects*, 2020, pp. 297–303.

[32] J. T. Colonel, C. J. Steinmetz, M. Michelen, and J. D. Reiss, "Direct design of biquad filter cascades with deep learning by sampling random polynomials," in *International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2022, pp. 3104–3108.

[33] S. Nercessian, A. Sarroff, and K. J. Werner, "Lightweight and interpretable neural modeling of an audio distortion effect using hyperconditioned differentiable biquads," in *International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2021, pp. 890–894.

[34] T. Kim, Y.-H. Yang, A. Sincia, and J. Nam, "Joint estimation of fader and equalizer gains of dj mixers using convex optimization," in *International Conference on Digital Audio Effects*. DAFx, 2022, pp. 312–319.

[35] C. J. Steinmetz, N. J. Bryan, and J. D. Reiss, "Style transfer of audio effects with differentiable signal processing," *Journal of the Audio Engineering Society*, vol. 70, no. 9, pp. 708–721, 2022.

[36] X. Serra and J. Smith, "Spectral modeling synthesis: A sound analysis/synthesis system based on a deterministic plus stochastic decomposition," *Computer Music Journal*, vol. 14, no. 4, pp. 12–24, 1990.

[37] G. Fant, "The LF-model revisited. transformations and frequency domain analysis," *Speech Trans. Lab. Q. Rep., Royal Inst. of Tech. Stockholm*, vol. 2, no. 3, p. 40, 1995.

[38] C. Gobl, "Reshaping the transformed LF model: Generating the glottal source from the waveshape parameter Rd," in *INTERSPEECH*. International Speech Communication Association (ISCA), Aug. 2017, pp. 3008–3012.

[39] Y.-Y. Yang, M. Hira, Z. Ni, A. Astafurov, C. Chen, C. Puhrsch, D. Pollack, D. Genzel, D. Greenberg, E. Z. Yang, J. Lian, J. Hwang, J. Chen, P. Goldsborough, S. Narenthiran, S. Watanabe, S. Chintala, and V. Quenneville-Bélair, "Torchaudio: Building blocks for audio and speech processing," in *International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2022, pp. 6982–6986.

[40] C.-C. Chu, F.-R. Yang, Y.-J. Lee, Y.-W. Liu, and S.-H. Wu, "MPop600: A mandarin popular song database with aligned audio, lyrics, and musical scores for singing voice synthesis," in *Asia-Pacific Signal and Information Processing Association Annual Summit and Conference (APSIPA ASC)*. IEEE, 2020, pp. 1647–1652.

[41] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," *arXiv preprint arXiv:1412.6980*, 2014.

[42] K. Kilgour, M. Zuluaga, D. Roblek, and M. Sharifi, "Fréchet audio distance: A metric for evaluating music enhancement algorithms," *arXiv preprint arXiv:1812.08466*, 2018.

[43] D. Barry, Q. Zhang, P. W. Sun, and A. Hines, "Go Listen: An end-to-end online listening test platform," *Journal of Open Research Software*, 2021.

[44] Y. Zhang, J. Hare, and A. Prugel-Bennett, "Deep set prediction networks," *Advances in Neural Information Processing Systems*, vol. 32, 2019.

[45] B. Hayes, C. Saitis, and G. Fazekas, "The responsibility problem in neural networks with unordered targets," 2023. [Online]. Available: https://openreview.net/forum?id=jd7Hy1jRiv4