
Fast Griffin Lim based Waveform Generation Strategy for Text-to-Speech Synthesis

Ankit Sharma^a · Puneet Kumar^a · Vikas Maddukuri^{b*} · Nagasai Madamshetti^b · Kishore KG^b · Sahit Sai Sriram Kavuru^b · Balasubramanian Raman^a · Partha Pratim Roy^a

Received: date / Accepted: date

Abstract The performance of text-to-speech (TTS) systems heavily depends on spectrogram to waveform generation, also known as the speech reconstruction phase. The time required for the same is known as synthesis delay. In this paper, an approach to reduce speech synthesis delay has been proposed. It aims to enhance the TTS systems for real-time applications such as digital assistants, mobile phones, embedded devices, etc. The proposed approach applies Fast Griffin Lim Algorithm (FGLA) instead Griffin Lim algorithm (GLA) as vocoder in the speech synthesis phase. GLA and FGLA are both iterative, but the convergence rate of FGLA is faster than GLA. The proposed approach is tested on LJSpeech, Blizzard and Tatoeba datasets and the results for FGLA are compared against GLA and neural Generative Adversarial Network (GAN) based vocoder. The performance is evaluated based on synthesis delay and speech quality. A 36.58% reduction in speech synthesis delay has been observed. The quality of the output speech has improved, which is advocated by higher Mean opinion scores (MOS) and faster convergence with FGLA as opposed to GLA.

Keywords Tacotron · Vocoder · Text to Speech Synthesis Delay · Dilated Convolutional Neural Network

1 Introduction

The conclusive step in a text-to-speech (TTS) system is the generation of speech from the spectrogram representation of the signal. This process is known as the waveform reconstruction while the generation of intermediate signal from input text is called the construction process. The waveform generated in the reconstruction process is the time-domain signal obtained from its intermediate spectrogram. The overall performance of a TTS system depends on the waveform processing involved in the reconstruction phase [1]. The main challenge in the TTS systems

* Corresponding author, E-mail: mvikas@ec.iitr.ac.in

^a Computer Science and Engg. Dept., Indian Institute of Technology, Roorkee, India, 247667

^b Electronics and Comm. Engg. Dept., Indian Institute of Technology, Roorkee, India, 247667

is to optimize the waveform processing time while maintaining or improving the quality of the generated speech [2].

The TTS systems have emerged as valuable tools for day-to-day applications such as digital assistants, mobile phones, embedded devices, etc. Most of these devices have limited computational capacity and they are sometimes used in off-line mode. Reducing the speech synthesis delay for them could be very useful in real-life applications such as - human computer interface, navigation systems, telecommunication and multimedia, aid to physically challenged people, daily appliances like TVs, washing machines, etc. [3]. For such applications, TTS systems are expected to have quick response time and generate speech with good quality. Hence, it becomes even more important to fasten up spectrogram to speech reconstruction for real-time applications. With the aim to make TTS systems more suitable for real-time applications, it is important to improve their response time while retaining the quality of the synthesized speech.

Many software and hardware-based techniques have been suggested in the past for waveform optimization during speech synthesis [4, 5]. Most of the traditionally used techniques were based on software components such as concatenative speech synthesis, parametric speech synthesis, etc. The current computing trend has shifted towards deep learning due to the availability of hardware resources and training data. The state-of-the-art of TTS systems have also started leveraging deep neural network (DNN) based techniques for speech synthesis [6]. The performance of text-to-speech systems has significantly improved especially after the introduction of end-to-end neural waveform generation methods [7, 8]. In spite of significant performance boost, even end-to-end neural waveform generation approaches suffer from sluggish speech reconstruction process. Therefore, there is a need to look for more efficient waveform optimization approaches to enhance the speed and quality of machine synthesized speech.

There are three stages in text-to-speech process: text analysis, linguistic analysis and waveform generation. Traditional TTS systems are based on complex multi-stage hand-engineered pipelines. The present state-of-the-art is end-to-end neural speech synthesis which puts together these stages of TTS process into a single layered pipeline through the use of DNNs. All three phases of TTS take place without human intervention for acoustic feature crafting. However, the hyper-parameters and configuration settings to cater a specific stage of the TTS can be set up-front. In that context, the implementation settings for waveform generation can be set in the form of appropriate choice of the reconstruction algorithm. Griffin-Lim algorithm (GLA) is the most predominantly used reconstruction algorithm for speech synthesis [9].

GLA is an iterative algorithm that tries to produce a signal from the spectrogram and does not have any information about phase. However, GLA needs many iterations and the perceptual quality of the output speech is not always very good [10]. An optimized version of GLA is available in the literature which is known as Fast Griffin Lim Algorithm (FGLA) [11]. FGLA naturally requires lesser iterations to construct the phase from spectrogram representation for general signal processing applications. However, it has not been applied and tested for speech synthesis. In this paper, we have applied FGLA for speech synthesis process of neural TTS systems. We've formulated an experiment to optimize the waveform processing of linear spectrograms in Tacotron TTS system. FGLA based reconstruction strat-

egy has been applied to reduce the speech synthesis delay. And, observations have been made in context of the quality of the synthesized speech and the number of iterations required for the convergence of the reconstruction algorithm.

The proposed speech synthesis systems generate the speech from the magnitude spectral envelope. We have conducted a Mean opinion scores (MOS) study to test the quality of audio produced by FGLA with a lesser number of iterations and GLA with a greater number of iterations with an optimal number of training steps. The experiments have been conducted for LJSpeech, Blizzard and Tatoeba datasets. It resulted into 36.58% reduction in speech synthesis time. The results have reflected higher quality of the output speech in terms of improved MOS. The number of training steps and iterations were determined by experimental observations. The convergence patterns of fourier transform plots of the resultant waveforms are found to be in-line with the choice of number of training iterations.

1.1 Contribution

The major contributions of the current research work are:

- An FGLA based method has been proposed to reconstruct .wav speech files from linear spectrograms. In TTS applications, reconstruction of a waveform from spectrogram plays an important role because synthesis time is equivalent to the waiting time for application users. Users expect the speech output promptly. The proposed method has reflected into reduced synthesis time which is likely to enhance the experience of TTS application users.
- The quality of the synthesis speech has been maintained while reducing the synthesis time. A market-based application cannot compromise about the quality of the synthesized speech. Speech quality also depends on the trained model. Hence, the model is trained upto optimal number of steps and the speech quality checking process has been carried out on three datasets. On all three datasets, FGLA based speech reconstruction produced better quality speech than GLA based construction.
- TTS models have been trained on LJSpeech, Tatoeba and Blizzard datasets and the waveform reconstruction has been carried out for GLA, FGLA and the GAN based vocoder. Optimal number of training steps and iterations have determined experimentally. The .wav files generated by the TTS models have been evaluated based on the quality of the output speech and the synthesis time. The speech quality has been analyzed by evaluating in terms of Mean Opinion Score (MOS) and the synthesis delay has been analyzed by measuring the time needed for the TTS model to synthesize the output speech.

1.2 Organization

The rest of the paper is organized as follows. Existing work on waveform processing for TTS systems has been surveyed in Section 2. Section 3 formulates the problem statement. The details of the proposed methodology have been outlined in Section 4. Section 5.1 presents the experimental setup. Analysis of the observed results has been presented in Section 5.2. Finally, Section 6 concludes the paper and highlights the scope for future research.

2 Literature Review

In recent years, text-to-speech processing has witnessed significant improvements. Traditionally, concatenative and parametric speech synthesis methods have been used for the task of text-to-speech conversion. In last couple of years, neural TTS systems have provided substantial performance boost in the quality of machine synthesized speech. A review of various research attempts in context of aforementioned methods alongwith the waveform optimization strategies followed by them have been provided in the following sections and their summary is presented in Table 1.

2.1 Various Speech Synthesis Methods

2.1.1 Concatenative Speech Synthesis

Concatenative models have dominated speech synthesis process since 1970's. They are based on searching and collecting small samples of speech components from the voice database [12]. The voice quality of the speech synthesized by them is more natural, however, they require a huge amount of voice database. There are two types of costs associated with them - a) searching cost and b) concatenative cost. Searching cost deals with searching specific voice segments corresponding to required broken portion and concatenative cost is related to joining these segments. As pointed by G. Coormanne et al. [13], one of the problems with these models is that they do not produce a good quality speech if a suitable match in the database corresponding to the required segment is not obtained. Another challenge with concatenative models is that they require complete dataset for generating a new set of voice. It is difficult to select the target unit from the voice database in order to minimize the difference between the required and selected samples [14].

2.1.2 Parametric Speech Synthesis

Parametric speech synthesis is another widely used process to generate speech from text. TTS models use the same statistical models derived from the data [15]. They follow a parameter generation approach unlike fetching the speech samples from the database. Hidden Markov Model (HMM) based TTS architectures are among the most famous parametric models. The primary step involved in them is to find a parametric form of speech including spectral and excitation parameters from the voice corpus and then model them by using a set of generative models [16]. In context of using HMM-based TTS systems, Y. Junichi et al. [17] predicted the parameters then synthesized the speech for a given text. The benefit of this approach is that it does not require the complete dataset at synthesis time. T. Masuko et al. [18] were success to change the speaker's voice easily using the parametric speech synthesis while L. Soojeong et al. [19] tried statistical parametric method for enhancement of the speech. However, a disadvantage associated with parametric methods is that the voice quality of the synthesized speech is not as natural as in case of concatenative speech synthesis.

2.1.3 Neural Speech Synthesis

Concatenative and parametric TTS systems have practical difficulties, for example, their different components need to be modeled and processed separately [20]. Deep learning-based end-to-end neural TTS systems, which are the current state-of-the-art solve this problem. They put together the intermediate stages of TTS process into a single, layered pipeline through the use of DNN which is carried out without human intervention for acoustic feature engineering. The recent neural TTS systems include Wavenet [21], Char2Wav [20], Tacotron [22], Tacotron 2 [23], DeepVoice [24], DeepVoice 2 [25], DeepVoice 3 [26], VoiceLoop [27]. Wavenet is based on a generative model that predicts samples based on probability distribution. Tacotron [22] produces spectrograms from the text and then produces corresponding waveform using a vocoder. However, waveform generation has been a time-consuming process for the initial TTS systems and the speech output was not human-like. Use of improved spectrogram methods such as mel-spectrogram and better vocoders such as WORLD, GLA, etc. have helped solving these problems [28].

Char2Wav predicts the parameters of the WORLD vocoder and uses a SampleRNN conditioned upon WORLD parameters for waveform generation. WORLD [29] on the other hand consists of three analysis algorithms for determining the fundamental frequency (F_0), spectral envelope and aperiodic parameters. Tacotron is another end-to-end model that uses seq-to-seq learning to map the text to spectrogram as intermediate data and then audio is generated by using vocoder. It uses Griffin Lim as the vocoder that generates audio waveforms from the linear spectrogram. It takes linear scale magnitude spectrogram and number of iterations as input and produces the corresponding waveform. It was observed that GLA in Tacotron converges in about 60 iterations [22]. Tacotron incorporates the GLA for phase estimation, followed by an inverse Short-Time Fourier transform (STFT) for waveform reconstruction. Tacotron 2 is an entirely neural network-based approach for speech synthesis which combines the seq-to-seq model feature used in Tacotron and generates the mel-spectrogram and performs speech synthesis using modified Wavenet vocoder.

In DeepVoice [24], Wavenet architecture is modified and a fast synthesis system is developed during the audio synthesis stage. DeepVoice 2 [25] is a multi-speaker model that has taken Tacotron architecture as a base and performed modification in Griffin Lim algorithm with Wavenet based vocoder. Deepvoice 3[26] is a fully convolutional attention-based neural end-to-end TTS system. Its architecture is capable of transforming several textual features into vocoder parameters such as mel-spectrograms, linear scale log spectrograms, spectral envelope, fundamental frequency (F_0), aperiodicity parameters, etc. These vocoder features are given as input to the waveform synthesis models. It uses three different vocoders - WORLD, Griffin-Lim and Wavenet. Both WORLD and Griffin Lim use linear spectrogram whereas the modified Wavenet in Tacotron2 uses mel-spectrogram for waveform synthesis. VoiceLoop [27] is an attention-based neural text to speech system referenced by a working memory model called phonological loop. It is capable of producing voices that are sampled in the wild. VoiceLoop replaces convolutional RNNs with memory buffer.

In context of using neural vocoders, K. Oyamada et al. [30] focused on DNN based architecture to recover the phase information from magnitude spectrogram. K. Kumar et al. [31] proposed the MelGAN which is a fully convolutional, non-autoregressive vocoder. It generalized well for unseen speakers and showed significant speed up in speech construction from mel-spectrograms. In a similar work, WaveGlow [32] was proposed by replacing the vocoder part of Wavenet by deep neural architecture. It performed efficiently on large utterances but its performance degraded while converting small text samples into speech. TTS with neural vocoders such as WaveGlow have to repetitively go through serial steps of waveform construction which causes them to take more time while constructing small sentences. As observed by K. Oyamada et al. [30], on CPU, some of the neural vocoders took three times longer than GLA for the speech synthesis. Their training time is as high as a few weeks and however their inference is fast on the GPU, the large size of the trained model makes their application very difficult with real-time devices having CPU with constrained memory. [32, 31, 33].

2.2 Waveform Processing in TTS systems

The final speech generated by the TTS systems is in the form of waveform while the intermediate representation is called spectrogram. The raw text input is converted into sampled embedding vector in the pre-processing phase, from which the intermediate frequency-time representation, that is, spectrogram is generated [34]. It is called the ‘Construction Phase’. Then, waveform is generated in the ‘Reconstruction Phase’ using the vocoders. The efficiency of reconstruction algorithm majorly determines the overall performance of the TTS system. GLA has been the most predominantly used reconstruction algorithm for speech synthesis [10]. A time-domain signal can be reconstructed from its amplitude spectrogram using the information about its phase. When no information is available about the phase and only the amplitude spectrogram is available, GLA is particularly suited for phase reconstruction. However, GLA needs many iterations and the perceptual quality of the output speech is not always very good [35].

There have been a number of attempts to optimize the waveform processing for speech reconstruction. For instance, Sercan et al. [10] implemented transposed convolution layers alongwith non-linear interpolation which resulted into better utilization of modern multi-core processors than simple iterative strategy. In another work, Y. Fisher [36] used multi-scale context aggregation by dilated convolutions that resulted in simplified network alongwith increased state-of-the-art accuracy. In the context of waveform processing based applications, Z. Cheng and J. Shen et al. [37] used the properties of the audio waveforms to recommend music based on the venue and surrounding of the user. As an attempt to enhance the vocoder module, M. Morise et al. [29] proposed a new vocoder, WORLD for feature extraction and waveform synthesis. Y. Masuyama [38] proposed an enhanced phase reconstruction technique by combining DNN with GLA to build GLA-inspired neural network layers for waveform generation.

Some of the distinctly related work in the area of signal processing maps to the utilization of fourier transformation techniques such as Gabor Transform [39, 40]. It is a special form of fourier transform that is used to determine the frequency and phase content of the signals represented in the form of spectrograms. In this

direction, a real-time fast fourier transform algorithm was proposed by H. Sorensen et al. [41]. Successful research has also been carried out to achieve phase recovery with lesser number of iterations as compared to GLA [10]. It gives a hint to look out for alternative reconstruction algorithms requiring lesser iterations while maintaining the quality of the synthesized speech.

Though there have been various attempts to optimize the waveform processing in context of GLA. However, better alternatives for waveform reconstruction have not been explored to their best potential. An optimized version of GLA known as Fast Griffin Lim Algorithm (FGLA) is available in the literature [11]. It requires lesser iterations to construct the phase from spectrogram representation for general signal processing applications. However, it has not been applied and tested for speech synthesis application. In this paper, FGLA based waveform generation method has been proposed with the aim to reduce synthesis delay. It aims to overcome the challenges faced by concatenative and parametric TTS systems by getting rid of the need of human intervention for acoustic feature engineering. Some of the challenges of using neural vocoders for real-time TTS applications such as - slow speech synthesis with CPU, larger model size, complex architecture, etc. have also been considered and addressed.

3 Problem Formulation

The major objective of the proposed research work is to optimize the waveform generation process during speech synthesis by TTS systems. The speech synthesis time should be reduced without changing the quality of the output speech. The training phase for TTS device is performed once in a given system until there is a change in algorithm. The synthesis phase is executed on real-time speech synthesis devices having low computational power. This phase is repeated every time a text is converted into speech. System resources at training stage are generally of high computing power. However, most of the real-time speech synthesis systems have limited computing capabilities. Hence, the synthesis algorithm should take less amount of memory to make the speech synthesis more suitable for real-time applications. The problem statement is subjected to the following constraints:

- i)* The average time taken (T) to convert the corresponding spectrogram to the waveform should be minimized. T corresponds to the synthesis delay for n samples.
- ii)* The number of iterations required (itr) for the output waveform to converge should be minimized. That is, their plots should reach to an optimal state as soon as possible.
- iii)* Quality of the synthesized speech ($qual$) should be maintained. The reduction in speech synthesis time should not affect it.
- iv)* The speech synthesis process should result into optimal resource utilization ($util$). It should cater to the limited computational resources of real-time TTS systems.

The aforementioned constraints can be modeled mathematically as shown in Eq. 1.

Table 1: Summary of literature review.

TTS Model	Basic Method	Properties	Text Pre-processing	Waveform Preprocessing	Spectrogram Type	Construction	Reconstruction
Concatenative	Searches the audio segment from speech database that is most relevant to the text	Simple to implement. Results into good quality speech	-	-	-	-	-
Parametric	Keeps track of parametric form of speech including spectral parameters	Speaker voice can be changed with minimum efforts	-	-	-	-	-
Wavenet [21]	Uses dilated regressive CNN to predict present sample from past sample	High quality. Can generate multi-speaker voice.	Yes	No	Mel-spectrogram	-	-
Char2Wav [20]	Uses bidirectional RNN to produce waveform from textual content only	Expert linguistic knowledge is not required	No	Yes	Linear-spectrogram	-	SampleRNN
Tacotron [21]	Encoder-decoder architecture based on RNNs	Fully end-to-end; robust and fast processing	Yes	Approximate	Linear-spectrogram	CBHG	GLA
Tacotron2 [23]	Encoder-decoder architecture based on RNNs	Better speech quality and smaller model size than Tacotron	Yes	Approximate	Mel-spectrogram	Convolution based	Modified Wavenet
DeepVoice [24]	Uses five different DNNs for TTS; needs less parameters and faster than Wavenet	Faster processing	No	No	Linear-spectrogram	CBHG	GLA
DeepVoice2 [25]	Wavenet based spectrogram to audio generation	Can generate multi-speaker voices with less training	No	Approximate	Linear-spectrogram	Attention based encoder	GLA, Wavenet
VoiceLoop [27]	Uses shifting buffer memory instead of RNNs	Robust; produces lesser errors	Yes	Exact	Linear-spectrogram	Buffer shallow network	WORLD

Here, GLA: Griffin Lim Algorithm; CBHG: (1-D convolution bank + highway network + bidirectional GRU)

Subject to constraints :

$$\begin{cases} \text{minimize } (T, itr) \\ \text{maximize } (qual, util) \end{cases}$$

Where :

$$\begin{cases} n : \text{number of inputs.} \\ itr : \text{number of iterations required for the output waveform to converge.} \\ qual : \text{quality of the synthesized speech.} \\ util : \text{utilization of the computing capacity of TTS platform.} \\ T = \{t_1, t_2, t_3, \dots, t_n\}, \text{ time to convert coefficients into waveforms.} \\ t_1, t_2, \dots, t_n \text{ are the times taken to convert the spectrograms to waveforms.} \\ S : \{u_1, u_2, u_3, \dots, u_n\}, \text{ set of spectrograms generated by the TTS system.} \\ u_1, u_2, \dots, u_n \text{ are the coefficient matrices of the spectrogram.} \\ W = \{v_1, v_2, v_3, \dots, v_n\}, \text{ set of corresponding waveforms generated by a vocoder.} \\ v_1, v_2, \dots, v_n \text{ are the waveforms produced from } u_1, u_2, \dots, u_n. \end{cases} \quad (1)$$

4 Methodology

In general, a TTS system contains three phases: a) text analysis (text to words), b) linguistic analysis (words to phonemes) and c) waveform generation (phonemes to sound). The first and second phases are carried out during the training phase. The TTS model is trained on text data and intermediate spectrogram is generated from the trained model for given input text. The third phase takes place during the synthesis when phase waveform is synthesized through this spectrogram. The proposed method aims to optimize the reconstruction of original speech signal from the intermediate spectrogram. Generally, a magnitude spectrogram doesn't contain the complete phase information. A reconstruction algorithm such as GLA iteratively recovers that information. GLA is an iterative algorithm that takes a high number of steps to recover the phase information. The proposed approach applies an optimized alternative FGLA, for phase reconstruction in waveform generation phase.

The waveform analysis can be performed easily in the frequency domain [39]. Fourier transform is most widely used transformation that converts time domain signal into frequency domain signal. That is why, the proposed methodology utilizes fourier transform and its variants such as Short-Time Fourier Transform (STFT), Discrete Fourier Transform (DFT), Gabor Transform, etc. during various steps of waveform generation. STFT is a series of fourier transforms of a subset of the signal. When frequency components of a signal vary with time, STFT is used to retrieve the time-localized frequency information. Gabor Transform is a special kind of STFT representation which is used to discover the phase information and sinusoidal frequency of the subsets of a time varying signal. The time-frequency analysis is carried out by first multiplying the function by a Gaussian function and then transforming it with a fourier transform. During the synthesis phase, input is a spectrogram and output is a waveform. The STFT can be represented as a matrix of coefficients where column index represents time and row index represents frequency of the respective DFT coefficient. The magnitude of each coefficient in the

respective index is computed and this matrix can be treated as a image known as spectrogram of the signal. The choice of base implementation has been explained in Section 4.1. The proposed methodology is described in the following sections. It is visually represented in Fig. 1 and mathematically depicted in Algorithm 1.

4.1 Rationale Behind Selecting Fast Griffin Lim Algorithm with Tacotron

The research work presented in this paper primarily aims to reduce the speech synthesis delay in text-to-speech systems. A fundamental experiment to apply FGLA based waveform generation from linear-spectrogram has been formulated. The most commonly used waveform reconstruction algorithm for speech synthesis is GLA. The most acclaimed TTS system that uses linear-spectrogram for intermediate representation is Tacotron, developed by Google [22]. GLA was used as vocoder in Tacotron that produces approximate waveform corresponding to the input spectrogram, not the exact waveform. FGLA has been chosen instead of GLA because FGLA is known to take lesser iterations to recover the phase from spectrogram [11]. However, other vocoders such as Wavenet [21], WORLD [29], etc. are available in the literature as potential choices for vocoders but they process mel-spectrograms while current research targeted to work with linear-spectrograms during the intermediate step of the speech synthesis process. Thus, FGLA with Tacotron emerged out as the most suitable choice for the experiment in consideration.

4.2 Strategy for Waveform Optimization

STFT is used for the observable comparative analysis in the frequency domain [9]. FGLA attempts to reconstruct the speech signal from the intermediate spectrogram of the signal. For that purpose, it finds the real signal $X^* \in R_L$ from a given set of spectral magnitude coefficients s , such that the magnitude of STFT of X^* is as close as possible to the input signal. It helps in more accurate reconstruction of the signal. Any arbitrary set of complex numbers cannot be chosen as STFT coefficients, i.e., only a certain set of complex numbers correspond to STFT of a waveform. In the same way the input that we get may not be a valid spectrogram. A valid spectrogram S would have the magnitude of the coefficients as close as possible to the input.

The relevant terms have been defined in Section 3. Two important concepts that are utilized by the proposed method are Gabor Transform and Projection. G^*x is the inverse Gabor Transform of x . It is a special case of STFT that is helpful in extracting the feature patterns from the spectrogram representation [42]. It helps in finding the time needed to convert the spectrogram into waveform [43]. Three phases of the proposed method have been described as follows.

4.2.1 Stage 1: Initialisation

- (i) First we initialize the coefficient matrix c , which is of same dimensions of the input spectrogram and contains the element of set C_1, c_0 .

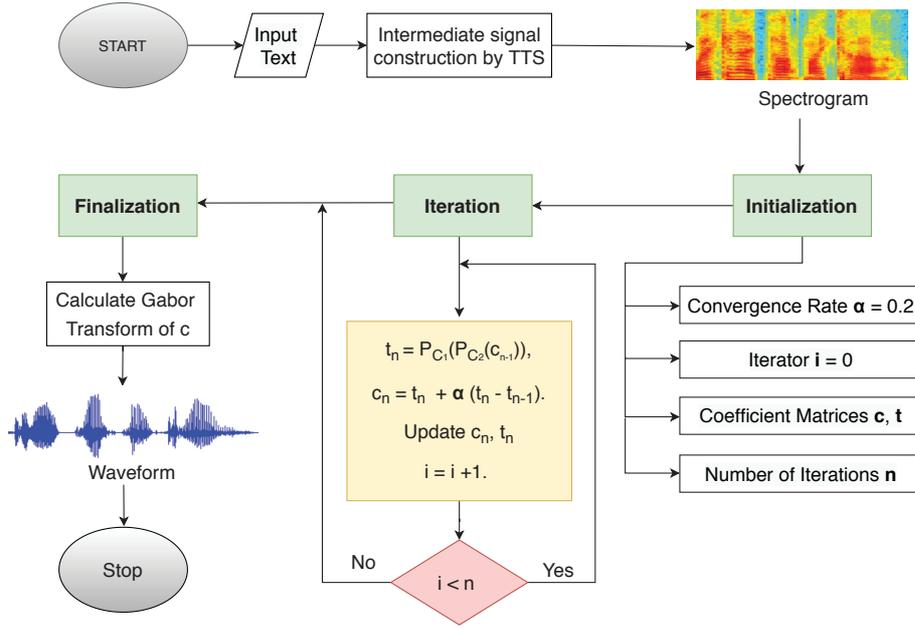


Fig. 1: Flow diagram depicting the proposed methodology. Here, green colored boxes show various stages of the methodology; red box shows the loop decision and yellow box shows the loop iteration. Theoretical analysis of the complexity and the effect of convergence rate α is discussed in Section 4.2. α is an important hyper parameter impacting the complexity and causing the speed-up in waveform processing. Its appropriate value is determined in Section 5.1.1.

- (ii) The magnitude of every element is made equal to the element in input matrix in the corresponding position. That is, projecting on to set C_2 .
- (iii) Then we initialise another matrix t of same dimensions.
- (iv) The projection of the modified coefficients of the transform on to set C_2 is defined as follows.

$$P_{C_2}(c) = s.e^{i.\angle c} \quad (2)$$

The above matrix is projected on to set C_1 , followed by projection on to set C_2 to get t_0 . Here, t_i and c_i denote the matrices after i iterations.

$$\begin{cases} C_0 = s.e^{i.\angle(c_0)} \\ t_0 = P_{C_2}(P_{C_1}(c_0)) \end{cases} \quad (3)$$

- (v) Here, C_1 is the set of possible coefficients of STFT. C_2 is the set of complex numbers whose magnitude is equal to the magnitude spectrum coefficients. G_x is the Gabor Transform of x .

$$\begin{cases} C_1 = \{c : \exists x \in R_L \|c = G_x\} \\ C_2 = \{c \in C_{MN} \| |c| = s\} \end{cases} \quad (4)$$

4.2.2 Stage 2: Iteration

- (i) The magnitude of the elements of the coefficient matrix is made equal to the input matrix, keeping the phase unchanged.
- (ii) Inverse Gabor Transform is then applied on the resultant coefficient followed by Gabor Transform. The projection of the modified coefficient on to the set C_1 is defined as follows. This is the influential step to make the FGLA faster than GLA [11].

$$\begin{cases} P_{C_1}(c) = GG^*c \\ t_n = P_{C_1}(P_{C_2}(c_{n-1})) \end{cases} \quad (5)$$

- (iii) This projection on to set C_1 in current step is subtracted with the projection in set C_1 and multiplied by a factor, convergence rate α . Choosing α close to one but not exactly one yields better results. This product is added to the projection in the current step, the initialised coefficient as updated to this value.

$$c_n = t_n + \alpha(t_n - t_{n-1}) \quad (6)$$

- (iv) The above steps are repeated iteratively and in each step the coefficients converge close to a real signal whose magnitude spectrum is approximately is equal to the input spectrum.

4.2.3 Stage 3: Final Waveform Generation

- (i) Inverse Gabor Transform is applied on the final coefficient to get the waveform.

$$x^* = G^*c_n \quad (7)$$

The aforementioned phases of the proposed methodology are depicted in Algorithm 1 and the theoretical analysis of its complexity is discussed below.

Complexity Analysis

As shown in Algorithm 1, FGLA has single iteration loop and as per GLA paper [9], GLA also involves single loop. Hence, the theoretical time complexities for both FGLA and GLA are $O(n)$. Experiments revealed that FGLA could produce the waveform of same quality with 30 iterations as compared to GLA with 60 iterations. Lesser number of iterations for FGLA is a determining factor for the reduction in the synthesis delay. Convergence rate α is another important hyper-parameter impacting the complexity and causing the speed-up in the waveform processing. Its value ranges from 0 to 1. For $\alpha = 0$, FGLA behaves as GLA. As its value increases, speed-up also increases till a limit with faster convergence and then it starts decreasing. As discussed in Section 5.1.1, the appropriate value of α has been determined as 0.2.

Algorithm 1: FGLA based Waveform Optimization for text-to-speech

Input n : Number of iterations.
Input k : Number of input variables.
Input C_1 : Set of possible coefficients of STFT.
Input C_2 : Set of complex numbers with magnitude same as of spectrum coefficients.
Define s : Spectral magnitude coefficients.
Define c : Coefficient matrix.
Define t : Matrix of same dimensions as c .
Define α : Convergence Rate.
Define x^* : Final waveform.
Define G^* : Inverse Gabor Transform of x^* .
Input $S = \{u_1, u_2, u_3, \dots, u_k\}$: set of spectrograms generated by TTS model.
Define u_1, u_2, \dots, u_k : coefficient matrices of the spectrogram.
Output $W = \{v_1, v_2, v_3, \dots, v_k\}$: set of waveforms generated by a vocoder.
Define v_1, v_2, \dots, v_k : produced waveform.
Define $T = \{t_1, t_2, t_3, \dots, t_k\}$: time to convert coefficients into waveform.
Define t_1, t_2, \dots, t_k : time to convert the spectrograms to waveforms.

Procedure WaveOpti

- 1: **Stage 1: Initialisation**
- 2: //Projection of modified transform coefficients on to set C_2
- 3: $P_{C_2}(c) = s.e^{i.\angle c}$
- 4: // Projection of above matrix on set C_1 and then C_2
- 5: $C_0 = s.e^{i.\angle(c_0)}$
- 6: $t_0 = P_{C_2}(P_{C_1}(c_0))$.
- 7: **Stage 2: Iteration**
- 8: //Projection of modified transform coefficients on to set C_2
- 9: $P_{C_1}(c) = GG^*c$
- 10: //Update t and c for each iteration
- 11: **for** i in n **do**
- 12: $t_i = P_{C_1}(P_{C_2}(c_{i-1}))$
- 13: $c_i = t_i + \alpha(t_i - t_{i-1})$.
- 14: **end for**
- 15: **Stage 3: Waveform Generation**
- 16: //Inverse Gabor Transform of final coefficients
- 17: $x^* = G^*c_n$

5 Implementation and Results

This section discusses and evaluates the experiments to apply FGLA based waveform generation from linear-spectrogram.

5.1 Experimental Set-up

This section demonstrates the experimental implementation and analyses the results. A fundamental experiment to optimize the waveform processing of linear spectrograms in Tacotron TTS system has been formulated. The model training is done on Nvidia Tesla K80 GPU machine with 24GB RAM and 4992 CUDA cores. Text to speech synthesis is done on Intel(R) Core(TM) i7-7700, 4.2 GHz CPU with 16GB RAM and 64-bit Windows 10 OS machine. The Machine Learning libraries used in this implementation are Numpy, Tensorflow and Keras. The choice of parameters and datasets has been detailed in the following sections.

Table 2: Hyper-parameter choices.

Parameter	Value
Convergence Rate, α	0.2
Sampling Rate	20000
Frame Shift (ms)	12.5
Learning Rate	0.0002

The questions that the experimental set-up tries to answer are - "What is the ideal number of training steps for TTS model training?"; "How to determine the appropriate number of iterations?"; "How to evaluate the speech synthesized by the TTS system in terms of quality, speed and convergence?" Suitable number of training steps and iterations are experimentally determined in Section 5.1.3 and 5.1.4. Then as per Table 7, use-case sentences are formulated according to various complexity levels. In Section 5.2, speech-synthesis for these sentences has been evaluated in terms of speech quality, synthesis delay and convergence.

5.1.1 Hyper-parameter Selection

During the synthesis, the convergence rate α was experimentally chosen as 0.2. We experimented with the α values starting from 0.1 with learning rate 0.0002 and performed the iterations for spectrogram to waveform construction and analysed the construction time. The most suitable value of α corresponding to the optimal construction time emerged out as 0.2. The basic entity for training the model is .text and .wav file. Wav file signal has to be sampled for the analysis. Table 2 represents the important parameters for the current analysis and their selected values. Here, 'Sampling rate' denotes the number of samples per second, 'Frame shift' specifies amount by which window will slide. 'Learning rate' shows how fast network learns by adjusting weights. The experimentally determined values for sampling rate, frameshift and learning rate are 20000, 12.5 ms and .0002 respectively. Tacotron has been trained from scratch for the datasets described in Section 5.1.2 for various number of iterations and speech synthesis time has been observed for GLA and FGLA both.

5.1.2 Datasets

The original Tacotron paper had used LJSpeech dataset. We have trained and tested the TTS model with Blizzard and Tatoeba datasets as well. The details of all the datasets used in the implementation is provided in the following sections.

- (i) **LJ Speech Dataset** [44]: It is a single-speaker, public domain speech-dataset containing 13100 audio samples ranging from 1 to 10 seconds. The total duration of the dataset is about 24 hours. Each audio-file is a single-channel 16-bit PCM WAV with a sample rate of 22050 Hz. The properties of LJSpeech dataset are detailed in Table 3.
- (ii) **Tatoeba Dataset** [45]: This audio corpus is a crowdsourced dataset of sentences and translations. It contains a subset of the English sentences of Tatoeba. We have not used the complete dataset. Sentences have been filtered out and

Table 3: Details of LJSpeech dataset

Parameters	Values
Total Clips	13,100
Total Words	225,715
Total Characters	1,308,678
Total Duration	23:55:17
Mean Clip Duration	6.57 sec
Min Clip Duration	1.11 sec
Max Clip Duration	10.10 sec
Mean Words per Clip	17.23
Distinct Words	13,821

Table 4: Details of Blizzard dataset

Audiobook Name	Total Audio Length
Tramp Abroad	15:46:01
Life on the Mississippi	14:47:27
The Man That Corrupted Hadleyburg and Other Stories	13:04:00

299152 sentences have been included in processing. The average length of the audio clips ranges from 1 to 5 seconds. It is an open dataset. Various contributors keep on adding new audio clips and sentences.

- (iii) **Blizzard Dataset** [46]: It is available under Creative Commons Attribution Share-Alike license. It contains the samples of three audiobooks read by a single American English narrator. The books name and recording time are given in Table 4. Audio file format of blizzard corpus is 16-bit WAV, mono and sampling frequency is 44100 Hz.

5.1.3 Determination of Appropriate Number of Training Steps

The quality of synthesized speech also depends on the number of training steps. However, training the model for more steps requires more computation and time. Hence, it is important to determine the optimal number of steps. We have trained the Tacotron model till 400K steps and observed the MOS values of the speech synthesized with it. MOS is a subjective evaluation score to denote the quality of a speech utterance [47]. The MOS scores corresponding to various checkpoints are shown in Table 5. It has been observed that the MOS values improves rapidly till 250k steps but their convergence slows down significantly after that. Hence, the TTS model have been trained for at least 250k steps for the final training of each use-case. This analysis is performed with configuration same as original Tacotron paper, i.e., using GLA algorithm with 60 iterations.

5.1.4 Determination of Appropriate Number of Iterations

FGLA is supposed to converge faster than GLA. However, the suitable value for FGLA's number of iterations needs to be determined effectively. With that aim,

Table 5: Determination of appropriate no. of steps.

Steps	MOS		
	LJSpeech	Tatoeba	Blizzard
40k	7.0	7.1	5
80K	7.5	7.3	5.2
120K	7.6	7.5	5.4
160K	7.6	7.6	6.0
200K	7.65	7.8	6.3
240K	7.9	8	6.5
280K	8	8.1	6.7
320K	8.2	8.25	7

Table 6: Determination of appropriate no. of iterations.

Dataset	Iterations	GLA	FGLA
LJSpeech	20	6.0	6.8
	30	7.3	8.2
	60	7.6	8.2
Tatoeba	20	7.6	7.1
	30	7.6	8.25
	60	8.1	8.25
Blizzard	20	6.0	6.2
	30	6.4	7.1
	60	6.9	7.3

we trained Tacotron with GLA and FGLA both for 20, 30 and 60 iterations respectively. Then we checked the MOS values for the audio samples synthesized with the model thus trained. These values have been illustrated in Table 6. The speech quality for FGLA in terms of MOS scores is observed to be better than that for GLA. Moreover, FGLA is also observed to take lesser number of iterations to reach same to same MOS score as compared to GLA. FGLA with 30 iterations converged to equivalent MOS values for GLA with 60 iterations. Hence, 30 was selected as the appropriate number of iterations for FGLA to be used for Tacotron’s training. The correctness of this choice has been justified in Section 5.2.2.

5.1.5 Use-case Formulation

During speech synthesis, the test sentences are chosen according to various complexity levels. Various verbal and lingual combinations in terms of punctuation marks, abbreviations, special characters, exclamation and question mark, etc. have been included to form five use-case sentences of varying lengths. As mentioned earlier, the main objective of the work presented in this paper is to reduce the synthesis delay without affecting the quality. The trained models are tested to synthesize these sentences and their synthesis delay speech quality has been observed. The time taken in the synthesis process is proportional to the length of the text. So, checked sentences have variable-sized length. Every sentence has been

Table 7: List of use-case sentences.

S.No.	Sentences
1.	He said to him, "Is not your name Ahmed?"
2.	All of a sudden, there was a loud screaming, Please help me!
3.	I think I lost my wallet! I can't find it anywhere! Oh, I could just kick myself!
4.	"Sunshine on my shoulders makes me happy, sunshine in my eyes can make me cry."
5.	As the stranger entered the town, he was met by a police, man who asked, "Are you a traveler?" "So it would appear", He replied carelessly.

synthesized 10 times and then the average synthesis time has been considered. To make periodic observations, model checkpoint has been saved after every 1000 training steps. The size of the trained model has been observed to be of the order of 80 MB. The aforementioned use cases have been depicted in Table 7 and the detailed analysis has been presented in Table 6.

5.2 Result Analysis

The implementation has been carried out considering the number of training steps and iterations determined in the above section. This section presents and verifies the results in terms of speech synthesis using the TTS model thus trained. The results have been evaluated for five use-case sentences described in Table 7. During the result evaluation, we have parallelly checked the quality of audio generated for all three corpora at different intervals of the model trained given in Table 5. Mean opinion score (MOS) is calculated based on wave file generated by synthesis on the trained model. The optimal number of training steps for different datasets for various training steps are shown in the Table 5. The results have been analysed based on the quality of the output speech and synthesis delay. The choice of number of iterations made in Section 5.1 has also been verified by observing the convergence of the waveform plots for GLA and FGLA.

5.2.1 Quality Analysis

Quality analysis results have been shown in Table 5 and 6. It was observed that, after 250k steps of model training, the output speech included prosody features. That made the voice more feasible for real-time speech synthesis. Speech quality also depends on the corpus used for training. Model is trained up to 400K steps for all datasets and results have been generated. The quality of the synthesized speech is expected to be clear to understand and non-robotic in nature. The more human-like the voice is, have a higher value of MOS and results easy to understand. X

axis shows duration of model trained and Y axis represents MOS of speech taken by 10 evaluators. The output speech has been evaluated at regular training step intervals for different data sets. The speech quality was observed to depend mainly on the number of iterations used in the algorithm. The graph shows that in the initial stage learning rate is very high but after 250k training steps learning rate is very slow. The observed results in context of the quality of the output speech are visually illustrated in Fig. 2, 3 and 4.

Accuracy of MOS Determination

To further evaluate the quality of the proposed methodology’s results, 100 text samples with known MOS scores are considered. Corresponding speech is synthesized for them using GLA, FGLA and GAN based vocoders. MOS of the synthesized speech have been evaluated. If the variation in the MOS of the synthesized speech i.e. MOS_s and the known MOS i.e. MOS_g is less than the error margin e , then the sample is assumed to be accurately determined. The error margin is taken as 0.45 which is 5% of the MOS of natural human voice [48]. The calculations are done as per Eq. 8 and the results for the considered samples are summarized in Table 8.

$$\left\{ |MOS_g - MOS_s| < e \Rightarrow \text{Accurately determined.} \right\}$$

Where :

$$\left\{ \begin{array}{l} e : \text{error margin.} \\ MOS_g : \text{ground truth MOS.} \\ MOS_s : \text{MOS of synthesized speech.} \end{array} \right. \quad (8)$$

Table 8: Accuracy of MOS determination

Vocoder	Avg. MOS	Min. MOS	Max. MOS	Accuracy
GLA	7.5	5.2	7.6	72%
FGLA	7.8	5.4	8.2	81%
GAN Vocoder	7.6	4.8	8.0	79%

The accuracy for FGLA came out to be much better than GLA and comparable to GAN based neural vocoder. It should be noted that GAN vocoder has been evaluated on CPU, in-line with the goal of optimizing the waveform processing from linear spectrograms for real-time devices with limited processing capabilities.

5.2.2 Synthesis Delay Analysis

‘Synthesis Delay’ is the time required for the output speech to start getting produced by the TTS system. The same has been computed and compared for all the use-case sentences described in Table 7 and datasets mentioned in Section 5.1.2. The observations have been drawn for GLA with 30 iterations, FGLA with 60

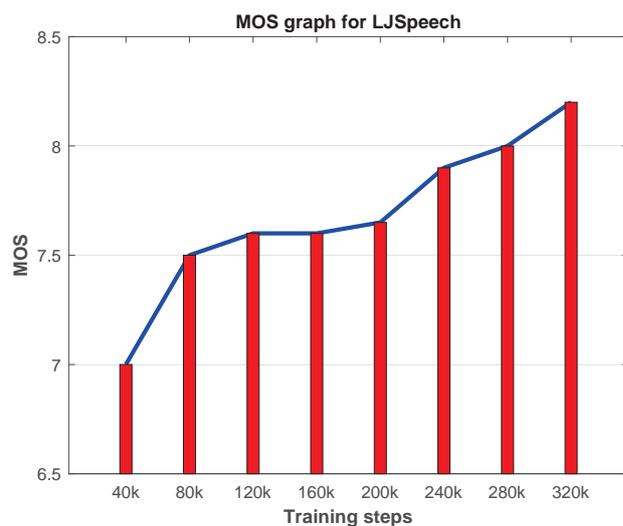


Fig. 2: Quality analysis of synthesized speech for LJSpeech dataset

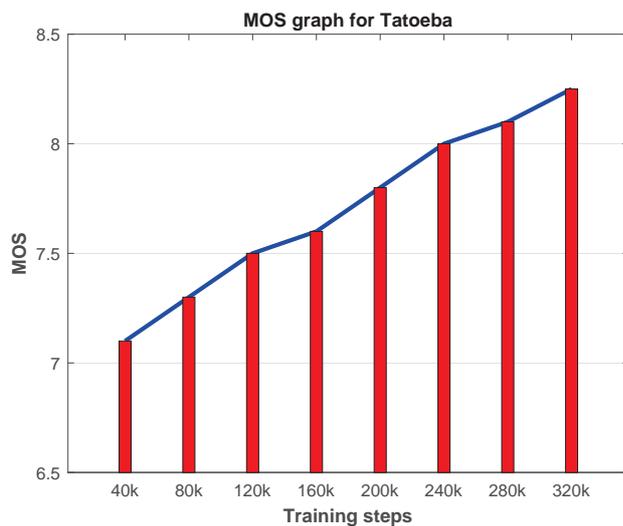


Fig. 3: Quality analysis of synthesized speech for Tatoeba dataset

iterations and GAN based neural vocoder [30]. Every sentence is synthesised 10 times and then average synthesis delay has been calculated. Same computational configuration has been maintained on the testing machines while doing so. As our aim is to optimize the waveform processing keeping CPU based low memory devices for real-time usage, the waveform reconstruction has been carried out on CPU for GLA, FGLA and the GAN based vocoder.

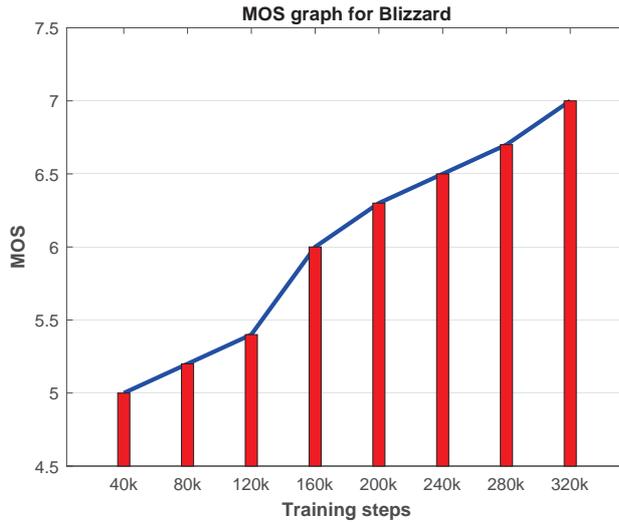


Fig. 4: Quality analysis of synthesized speech for Blizzard dataset

The observed synthesis time for the aforementioned cases have been depicted numerically in Table 9 and visually in Fig. 5. FGLA came out to be 49.12%, 33.57% and 26.52% faster than GLA in terms of Synthesis Delay for LJSpeech, Tatoeba and Blizzard datasets. The overall reduction in the Synthesis Delay has been observed to be 36.58%. While the average speech synthesis time for GAN based vocoder on CPU came out to be 3.65, 2.88 and 2.76 times more than FGLA. It should also be noted that FGLA produced better quality speech with lesser training iterations as compared to GLA. With the proposed waveform generation strategy, LJ Speech dataset has shown more reduction in the synthesis delay than other datasets. In context of MOS scores, Tatoeba dataset showed faster progress while Blizzard dataset showed lower values as compared to LJSpeech.

5.2.3 Convergence Plots

As discussed in Section 5.1.4, appropriate number of iterations for FGLA were determined as 30. Here, we have verified that choice by observing the convergence in the plots of the resulting waveforms. The comparative analysis in the frequency domain is easier to observe. That's why, fourier transformations of the waveforms are considered. The transforms of the speech produced from FLGA with 30 iterations and FLGA with 60 iterations are plotted and compared. It is found that the two plots overlap with each other. Similarly, fourier transforms of waveforms produced by GLA 60 iterations and GLA 30 iterations are plotted and compared. It is found that the plots do not overlap. This means that the waveform produced by FLGA 30 iterations and FGLA 60 iterations are same, thus a good quality of speech is produced using FLGA 30 itself, whereas, using GLA requires 60 iterations give a better quality of speech that GLA 60 iterations. The convergence plots

Table 9: Synthesis time (ms) for various use-cases and datasets

S.N.	Use-case Sentences	LJSpeech			Tatoeba			Blizzard		
		GLA (60)	FGLA (30)	GAN vocoder	GLA (60)	FGLA (30)	GAN vocoder	GLA (60)	FGLA (30)	GAN vocoder
1.	He said to him, "Is not your name Ahmed?"	10026	5173	18900	9684	6456	17950	9800	6725	16850
2.	All of a sudden, there was a loud screaming, Please help me!	9698	4940	18190	10416	6880	19950	8997	6726	18870
3.	I think I lost my wallet! I can't find it anywhere! Oh, I could just kick myself!	9702	4920	17000	10180	6799	20010	8957	6702	19010
4.	"Sunshine on my shoulders makes me happy, sunshine in my eyes can make me cry."	9849	4959	18700	10550	7010	19880	8940	6757	18550
5.	As the stranger entered the town, he was met by a police, man who asked, "Are you a traveler?" "So it would appear", He replied carelessly.	9828	4992	18560	10300	6821	20150	9129	6763	19950

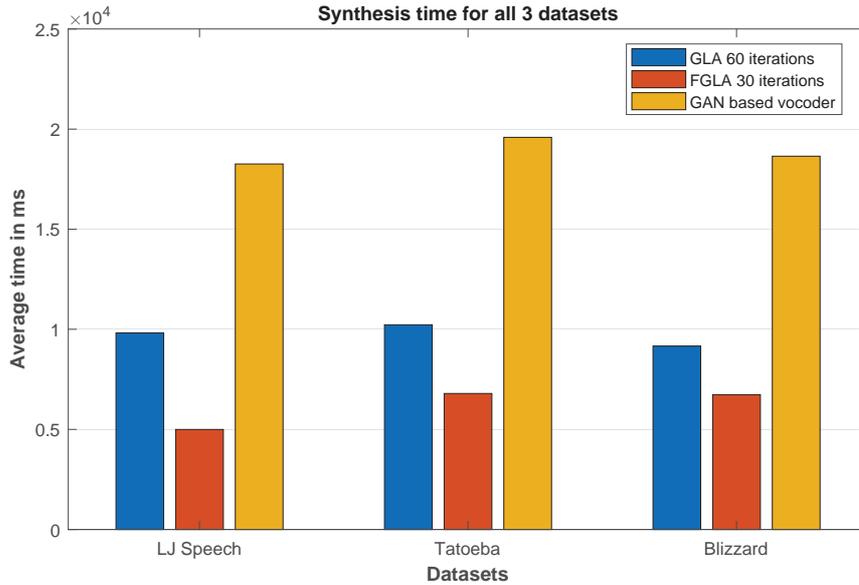


Fig. 5: Synthesis Delay for the use-cases formulated in Table 7

for the fourth sentence from Table. 7 are shown in Fig. 6 and Fig. 7 for GLA and FGLA respectively. The fourth sentence captures sufficient variations in terms of sentence length, special characters, punctuation marks, etc. The plots for the rest of the sentences are included in the supplementary material.

6 Conclusion

In this paper, a FGLA based method is proposed to optimize the waveform generation process to reduce speech synthesis delay. The final speech, i.e., waveform, is reconstructed from intermediate spectrogram. GLA has mostly been used, especially when phase information about the waveform is missing. But GLA is slow, which causes delay in the speech synthesis. A faster alternative of GLA, i.e., FGLA has been used in the proposed method that resulted in 36.58% reduction in speech synthesis time. In the presented work, experiments were performed to optimize waveform generation from linear spectrogram in single-speaker TTS systems. The proposed approach is compared against GLA and GAN based neural vocoder in terms of speech quality and synthesis delay. The quality of the synthesized speech has been checked using MOS based evaluation. The quality is observed to be retained in spite of the reduction in the synthesis time. The number of training steps and iterations were determined by experimental observation. This choice was verified through the convergence of fourier transform plots of the resultant waveforms.

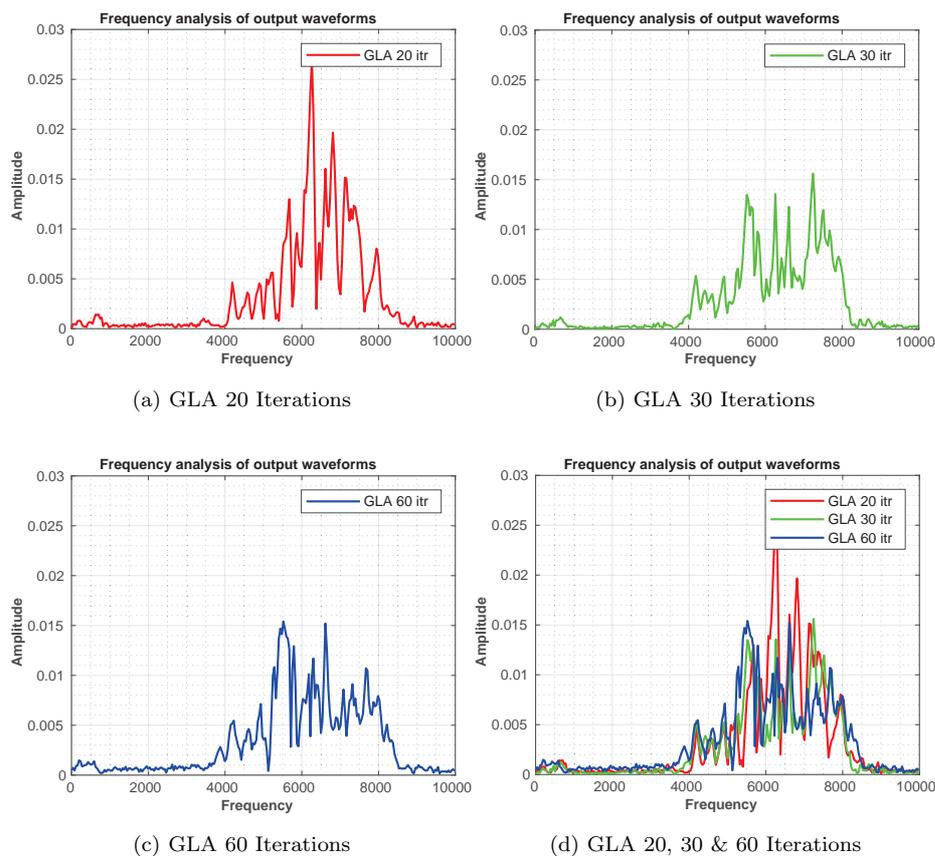


Fig. 6: GLA Convergence Plots

In future, we will work to optimize the waveform processing for the TTS systems trained with multi-speaker datasets using mel-spectrograms as intermediate representation. We will also explore more neural vocoders for speech synthesis in real-time applications. It is planned to work on the challenges involved with them such as reducing the model size and reducing the computational requirements especially for synthesizing small sentences.

References

1. Hideyuki Mizuno, Masanobu Abe, and Tomohisa Hirokawa. Waveform-based speech synthesis approach with a formant frequency modification. In *IEEE International Conference on Acoustics, Speech, and Signal Processing*, volume 2, pages 195–198, 1993.
2. Yi Zhao, Shinji Takaki, Hieu-Thi Luong, Junichi Yamagishi, Daisuke Saito, and Nobuaki Minematsu. Wasserstein gan and waveform loss-based acous-

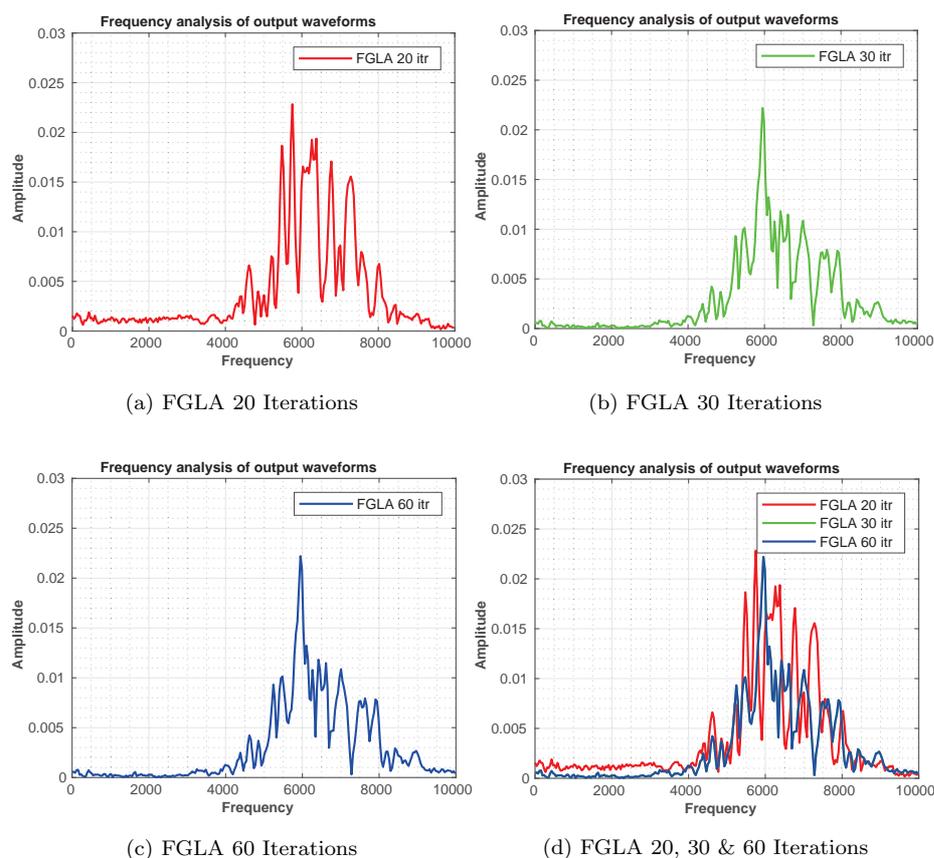


Fig. 7: FGLA Convergence Plots

tic model training for multi-speaker text-to-speech synthesis systems using a wavenet vocoder. *IEEE Access*, 6:60478–60488, 2018.

3. Boland T Jones, David Michael Guthrie, Laurence Schaefer, and J Douglas Martin. Real-time speech-to-text conversion in an audio conference session, January 31 2017. US Patent 9,560,206.
4. Pravin Ghate and S D Shirbahadurkar. A survey on methods of tts and various test for evaluating the quality of synthesized speech. *International Journal of Development Research*, 07:15236–15239, 2017.
5. Andy Aaron, Raimo Bakis, Ellen M Eide, and Wael M Hamza. Systems and methods for text-to-speech synthesis using spoken example, November 11 2014. US Patent 8,886,538.
6. Yonghui Wu, Mike Schuster, Zhifeng Chen, Quoc V Le, Mohammad Norouzi, Wolfgang Macherey, Maxim Krikun, Yuan Cao, Qin Gao, Klaus Macherey, et al. Google’s neural machine translation system: Bridging the gap between human and machine translation. *arXiv preprint arXiv:1609.08144*, 2016.
7. Suyoun Kim, Takaaki Hori, and Shinji Watanabe. Joint ctc-attention based end-to-end speech recognition using multi-task learning. In *IEEE international*

- conference on acoustics, speech and signal processing (ICASSP)*, pages 4835–4839, 2017.
8. Ye Jia, Yu Zhang, Ron Weiss, Quan Wang, Jonathan Shen, Fei Ren, Patrick Nguyen, Ruoming Pang, Ignacio Lopez Moreno, Yonghui Wu, et al. Transfer learning from speaker verification to multispeaker text-to-speech synthesis. In *Advances in neural information processing systems (NeuroIPS)*, pages 4480–4490, 2018.
 9. Daniel Griffin and Jae Lim. Signal estimation from modified short-time fourier transform. *IEEE Transactions on Acoustics, Speech, and Signal Processing*, 32(2):236–243, 1984.
 10. Yoshiki Masuyama, Kohei Yatabe, and Yasuhiro Oikawa. Griffin–lim like phase recovery via alternating direction method of multipliers. *IEEE Signal Processing Letters*, 26(1):184–188, 2018.
 11. N. Perraudin, P. Balazs, and P. L. Søndergaard. A fast griffin-lim algorithm. In *IEEE Workshop on Applications of Signal Processing to Audio and Acoustics*, pages 1–4, 2013.
 12. A. J. Hunt and A. W. Black. Unit selection in a concatenative speech synthesis system using a large speech database. In *IEEE International Conference on Acoustics, Speech, and Signal Processing Conference Proceedings*, volume 1, pages 373–376, 1996.
 13. Geert Coorman, Filip Deprez, Mario De Bock, Justin Fackrell, Steven Leys, Peter Rutten, Jan De Moortel, Andre Schenk, and Bert Van Coile. Speech synthesis using concatenation of speech waveforms, May 15 2007. US Patent 7,219,060.
 14. Keiichi Tokuda, Yoshihiko Nankaku, Tomoki Toda, Heiga Zen, Junichi Yamagishi, and Keiichiro Oura. Speech synthesis based on hidden markov models. *Proceedings of the IEEE*, 101(5):1234–1252, 2013.
 15. Heiga Zen, Keiichi Tokuda, and Alan W Black. Statistical parametric speech synthesis. *Elsevier Speech Communication*, 51(11):1039–1064, 2009.
 16. Keiichi Tokuday and Heiga Zen. Directly modeling speech waveforms by neural networks for statistical parametric speech synthesis. In *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 4215–4219, 2015.
 17. Junichi Yamagishi, Bela Usabaev, Simon King, Oliver Watts, John Dines, Jilei Tian, Rile Hu, Yong Guan, Keiichiro Oura, Keiichi Tokuda, Reima Karhila, and Mikko Kurimo. Thousands of voices for hmm-based speech synthesis–analysis and application of tts systems built on various asr corpora. *IEEE Transactions on Audio, Speech, and Language Processing*, 18:984–1004, 2009.
 18. Takashi Masuko, Keiichi Tokuda, Takao Kobayashi, and Satoshi Imai. Voice characteristics conversion for hmm-based speech synthesis system. In *1997 IEEE International Conference on Acoustics, Speech, and Signal Processing*, volume 3, pages 1611–1614, 1997.
 19. Soojeong Lee and Joon-Hyuk Chang. Spectral difference for statistical model-based speech enhancement in speech recognition. *Springer Multimedia Tools and Applications*, 76(23):24917–24929, 2017.
 20. Jose Sotelo, Soroush Mehri, Kundan Kumar, Joao Felipe Santos, Kyle Kastner, Aaron Courville, and Yoshua Bengio. Char2wav: End-to-end speech synthesis. 2017.

21. Aaron van den Oord, Sander Dieleman, Heiga Zen, Karen Simonyan, Oriol Vinyals, Alex Graves, Nal Kalchbrenner, Andrew Senior, and Koray Kavukcuoglu. Wavenet: A generative model for raw audio. *arXiv preprint arXiv:1609.03499*, 2016.
22. Yuxuan Wang, RJ Skerry-Ryan, Daisy Stanton, Yonghui Wu, Ron J Weiss, Navdeep Jaitly, Zongheng Yang, Ying Xiao, Zhifeng Chen, Samy Bengio, et al. Tacotron: Towards end-to-end speech synthesis. *arXiv preprint arXiv:1703.10135*, 2017.
23. Jonathan Shen, Ruoming Pang, Ron J Weiss, Mike Schuster, Navdeep Jaitly, Zongheng Yang, Zhifeng Chen, Yu Zhang, Yuxuan Wang, Rj Skerrv-Ryan, et al. Natural tts synthesis by conditioning wavenet on mel spectrogram predictions. In *International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 4779–4783. IEEE, 2018.
24. Sercan O Arik, Mike Chrzanowski, Adam Coates, Gregory Diamos, Andrew Gibiansky, Yongguo Kang, Xian Li, John Miller, Andrew Ng, Jonathan Raiman, et al. Deep voice: Real-time neural text-to-speech. In *Proceedings of the 34th International Conference on Machine Learning(ICML)*, volume 70, pages 195–204, 2017.
25. Andrew Gibiansky, Sercan Arik, Gregory Diamos, John Miller, Kainan Peng, Wei Ping, Jonathan Raiman, and Yanqi Zhou. Deep voice 2: Multi-speaker neural text-to-speech. In *Advances in neural information processing systems*, pages 2962–2970, 2017.
26. Wei Ping, Kainan Peng, Andrew Gibiansky, Sercan O Arik, Ajay Kannan, Sharan Narang, Jonathan Raiman, and John Miller. Deep voice 3: Scaling text-to-speech with convolutional sequence learning. *arXiv preprint arXiv:1710.07654*, 2017.
27. Yaniv Taigman, Lior Wolf, Adam Polyak, and Eliya Nachmani. Voiceloop: Voice fitting and synthesis via a phonological loop. *arXiv preprint arXiv:1707.06588*, 2017.
28. D. Griffin and Jae Lim. Signal estimation from modified short-time fourier transform. *IEEE Transactions on Acoustics, Speech, and Signal Processing*, 32(2):236–243, April 1984.
29. Masanori Morise, Fumiya Yokomori, and Kenji Ozawa. World: a vocoder-based high-quality speech synthesis system for real-time applications. *IEICE Transactions on Information and Systems*, 99(7):1877–1884, 2016.
30. Keisuke Oyamada, Hirokazu Kameoka, Takuhiro Kaneko, Kou Tanaka, Nobukatsu Hojo, and Hiroyasu Ando. Generative adversarial network-based approach to signal reconstruction from magnitude spectrogram. In *26th IEEE European Signal Processing Conference (EUSIPCO)*, pages 2514–2518, 2018.
31. Kundan Kumar, Rithesh Kumar, Thibault de Boissiere, Lucas Gestein, Wei Zhen Teoh, Jose Sotelo, Alexandre de Brebisson, Yoshua Bengio, and Aaron C Courville. Melgan: Generative adversarial networks for conditional waveform synthesis. In *Advances in Neural Information Processing Systems*, pages 14881–14892, 2019.
32. Ryan Prenger, Rafael Valle, and Bryan Catanzaro. Waveglow: A flow-based generative network for speech synthesis. In *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 3617–3621, 2019.
33. Aaron van den Oord, Yazhe Li, Igor Babuschkin, Karen Simonyan, Oriol Vinyals, Koray Kavukcuoglu, George van den Driessche, Edward Lockhart,

- Luis C Cobo, Florian Stimberg, et al. Parallel wavenet: Fast high-fidelity speech synthesis. *arXiv preprint arXiv:1711.10433*, 2017.
34. Kishore Prahallad. Speech technology: Spectrogram, cepstrum and mel-frequency analysis. https://archive.org/details/SpectrogramCepstrumAndMel-frequency_636522, 2016.
 35. Sercan O Arik, Heewoo Jun, and Gregory Diamos. Fast spectrogram inversion using multi-head convolutional neural networks. *IEEE Signal Processing Letters*, 26(1):94–98, 2018.
 36. Fisher Yu and Vladlen Koltun. Multi-scale context aggregation by dilated convolutions. *arXiv preprint arXiv:1511.07122*, 2015.
 37. Zhiyong Cheng and Jialie Shen. On effective location-aware music recommendation. *ACM Transactions on Information Systems (TOIS)*, 34(2):1–32, 2016.
 38. Yoshiki Masuyama, Kohei Yatabe, Yuma Koizumi, Yasuhiro Oikawa, and Noboru Harada. Deep griffin–lim iteration. In *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 61–65, 2019.
 39. Ronald Newbold Bracewell and Ronald N Bracewell. *The Fourier transform and its applications*, volume 31999. McGraw-Hill New York, 1986.
 40. T Malathi and Manas Kamal Bhuyan. Performance analysis of gabor wavelet for extracting most informative and efficient features. *Springer Multimedia Tools and Applications*, 76(6):8449–8469, 2017.
 41. H V Sorensen, D Jones, Michael Heideman, and C Burrus. Real-valued fast fourier transform algorithms. *IEEE Transactions on acoustics, speech, and signal processing*, 35(6):849–863, 1987.
 42. Shie Qian and Dapang Chen. Discrete gabor transform. *IEEE Transactions on signal processing*, 41(7):2429–2438, 1993.
 43. Marc Levoy. *Volume rendering using the fourier projection-slice theorem*. Computer Systems Laboratory, Stanford University, 1992.
 44. Keith Ito. The lj speech dataset. <https://keithito.com/LJ-Speech-Dataset/>, 2017.
 45. Sysko. Tatoeba speech dataset. <https://tatoeba.org/eng/>, 2013.
 46. Norbert Braunschweiler, Mark J. F. Gales, and Sabine Buchholz. Lightly supervised recognition for automatic alignment of large coherent speech recordings. In *INTERSPEECH*, 2010.
 47. Pier Luigi Salza, Enzo Foti, Luciano Nebbia, and Mario Oreglia. Mos and pair comparison combined methods for quality evaluation of text-to-speech systems. *Acta Acustica united with Acustica*, 82(4):650–656, 1996.
 48. Bret Kinsella. Speech synthesis becomes more humanlike. <https://voicebot.ai/2017/12/21/speech-synthesis-becomes-humanlike/>, 2017.