Codec 2

David Rowe

Revision: 0b6a207 on branch: dr-codec2-doc

November 26, 2023

1 Introduction

Codec 2 is an open source speech codec designed for communications quality speech between 700 and 3200 bit/s. The main application is low bandwidth HF/VHF digital radio. It fills a gap in open source voice codecs beneath 5000 bit/s and is released under the GNU Lesser General Public License (LGPL).

Key feature includes:

1. A range of modes supporting different bit rates, currently (Nov 2023): 3200, 2400, 1600, 1400, 1300, 1200, 700C. The number is the bit rate, and the supplementary letter the version (700C replaced the earlier 700, 700A, 700B versions). These are referred to as "Codec 2 3200", "Codec 2 700C" etc.

2. Modest CPU (a few 10s of MIPs) and memory (a few 10s of kbytes of RAM) requirements such that it can run on stm32 class microcontrollers with hardware FPU.

3. Codec 2 has been designed for digital voice over radio applications, and retains intelligible speech at a few percent bit error rate.

4. An open source reference implementation in the C language for C99/gcc compilers, and a *cmake* build and test framework that runs on Linux. Also included is a cross compiled stm32 reference implementation.

5. Ports to non-C99 compilers (e.g. MSVC, some microcontrollers, native builds on Windows) are left to third party developers - we recommend the tests also be ported and pass before considering the port successful.

The Codec 2 project was started in 2009 in response to the problem of closed source, patented, proprietary voice codecs in the sub-5 kbit/s range, in particular for use in the Amateur Radio service.

This document describes Codec 2 at two levels. Section 2 is a high level description aimed at the Radio Amateur, while Section 3 contains a more detailed description with math and signal processing theory. Combined with the C

source code, it is intended to give the reader enough information to understand the operation of Codec 2 in detail and embark on source code level projects, such as improvements, ports to other languages, student or academic research projects. Issues with the current algorithms and topics for further work are also included.

2 Codec 2 for the Radio Amateur

2.1 Model Based Speech Coding

A speech codec takes speech samples from an A/D converter (e.g. 16 bit samples at 8 kHz or 128 kbits/s) and compresses them down to a low bit rate that can be more easily sent over a narrow bandwidth channel (e.g. 700 bits/s for HF). Speech coding is the art of "what can we throw away". We need to lower the bit rate of the speech while retaining speech you can understand, and making it sound as natural as possible.

As such low bit rates we use a speech production "model". The input speech is anlaysed, and we extract model parameters, which are then sent over the channel. An example of a model based parameter is the pitch of the person speaking. We estimate the pitch of the speaker, quantise it to a 7 bit number, and send that over the channel every 20ms.

The model based approach used by Codec 2 allows high compression, with some trade offs such as noticeable artefacts in the decoded speech. Higher bit rate codecs (above 5000 bit/s), such as those use for mobile telephony or voice on the Internet, tend to pay more attention to preserving the speech waveform, or use a hybrid approach of waveform and model based techniques. They sound better but require a higher bit rate.

Recently, machine learning has been applied to speech coding. This technology promises high quality, artefact free speech quality at low bit rates, but currently (2023) requires significantly more memory and CPU resources than traditional speech coding technology such as Codec 2. However the field is progressing rapidly, and as the cost of CPU and memory decreases (Moore's law) will soon be a viable technology for many low bit rate speech applications.

2.2 Speech in Time and Frequency

To explain how Codec 2 works, lets look at some speech. Figure 1 shows a short 40ms segment of speech in the time and frequency domain. On the time plot we can see the waveform is changing slowly over time as the word is articulated. On the right hand side it also appears to repeat itself - one cycle looks very similar to the last. This cycle time is the "pitch period", which for this example

is around $P = 35$ samples. Given we are sampling at $F_s = 8000$ Hz, the pitch period is $P/F_s = 35/8000 = 0.0044$ seconds, or 4.4ms.

Now if the pitch period is 4.4ms, the pitch frequency or *fundamental* frequency F_0 is about $1/0.0044 \approx 230$ Hz. If we look at the blue frequency domain plot at the bottom of Figure 1, we can see spikes that repeat every 230 Hz. Turns out of the signal is repeating itself in the time domain, it also repeats itself in the frequency domain. Those spikes separated by about 230 Hz are harmonics of the fundamental frequency F_0.

Note that each harmonic has it's own amplitude, that varies across frequency. The red line plots the amplitude of each harmonic. In this example there is a peak around 500 Hz, and another, broader peak around 2300 Hz. The ear perceives speech by the location of these peaks and troughs.

2.3   Sinusoidal Speech Coding

A sinewave will cause a spike or spectral line on a spectrum plot, so we can see each spike as a small sine wave generator. Each sine wave generator has it's own frequency that are all multiples of the fundamental pitch frequency (e.g. $230, 460, 690, ...$ Hz). They will also have their own amplitude and phase. If we add all the sine waves together (Figure 2) we can produce reasonable quality synthesised speech. This is called sinusoidal speech coding and is the speech production "model" at the heart of Codec 2.

The model parameters evolve over time, but can generally be considered constant for a short time window (a few 10s of ms). For example pitch evolves over time, moving up or down as a word is articulated.

As the model parameters change over time, we need to keep updating them. This is known as the *frame rate* of the codec, which can be expressed in terms of frequency (Hz) or time (ms). For sampling model parameters Codec 2 uses a frame rate of 10ms. For transmission over the channel we reduce this to 20-40ms, in order to lower the bit rate. The trade off with a lower frame rate is reduced speech quality.

The parameters of the sinusoidal model are:

1. The frequency of each sine wave. As they are all harmonics of F_0 we can just send F_0 to the decoder, and it can reconstruct the frequency of each harmonic as $F_0, 2F_0, 3F_0, ..., LF_0$. We used 5-7 bits/frame to represent F_0 in Codec 2.

2. The amplitude of each sine wave, $A_1, A_2, ..., A_L$. These "spectral amplitudes" are really important as they convey the information the ear needs to understand speech. Most of the bits are used for spectral amplitude information. Codec 2 uses between 18 and 50 bits/frame for spectral amplitude information.

3. Voicing information. Speech can be approximated into voiced speech (vowels) and unvoiced speech (like consonants), or some mixture of the two.

Figure 1: A 40ms segment from the word "these" from a female speaker, sampled at 8kHz. Top is a plot again time, bottom (blue) is a plot against frequency. The waveform repeats itself every 4.3ms ($F_0 = 230$ Hz), this is the "pitch period" of this segment.
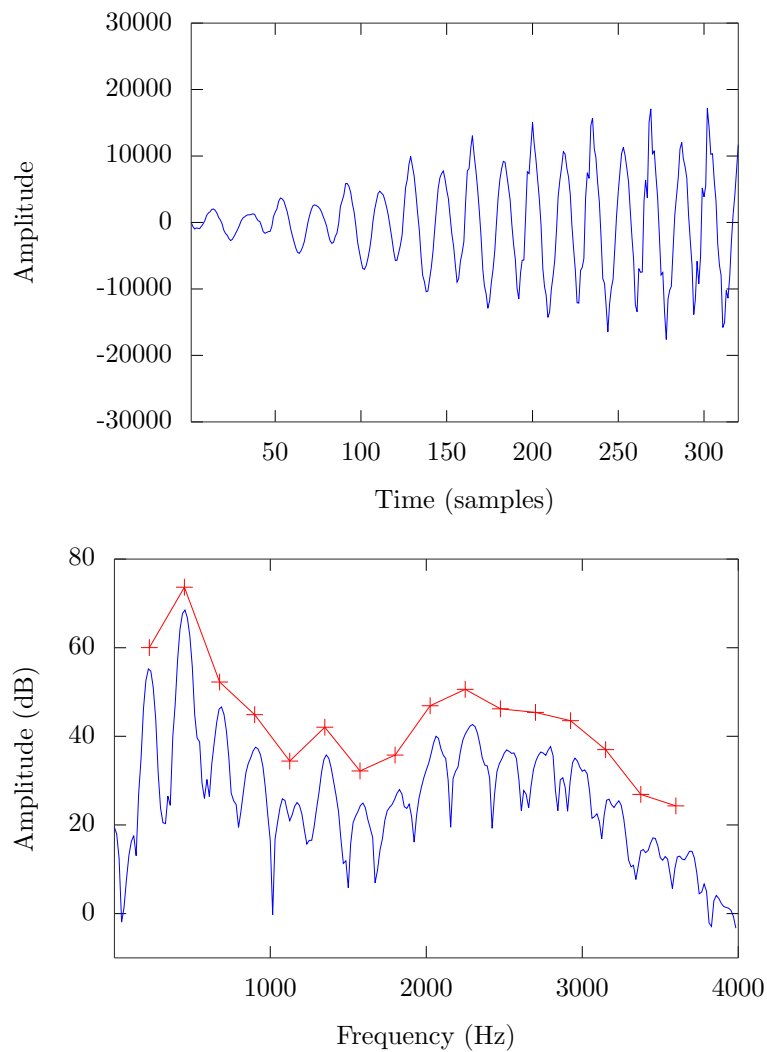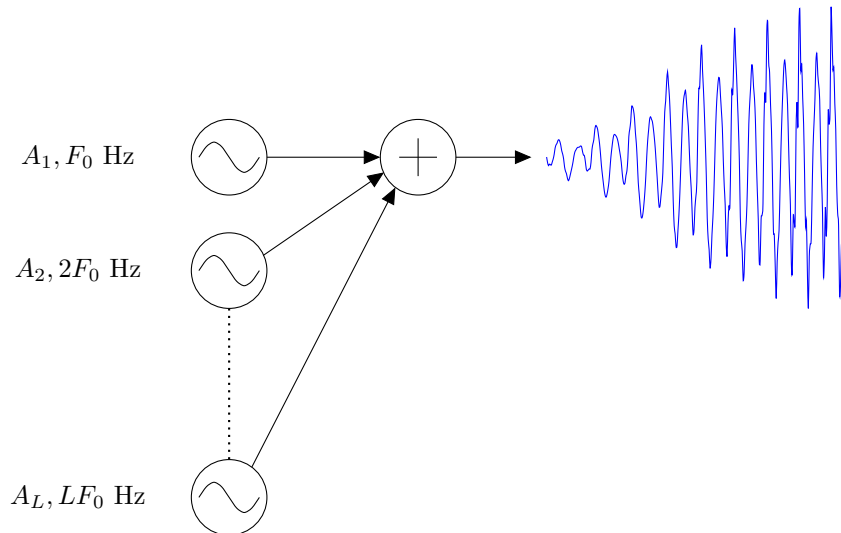
Figure 2: The sinusoidal speech model. If we sum a series of sine waves, we can generate a speech signal. Each sinewave has it's own amplitude $(A_1, A_2, ...A_L)$, frequency, and phase (not shown). We assume the frequencies are multiples of the fundamental frequency F_0. L is the total number of sinewaves we can fit in 4 kHz.

The example in Figure 1 above is voiced speech. So we need some way to describe voicing to the decoder. This requires just a few bits/frame.

4. The phase of each sine wave. Codec 2 discards the phases of each harmonic at the encoder and reconstruct them at the decoder using an algorithm, so no bits are required for phases. This results in some drop in speech quality.

2.4   Codec 2 Encoder and Decoder

This section explains how the Codec 2 encoder and decoder works using block diagrams.

The encoder is presented in Figure 3. Frames of input speech samples are passed to a Fast Fourier Transform (FFT), which converts the time domain samples to the frequency domain. The same frame of input samples is used to estimate the pitch of the current frame. We then use the pitch and frequency domain speech to estimate the amplitude of each sine wave.

Yet another algorithm is used to determine if the frame is voiced or unvoiced. This works by comparing the spectrum to what we would expect for voiced speech (e.g. lots of spectral lines). If the energy is more random and continuous rather than discrete lines, we consider it unvoiced.

Figure 3: Codec 2 Encoder
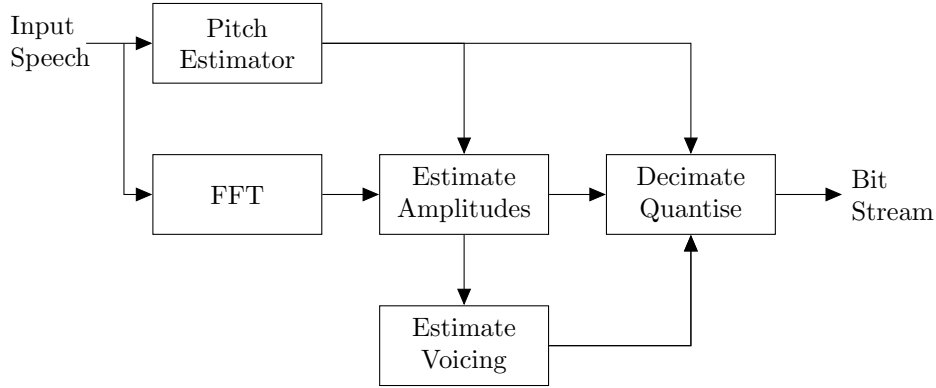
Up until this point the processing happens at a 10ms frame rate. However in the next step we "decimate" the model parameters - this means we discard some of the model parameters to lower the frame rate, which helps us lower the bit rate. Decimating to 20ms (throwing away every 2nd set of model parameters) doesn't have much effect, but beyond that the speech quality starts to degrade. So there is a trade off between decimation rate and bit rate over the channel.

Once we have the desired frame rate, we "quantise" each model parameter. This means we use a fixed number of bits to represent it, so we can send the bits over the channel. Parameters like pitch and voicing are fairly easy, but quite a bit of DSP goes into quantising the spectral amplitudes. For the higher bit rate Codec 2 modes, we design a filter that matches the spectral amplitudes, then send a quantised version of the filter over the channel. Using the example in Figure 1 - the filter would have a band pass peaks at 500 and 2300 Hz. It's frequency response would follow the red line. The filter is time varying - we redesign it for every frame.

You'll notice the term "estimate" being used a lot. One of the problems with model based speech coding is the algorithms we use to extract the model parameters are not perfect. Occasionally the algorithms get it wrong. Look at the red crosses on the bottom plot of Figure 1. These mark the amplitude estimate of each harmonic. If you look carefully, you'll see that above 2000Hz, the crosses fall a little short of the exact centre of each harmonic. This is an example of a "fine" pitch estimator error, a little off the correct value.

Often the errors interact, for example the fine pitch error shown above will mean the amplitude estimates are a little bit off as well. Fortunately these errors tend to be temporary, and are sometimes not even noticeable to the listener - remember this codec is often used for HF/VHF radio where channel noise is part of the normal experience.

Figure 4 shows the operation of the Codec 2 decoder. We take the sequence

Figure 4: Codec 2 Decoder
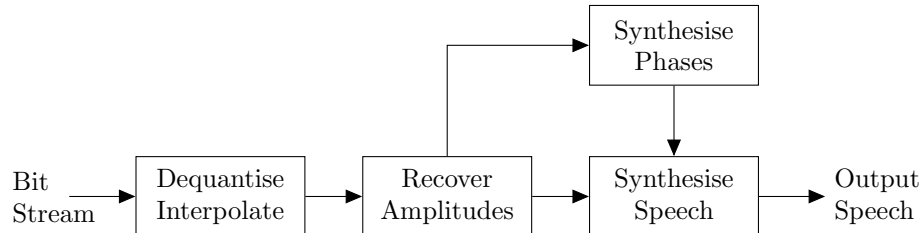
of bits received from the channel and recover the quantised model parameters, pitch, spectral amplitudes, and voicing. We then resample the model parameters back up to the 10ms frame rate using a technique called interpolation. For example say we receive a $F_0 = 200$ Hz pitch value then 20ms later $F_0 = 220$ Hz. We can use the average $F_0 = 210$ Hz for the middle 10ms frame.

The phases of each harmonic are generated using the other model parameters and some DSP. It turns out that if you know the amplitude spectrum, you can determine a "reasonable" phase spectrum using some DSP operations, which in practice is implemented with a couple of FFTs. We also use the voicing information - for unvoiced speech we use random phases (a good way to synthesise noise-like signals) - and for voiced speech we make sure the phases are chosen so the synthesised speech transitions smoothly from one frame to the next.

Frames of speech are synthesised using an inverse FFT. We take a blank array of FFT samples, and at intervals of F_0 insert samples with the amplitude and phase for each harmonic. We then inverse FFT to create a frame of time domain samples. These frames of synthesised speech samples are carefully aligned with the previous frame to ensure smooth frame-frame transitions, and output to the listener.

2.5 Bit Allocation

Table 2.5 presents the bit allocation for two popular Codec 2 modes. One additional parameter is the frame energy, this is the average level of the spectral amplitudes, or "AF gain" of the speech frame.

At very low bit rates such as 700 bits/s, we use Vector Quantisation (VQ) to represent the spectral amplitudes. We construct a table such that each row of the table has a set of spectral amplitude samples. In Codec 2 700C the table has 512 rows. During the quantisation process, we choose the table row that best matches the spectral amplitudes for this frame, then send the *index* of the table row. The decoder has a similar table, so can use the index to look up the output values. If the table is 512 rows, we can use a 9 bit number to quantise the spectral amplitudes. In Codec 2 700C, we use two tables of 512 entries each (18 bits total), the second one helps fine tune the quantisation from the first

table.

Vector Quantisation can only represent what is present in the tables, so if it sees anything unusual (for example a different microphone frequency response or background noise), the quantisation can become very rough and speech quality poor. We train the tables at design time using a database of speech samples and a training algorithm - an early form of machine learning.

Codec 2 3200 uses the method of fitting a filter to the spectral amplitudes, this approach tends to be more forgiving of small variations in the input speech spectrum, but is not as efficient in terms of bit rate.

Parameter	3200	700C
Pitch F_0	7	5
Spectral Amplitudes $\{A_m\}$	50	18
Energy	5	3
Voicing	2	1
Bits/frame	64	28
Frame Rate	20ms	40ms
Bit rate	3200	700

Table 1: Bit allocation of the 3200 and 700C modes

3 Detailed Design

3.1 Overview

Codec 2 is based on sinusoidal [3] and Multi-Band Excitation (MBE) [2] vocoders that were first developed in the late 1980s. Descendants of the MBE vocoders (IMBE, AMBE etc) have enjoyed widespread use in many applications such as VHF/UHF hand held radios and satellite communications. In the 1990s the author studied sinusoidal speech coding [4], which provided the skill set and a practical, patent free baseline for starting the Codec 2 project:

Some features of the Codec 2 Design:

1. A pitch estimator based on a 2nd order non-linearity developed by the author.

2. A single voiced/unvoiced binary voicing model.

3. A frequency domain IFFT/overlap-add synthesis model for voiced and unvoiced speech.

4. For the higher bit rate modes, spectral amplitudes are represented using LPCs extracted from time domain analysis and scalar LSP quantisation.

5. For Codec 2 700C, vector quantisation of resampled spectral amplitudes in the log domain.

6. Minimal interframe prediction in order to minimise error propagation and maximise robustness to channel errors.

7. A post filter that enhances the speech quality of the baseline codec, especially for low pitched (male) speakers.

3.2 Sinusoidal Analysis

Both voiced and unvoiced speech is represented using a harmonic sinusoidal model:

$$\hat{s}(n) = \sum_{m=1}^{L} A_m cos(\omega_0 mn + \theta_m) \tag{1}$$

where the parameters $A_m, \theta_m, m = 1...L$ represent the magnitude and phases of each sinusoid, ω_0 is the fundamental frequency in radians/sample, and $L = \lfloor \pi/\omega_0 \rfloor$ is the number of harmonics.

Figure 5 illustrates the processing steps in the sinusoidal analysis system at the core of the Codec 2 encoder. This algorithms described in this section is based on the work in [4], with some changes in notation.

Figure 5: Sinusoidal Analysis
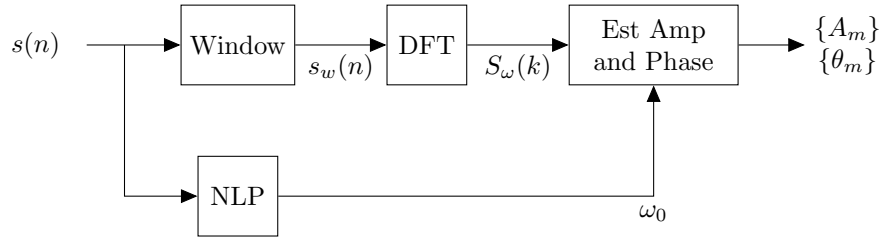
For the purposes of speech analysis the time domain speech signal $s(n)$ is divided into overlapping analysis windows (frames) of $N_w = 279$ samples. The centre of each analysis window is separated by $N = 80$ samples, or an internal frame rate or 10ms. To analyse the l-th frame it is convenient to convert the fixed time reference to a sliding time reference centred on the current analysis window:

$$s_w(n) = s(lN + n)w(n), \quad n = -N_{w2}...N_{w2} \tag{2}$$

where $w(n)$ is a tapered even window of N_w (N_w odd) samples with:

$$N_{w2} = \left\lfloor \frac{N_w}{2} \right\rfloor \tag{3}$$

A suitable window function is a shifted Hann window:

$$w(n) = \frac{1}{2} - \frac{1}{2}cos\left(\frac{2\pi(n - N_{w2})}{N_w - 1}\right) \tag{4}$$

9

where the energy in the window is normalised such that:

$$\sum_{n=0}^{N_w-1} w^2(n) = \frac{1}{N_{dft}} \tag{5}$$

To analyse $s(n)$ in the frequency domain the N_{dft} point Discrete Fourier Transform (DFT) can be computed:

$$S_w(k) = \sum_{n=-N_{w2}}^{N_{w2}} s_w(n)e^{-j2\pi kn/N_{dft}} \tag{6}$$

The magnitude and phase of each harmonic is given by:

$$\begin{aligned}
A_m &= \sqrt{\sum_{k=a_m}^{b_m-1} |S_w(k)|^2} \\
\theta_m &= arg\left(S_w(m\omega_0 N_{dft}/2\pi)\right) \\
a_m &= \left\lfloor \frac{(m-0.5)\omega_0 N_{dft}}{2\pi} + 0.5 \right\rfloor \\
b_m &= \left\lfloor \frac{(m+0.5)\omega_0 N_{dft}}{2\pi} + 0.5 \right\rfloor
\end{aligned} \tag{7}$$

The DFT indexes a_m, b_m select the band of $S_w(k)$ containing the m-th harmonic. The magnitude A_m is the RMS level of the energy in the band containing the harmonic. This method of estimating A_m is relatively insensitive to small errors in $F0$ estimation and works equally well for voiced and unvoiced speech. The phase is sampled at the centre of the band. For all practical Codec 2 modes the phase is not transmitted to the decoder so does not need to be computed. However speech synthesised using the phase is useful as a control during development, and is available using the *c2sim* utility.

3.3 Sinusoidal Synthesis

Synthesis is achieved by constructing an estimate of the original speech spectrum using the sinusoidal model parameters for the current frame. This information is then transformed to the time domain using an Inverse DFT (IDFT). To produce a continuous time domain waveform the IDFTs from adjacent frames are smoothly interpolated using a weighted overlap add procedure [3].

The synthetic speech spectrum is constructed using the sinusoidal model parameters by populating a DFT array $\hat{S}_w(k)$ with weighted impulses at the harmonic centres:

$$\begin{aligned}
\hat{S}_w(k) &= \begin{cases} A_m e^{j\theta_m}, & m = 1..L \\ 0, & otherwise \end{cases} \\
k &= \left\lfloor \frac{m\omega_0 N_{dft}}{2\pi} + 0.5 \right\rfloor
\end{aligned} \tag{8}$$

Figure 6: Sinusoidal Synthesis. At frame l we have $2N$ samples from the windowing function. The first N complete the current frame and are the synthesiser output. The second N are stored for summing with the next frame.
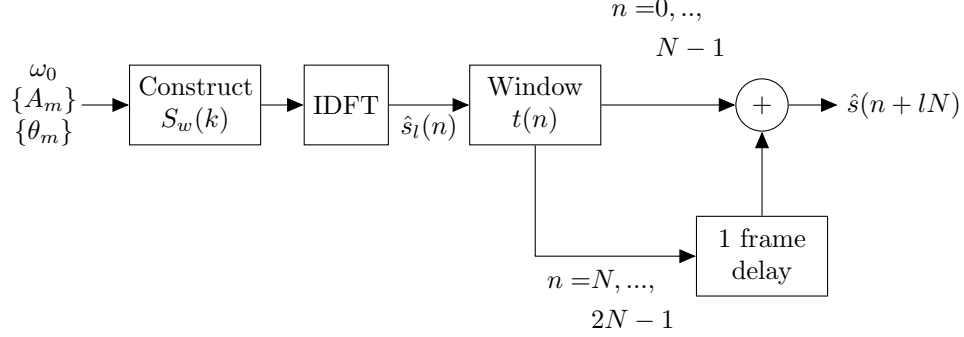
As we wish to synthesise a real time domain signal, $S_w(k)$ is defined to be conjugate symmetric:

$$\hat{S}_w(N_{dft} - k) = \hat{S}_w^*(k), \quad k = 1, ..N_{dft}/2 - 1 \tag{9}$$

where $\hat{S}_w^*(k)$ is the complex conjugate of $\hat{S}_w(k)$. This signal is converted to the time domain using the IDFT:

$$\hat{s}_l(n) = \frac{1}{N_{dft}} \sum_{k=0}^{N_{dft}-1} \hat{S}_w(k) e^{j2\pi kn/N_{dft}} \tag{10}$$

Where $N_{dft} > 2N$, to support the overlap add procedure below.

We introduce the notation $\hat{s}_l(n)$ to denote the synthesised speech for the l-th frame. To reconstruct a continuous synthesised speech waveform, we need to smoothly connect adjacent synthesised frames of speech. This is performed by windowing each frame of synthesised speech, then shifting and superimposing adjacent frames using an overlap add algorithm. A triangular window is defined by:

$$t(n) = \begin{cases} n/N, & 0 \le n < N \\ 1 - (n - N)/N, & N \le n < 2N \\ 0, & otherwise \end{cases} \tag{11}$$

The frame size, $N = 80$, is the same as the encoder. The shape and overlap of the synthesis window is not important, as long as sections separated by the frame size (frame to frame shift) sum to 1:

$$t(n) + t(N - n) = 1 \tag{12}$$

11

The continuous synthesised speech signal $\hat{s}(n)$ for the l-th frame is obtained using:

$$\hat{s}(n+lN) = \begin{cases} \hat{s}(n + (l-1)N) + \hat{s}_l(N_{dft} - N + 1 + n)t(n), & n = 0, 1, ..., N - 2 \\ \hat{s}_l(n - N - 1)t(n) & n = N - 1, .., 2N - 1 \end{cases} \tag{13}$$

From the N_{dft} samples produced by the IDFT (10), after windowing we have $2N$ output samples. The first N output samples $n = 0, ...N - 1$ complete the current frame l and are output from the synthesiser. However we must also compute the contribution to the next frame $n = N, N+1, ..., 2N-1$. These are stored, and added to samples from the next synthesised frame.

3.4 Non-Linear Pitch Estimation

The Non-Linear Pitch (NLP) pitch estimator was developed by the author, and is described in detail in chapter 4 of [4], and portions of this description are reproduced here. The post processing algorithm used for pitch estimation in Codec 2 is different from [4] and is described here. The C code *nlp.c* is a useful reference for the fine details of the implementation, and the Octave script *plnlp.m* can by used to plot the internal states and single step through speech, illustrating the operation of the algorithm.

The core pitch detector is based on a square law non-linearity, that is applied directly to the input speech signal. Given speech is composed of harmonics separated by F_0 the non-linearity generates intermodulation products at F_0, even if the fundamental is absent from the input signal due to high pass filtering.

Figure 7 illustrates the algorithm. The fundamental frequency F_0 is estimated in the range of 50-400 Hz. The algorithm is designed to take blocks of $M = 320$ samples at a sample rate of 8 kHz (40 ms time window). This block length ensures at least two pitch periods lie within the analysis window at the lowest fundamental frequency.

The speech signal is first squared then notch filtered to remove the DC component from the squared time domain signal. This prevents the large amplitude DC term from interfering with the somewhat smaller amplitude term at the fundamental. This is particularly important for male speakers, who may have low frequency fundamentals close to DC. The notch filter is applied in the time domain and has the experimentally derived transfer function:

$$H(z) = \frac{1 - z^{-1}}{1 - 0.95z^{-1}} \tag{14}$$

Before transforming the squared signal to the frequency domain, the signal is low pass filtered and decimated by a factor of 5. This operation is performed to limit the bandwidth of the squared signal to the approximate range of the fundamental frequency. All energy in the squared signal above 400 Hz is superfluous and would lower the resolution of the frequency domain peak picking stage. The low pass filter used for decimation is an FIR type with 48 taps and

Figure 7: The Non-Linear Pitch (NLP) algorithm
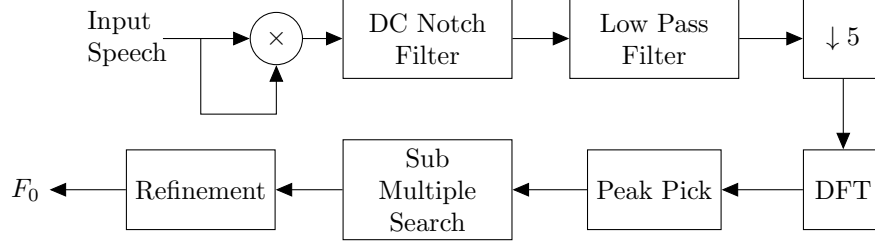
a cut off frequency of 600 Hz. The decimated signal is then windowed and the $N_{dft} = 512$ point DFT power spectrum $F_w(k)$ is computed by zero padding the decimated signal, where k is the DFT bin.

The DFT power spectrum of the squared signal $F_w(k)$ generally contains several local maxima. In most cases, the global maxima will correspond to F_0, however occasionally the global maxima $|F_w(k_{max})|$ corresponds to a spurious peak or multiple of F_0. Thus it is not appropriate to simply choose the global maxima as the fundamental estimate for this frame. Instead, we look at submultiples of the global maxima frequency $k_{max}/2, k_{max}/3, ...k_{min}$ for local maxima. If local maxima exists and is above an experimentally derived threshold we choose the submultiple as the F_0 estimate. The threshold is biased down for F_0 candidates near the previous frames F_0 estimate, a form of backwards pitch tracking.

The accuracy of the pitch estimate in then refined by maximising the function:

$$E(\omega_0) = \sum_{m=1}^{L} |S_w(bm\omega_0)|^2 \tag{15}$$

where the $\omega_0 = 2\pi F_0/F_s$ is the normalised angular fundamental frequency in radians/sample, b is a constant that maps a frequency in radians/sample to a DFT bin, and S_ω is the DFT of the speech spectrum for the current frame. This function will be maximised when mF_0 aligns with the peak of each harmonic, corresponding with an accurate pitch estimate. It is evaluated in a small range about the coarse F_0 estimate.

There is nothing particularly unique about this pitch estimator or it's performance. There are occasional artefacts in the synthesised speech that can be traced to "gross" and "fine" pitch estimator errors. In the real world no pitch estimator is perfect, partially because the model assumptions around pitch break down (e.g. in transition regions or unvoiced speech). The NLP algorithm could benefit from additional review, tuning and better pitch tracking. However it appears sufficient for the use case of a communications quality speech codec, and is a minor source of artefacts in the synthesised speech. Other pitch estimators could also be used, provided they have practical, real world implementations

that offer comparable performance and CPU/memory requirements.

3.5 Voicing Estimation

3.6 LPC/LSP based modes

3.7 Codec 2 700C

4 Further Work

1. Some worked examples aimed at the experimenter - e.g. using c2sim to extract and plot model parameters

2. How to use Octave tools to single step through codec operation

3. Table summarising source files with one line description

4. Add doc license (Creative Commons?)

5 Glossary

Symbol	Description	Units
a_m	lower DFT index of current band	
b_m	upper DFT index of current band	
b	Constant that maps a frequency in radians to a DFT bin	
$\{A_m\}$	Set of harmonic magnitudes $m = 1,...L$	dB
F_0	Fundamental frequency (pitch)	Hz
F_s	Sample rate (usually 8 kHz)	Hz
$F_w(k)$	DFT of squared speech signal in NLP pitch estimator	
L	Number of harmonics	
P	Pitch period	ms or samples
$\{\theta_m\}$	Set of harmonic phases $m = 1,...L$	dB
ω_0	Fundamental frequency (pitch)	radians/sample
$s(n)$	Input speech	
$s_w(n)$	Time domain windowed input speech	
$S_w(k)$	Frequency domain windowed input speech	

Table 2: Glossary of Symbols

References

[1] Enhancing HF Digital Voice with FreeDV, 2023. https://www.ardc.net/apply/grants/2023-grants/ enhancing-hf-digital-voice-with-freedv/.

[2] Daniel W Griffin and Jae S Lim. Multiband excitation vocoder. *IEEE Transactions on acoustics, speech, and signal processing*, 36(8):1223–1235, 1988.

[3] Robert McAulay and Thomas Quatieri. Speech analysis/synthesis based on a sinusoidal representation. *IEEE Transactions on Acoustics, Speech, and Signal Processing*, 34(4):744–754, 1986.

[4] David Grant Rowe. *Techniques for harmonic sinusoidal coding.* Citeseer, 1997. `https://www.rowetel.com/downloads/1997_rowe_phd_thesis.pdf`.