Digital storage and analysis of speech

4 – Fourier transforms and estimating formant position

by Ian H. Witten, M.A.,M.Sc., Ph.D., M.I.E.E., University of Calgary

Dr Witten continues his discussion of spectral analysis with an explanation of the discrete Fourier and fast Fourier transforms, and shows how to estimate the positions of formants.

Discrete Fourier transform

Let us return from the brief digression into techniques of digital signal analysis to the problem of determining the frequency spectrum of speech. Although a bank of bandpass filters such as is used in the channel vocoder is perhaps the most straightforward way to obtain a frequency spectrum, there are other techniques which are in fact more commonly used in digital speech processing.

It is possible to define the Fourier transform of a discrete sequence of points. To motivate the definition, consider first the ordinary Fourier transform (FT), which is

$$g(t)=\int_{-\infty}^{\infty} G(f)e^{+i2\pi ft}df$$

$$G(f)=\int_{-\infty}^{\infty} g(t)e^{-i2\pi ft}dt,$$

This takes a continuous time domain into a continuous frequency domain. Sometimes you see a normalizing factor $1/2\pi$ multiplying the integral in either the forward or the reverse transform. This is only needed when the frequency variable is expressed in radians/s, and we will find it more convenient to express frequencies in Hz.

The Fourier series (FS), which should also be familiar to you, operates on a periodic time waveform (or, equivalently, one that only exists for a finite period of time, which is notionally extended periodically). If a period lies in the time ranges $[0,b)$, then the transform is

$$g(t)=\sum_{r=-\infty}^{\infty} G(r)e^{+i2\pi rt/b}$$

$$G(r)=\frac{1}{b}\int_0^b g(t)e^{-i2\pi rt/b}\,dt.$$

The Fourier series takes a periodic time-domain function into a discrete frequency-domain one. Because of the basic duality between the time and frequency domains in the Fourier transforms, it is not surprising that another version of the transform can be defined which takes a periodic frequency domain function into a discrete time-domain one.

Fourier transforms can only deal with a finite stretch of a time signal by assuming that the signal is periodic, for if $g(t)$ is evaluated from its transform $G(r)$ according to the formula above, and t is chosen outside the interval [0,b), then a periodic extension of the function $g(t)$ is obtained automatically. Furthermore, periodicity in one domain implies discreteness in the other. Hence if we transform a finite stretch of a discrete time waveform, we get a frequency-domain representation which is also finite (or, equivalently, periodic), and discrete. This is the discrete Fourier transform (DFT), and takes a discrete periodic time-domain function into a discrete periodic frequency-domain one, as illustrated in Fig. 14. It is defined by

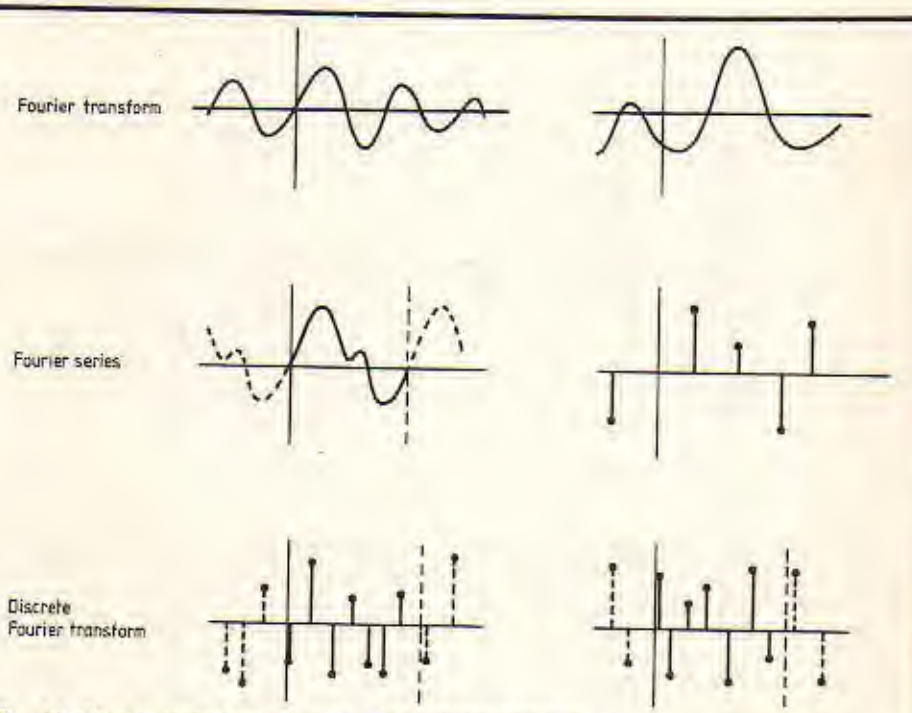


Fig. 14. Fourier transform, Fourier series and discrete FT.

$$g(n)=\frac{1}{N}\sum_{r=0}^{N-1} G(r)e^{i2\pi rn/N}$$

$$G(r)=\sum_{n=0}^{N-1} g(n)e^{-i2\pi rn/N},$$

or, writing $W=e^{-i2\pi/N}$,

$$g(n)=\frac{1}{N}\sum_{r=0}^{N-1} G(r)W^{-rn}$$

$$G(r)=\sum_{n=0}^{N-1} g(n)W^{rn}.$$

The $1/N$ in the first equation is the same normalizing factor as the $1/b$ in the Fourier series, for the finite time domain is $[0,N)$ in the discrete case and $[0,b)$ in the Fourier series case. It does not matter whether it is written into the forward or the reverse transform, but it is usually placed as shown above as a matter of convention.

As illustrated by Fig. 15, discrete Fourier transforms take an input of N real values, representing equally spaced time samples in the interval $[0,b)$, and produce as output N complex values, representing equally spaced frequency samples in the interval $[0,N/b)$. Note that the end-point of this frequency interval is the sampling frequency. It seems odd that the input is real and the output is the same number of *complex* quantities: we seem to be getting some numbers for nothing! However, this isn't so, for it is easy to show that if the input sequence is real, the output frequency spectrum has a symmetry about its mid-point (half the sampling frequency). This can be expressed as

DFT symmetry: $G(\frac{N}{2}+r)=G(\frac{N}{2}-r)^*$

if g is real-valued, where * denotes the conjugate of a complex quantity (that is, $(a+jb)^*=(a-jb)$.

It was argued above that the frequency spectrum in the DFT is periodic, with the spectrum from 0 to the sampling fre-

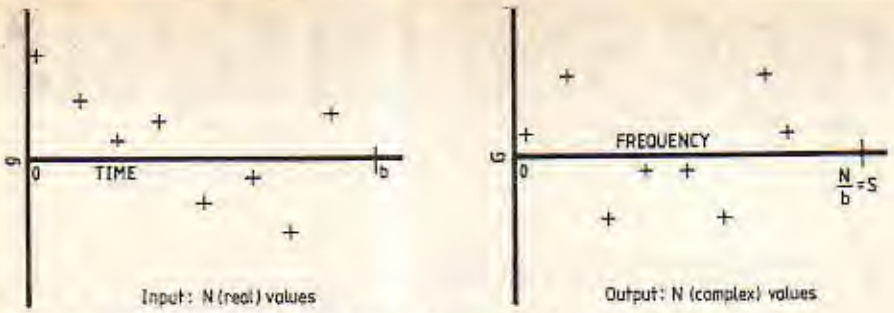


Fig. 15. Time and frequency domains for discrete Fourier transform (DFT).

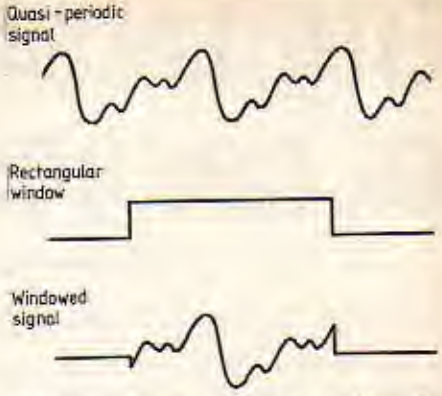


Fig. 17. Isolating part of waveform for analysis – windowing.

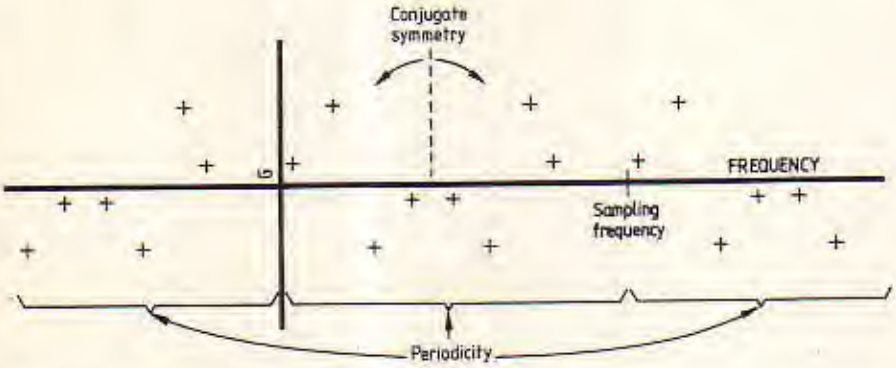


Fig. 16. Symmetry and periodicity in DFT.

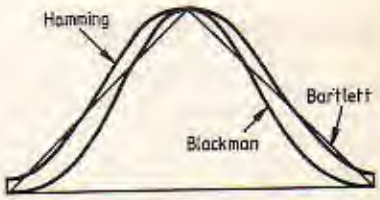


Fig. 18. Three window shapes to reduce effects of discontinuities at beginning and end of window period.

quency being repeated regularly up and down the frequency axis. It can easily be seen from the DFT equation that this is so. It can be written

DFT periodicity: $G(N+r) = G(r)$ always.

Figure 16 illustrates the properties of symmetry and periodicity.

Estimating the frequency spectrum of speech using the DFT

Speech signals are not exactly periodic. Although the waveform in a particular period will usually resemble those in the preceding and following pitch periods, it will certainly not be identical to them. As the articulation of the speech changes, the formant positions will alter. Furthermore, the pitch itself is certainly not constant, because the intonation of speech varies continually. Hence the fundamental assumption of the DFT, that the waveform is periodic, is not really justified. However, the signal is quasi-periodic, for changes from period to period will not usually be very great. One way of computing the short-term frequency spectrum of speech is to use pitch-synchronous Fourier transformation, where signal pitch periods are isolated from the waveform and processed with the DFT. This gives a rather accurate estimate of the spectrum. Unfortunately, it is difficult to determine the beginning and end of each pitch cycle, as we shall see later in this article when discussing pitch extraction techniques.

If a finite stretch of a speech waveform is isolated and Fourier transformed, without regard to pitch of the speech, then the periodicity assumption will be grossly violated. Figure 17 illustrates that the effect is the same as multiplying the signal by a rectangular window function, which is 0 except during the period to be analysed, where it is 1. The windowed sequence will almost certainly have discontinuities at its edges, and these will effect the resulting spectrum. The effect can be analysed quite easily, but we will not do so here. It is enough to say that the high frequencies associated with the edges of the window cause considerable distortion of the spectrum. The effect can be alleviated by using a smoother window than a rectangular one, and several have been investigated extensively. The commonly-used windows of Bartlett, Blackman, and Hamming are illustrated in Fig. 18.

Because the DFT produces the same number of frequency samples, equally spaced, as there were points in the time waveform, there is a tradeoff between frequency resolution and time resolution (for a given sampling rate). For example, a 256-point transform with sampling rate of 8 kHz gives the 256 equally-spaced frequency components between 0 and 8 kHz that are shown in Table 4. The top half of the frequency spectrum is of no interest, because it contains the complex conjugates of the bottom half (in reverse order), corresponding to frequencies greater than half the sampling frequency. Thus for a 30 Hz resolution in the frequency domain, 256 time samples, or a 32 ms stretch of speech, needs to be transformed. A common technique is to take overlapping periods in the time domain to give a new frequency spectrum every 16 ms. From the acoustic point of view this is a reasonable rate to recompute the spectrum, for as noted above when discussing channel vocoders the rate of change in the spectrum is limited by the speed that the speaker can move his vocal organs, and anything between 10 and 25 ms is a reasonable figure for transmitting or storing the spectrum.

time domain sample number	time	frequency domain sample number	frequency
0	0 μsec	0	0 Hz
1	125	1	31
2	250	2	62
3	375	3	94
4	500	4	125
. . .		. . .	
. . .		. . .	
. . .		. . .	
254	31750	254	7938
255	31875	255	7969

Table 4 Time domain and frequency domain samples for a 256-point DFT with 8 kHz sampling figure for transmitting or storing the spectrum.

The DFT is a complex transform, and speech is a real signal. It is possible to do two DFTs at once by putting one time waveform into the real parts of the input and another into the imaginary parts. This destroys the DFT symmetry property, for it only holds for real inputs. But given the DFT of a complex sequence formed in this way, it is easy to separate out the DFTs of the two real time sequences. If the two time sequences are $x(n)$ and $y(n)$, then the transform of the complex sequence

$$g(n) = x(n) + jy(n)$$

is $$G(r) = \sum_{n=0}^{N-1} [x(n)W^{rn} + y(n)W^{rn}].$$

It follows that the complex conjugate of the aliased parts of the spectrum, in the upper frequency region, are

$$G(N-r)^* = \sum_{n=0}^{N-1} [x(n)W^{-(N-r)n} - y(n)W^{-(N-r)n}],$$

and this is the same as

$$G(N-r)^* = \sum_{n=0}^{N-1} [x(n)W^{rn} - y(n)W^{rn}],$$

because W^N is 1 (recall the definition of W), and so W^{-Nn} is 1 for any n. Thus

$$X(r) = \frac{G(r)+G(N-r)^*}{2}$$

$$Y(r) = \frac{G(r)-G(N-r)^*}{2}$$

extracts the transforms $X(r)$ and $Y(r)$ of the original sequences x and y.

With speech, this trick is frequently used to calculate two spectra at once. Using 256-point transforms, a new estimate of the spectrum can be obtained every 16 ms by taking overlapping 32 ms stretches of speech, with a computational requirement of one 256-point transform every 32 ms.

The fast Fourier transform

Straightforward calculation of the DFT, expressed as

$$G(r) = \sum_{n=0}^{N-1} g(n)W^{rn},$$

for $r = 0, 1, 2, \ldots, N-1$, takes N^2 operation, where each operation is a complex multiply and add (for W is, of course, a complex number). There is a better way, invented in the early sixties, which reduces this to $N \log_2 N$ operations – a very considerable improvement. Dubbed the "fast Fourier transform" (FFT) for historical reasons, it would actually be better called the "Fourier transform", with the straightforward method above known as the "slow Fourier transform"! There is no reason nowadays to use the slow method, except for tiny transforms. It is worth describing the basic principle of the FFT, for it is surprisingly simple.

It is important to realize that the FFT involves no approximation. It is an *exact* calculation of the values that would be obtained by the slow method. Problems of aliasing and windowing occur in all discrete Fourier transforms, and they are neither alleviated nor exacerbated by the FFT.

To gain insight into the working of the FFT, imagine the sequence $g(n)$ split into halves, containing the even and odd points respectively.

even half $e(n)$ is $g(0)\ g(2)\ \ldots\ g(N-2)$

odd half $o(n)$ is $g(1)\ g(3)\ \ldots\ g(N-1)$.

Then is is easy to show that if G is the transform of g, E the transform of e, and O that of o, then

$$G(r) = E(r) + W^r O(r) \text{ for } r = 0, 1, \ldots, \frac{N}{2}-1,$$

and

$$G(\frac{N}{2}+r) = E(r) + W^{\frac{N}{2}+r} O(r) \text{ for } 0, 1, \ldots, \frac{N}{2}-1.$$

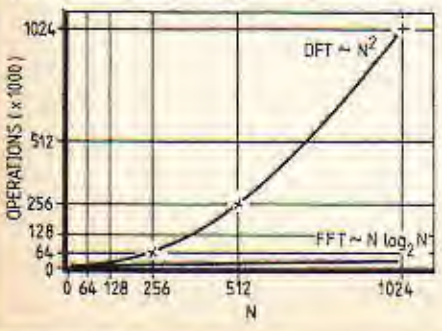
Calculation of the E and O transforms involves $(N/2)^2$ operations each, while combining them together according to the above relationship occupies N operations. Thus the total is $N + N^2/2$ operations, which is considerably less than N^2.

But don't stop there! The even half can itself be broken down into even and odd parts to expedite its calculation, and the same with the odd half. The only constraint is that the number of elements in the sequences splits exactly into two at each stage. Providing N is a power of 2, then, we are left at the end with some 1-point transforms to do. But transforming a single point leaves it unaffected! (Check the definition of the DFT.) A quick calculation shows that the number of operations needed is not $N + N^2/2$, but $N \log_2 N$. Figure 19 compares this with N^2, the number of operations for straightforward DFT calculation, and it can be seen that the FFT is very much faster.

The only restriction on the use of the FFT is that N must be a power of two. If it is not, alternative, more complicated, algorithms can be used which give comparable computational advantages. However, for speech processing the number of samples that are transformed is usually arranged to be a power of two. If a pitch synchronous analysis is undertaken, the time stretch that is to be transformed is dictated by the length of the pitch period, and will vary from time to time. Then, it is usual to pad out the time waveform with zeros to bring the number of samples up to a power of two; otherwise, if different-length time stretches were transformed the scale of the resulting frequency components would vary too.

The FFT provides very worthwhile cost savings over the use of a bank of bandpass filters for spectral analysis. Take the example of a 256-point transform with 8 kHz sampling, giving 128 frequency components spaced by 31.25 Hz from 0 up to almost 4 kHz. This can be computed on overlapping 32 ms stretches of the time waveform, giving a new spectrum every 16 ms, by a single FFT calculation every 32 ms (putting successive pairs of time stretches in the real and imaginary parts of the complex input sequence, as described earlier). The FFT algorithm requires $N \log_2 N$ operations, which is 2048 when $N = 256$. An additional 512 operations are required for the windowing calculation. Repeated every 32 ms, this gives a rate of 80,000 operations per second. To achieve a much lower frequency resolution with 20 bandpass filters, each of which are fourth-order, will need a great many more operations. Each filter needs between four and eight multiplications per sample, depending on its exact digital implementation. But new samples appear every 125 *micro*seconds, and so somewhere around a million operations are required every second. If we increased the frequency resolution to that obtained by the FFT, 128 filters would be needed, requiring between 4 and 8 million operations!

Fig. 19. Fast Fourier transform (FFT) requires many fewer operations than DFT. Size of transform plotted horizontally.



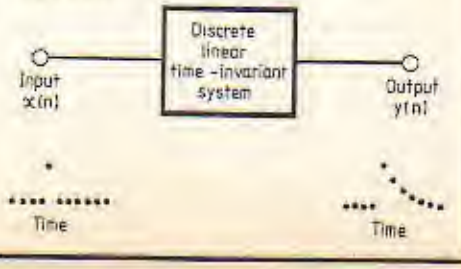
Formant estimation

Once the frequency spectum of a speech signal has been calculated, it may seem a simple matter to estimate the positions of the formants. But it is not! One reason for this is that, unless the analysis is pitch-synchronous, the frequency spectrum of the excitation source is mixed in with that of the vocal tract filter. There are other reasons, which will be discussed later in this section. But first, let us consider how to extract the vocal tract filter characteristics from the combined spectrum of source and filter. To do so we must begin to explore the theory of linear systems.

Discrete linear systems. Figure 20 shows an input signal exciting a filter to produce an output signal. For present purposes, imagine the input to be a glottal waveform, the filter a vocal tract one, and the output a speech signal (which is then subjected to high-frequency de-emphasis by radiation from the lips). We will consider here discrete systems, so that the input $x(n)$ and output $y(n)$ are sampled signals, defined only when n is integral. The theory is quite similar for continuous systems.

Assume that the system is linear; that is, if input $x_1(n)$ produces output $y_1(n)$ and input $x_2(n)$ produces output $y_2(n)$, then the sum of $x_1(n)$ and $x_2(n)$ will produce the sum of $y_1(n)$ and $y_2(n)$. It is easy to show from this that, for any constant multiplier a, the input $ax(n)$ will produce output $ay(n)$ – it is pretty obvious when $a=2$, or indeed any positive integer; for then $ax(n)$ can be written as $x(n)+x(n)+\ldots$. Assume further that the system is time-invariant; that is, if input $x(n)$ produces output $y(n)$ then a time-shifted version of x, say

Fig. 20. Linear system with input and output, with impulsive input and corresponding output.



$x(n+n_0)$ for some constant n_0, will produce the same output, only time-shifted; namely $y(n+n_0)$.

Now consider the discrete delta function $\delta(n)$, which is 0 except at $n=0$ when it is 1. If this single impulse is presented as input to the system, the output is called the impulse response, and will be denoted by $h(n)$. The fact that the system is time-invariant guarantees that the response does not depend upon the particular time at which the impulse occurred, so that, for example, the impulsive input $\delta(n+n_0)$ will produce output $h(n+n_0)$. A delta-function input and corresponding impulse response are shown in Fig. 20.

The impulse response of a linear, time-invariant system is an extremely useful thing to know, for it can be used to calculate the output of the system for any input at all! Specifically, an input signal $x(n)$ can be written

$$x(n)=\sum_{k=-\infty}^{\infty} x(k)\delta(n-k),$$

because $\delta(n-k)$ is non-zero only when $k=n$, and so for any particular value of n, the summation contains only one non-zero term – $x(n)$. The action of the system on each term of the sum is to produce an output $x(k)h(n-k)$, because $x(k)$ is just a constant, and the system is linear. Furthermore, the complete input $x(n)$ is just the sum of such terms, and since the system is linear, the output is the sum of $x(k)h(n-k)$. Hence the response of the system to an arbitrary input is

$$y(n)=\sum_{k=-\infty}^{\infty} x(k)h(n-k).$$

This is called a *convolution sum*, and is sometimes written

$$y(n)=x(n) * h(n).$$

Let's write this in terms of z-transforms. The (two-sided) z-transform of $y(n)$ is

$$Y(z)=\sum_{n=-\infty}^{\infty} y(n)z^{-n}=\sum_{n}\sum_{k} x(k)h(n-k)z^{-n}.$$

Writing z^{-n} as $z^{-(n-k)}z^{-k}$, and interchanging the order of summation, this becomes

$$Y(z)=\sum_{k}[\sum_{n} h(n-k)z^{-(n-k)}]x(k)z^{-k}$$
$$=\sum_{k} H(z)z^{-k}=H(z)\sum_{k} x(k)z^{-k}=H(z)X(z).$$

Thus convolution in the time domain is the same as multiplication in the z-transform domain; a very important result. Applied to the linear system of Fig. 20, this means that the output z-transform is the input z-transform multiplied by the z-transform of the system's impulse response.

What we really want to do is to relate the frequency spectrum of the output to the response of the system and the spectrum of the input. In fact, frequency spectra are very closely connected with z-transforms. A periodic signal $x(n)$ which repeats every N samples has DFT

$$\sum_{n=0}^{N-1} x(n)e^{-j2\pi rn/N},$$

and its z-transform is

$$\sum_{n=-\infty}^{\infty} x(n)z^{-n}.$$

Hence the DFT is the same as the z-transform of a single cycle of the signal, evaluated at the points $z=e^{j2\pi r/N}$ for $r=0,1,\ldots,N-1$. In other words, the frequency components are samples of the z-transform at N equally-spaced points around the unit circle. Hence the frequency spectrum at the output of a linear system is the product of the input spectrum and the frequency response of the system itself (i.e., the transform of its impulse-response function). It should be admitted that this statement is somewhat questionable, because to get from z-transforms to DFTs we have assumed that a single cycle only is transformed – and the impulse response function of a system is not necessarily periodic. The real action of the system is to multiply z-transforms, not DFTs. However, it is useful in imagining the behaviour of the system to think in terms of products of DFTs; and in practice it is always these rather than z-transforms which are computed because of the existence of the FFT algorithm.

The DFT frequency spectrum of a typical voiced speech signal shows humps at the formant positions. However, superimposed on this is an "oscillation" (in the frequency domain!) at the pitch frequency. This occurs because the transform of the vocal tract filter has been multiplied by that of the pitch pulse, the latter having components at harmonics of the pitch frequency. The oscillation must be suppressed before the formants can be estimated to any degree of accuracy.

One way of eliminating the oscillation is to perform pitch-synchronous analysis. This removes the influence of pitch from the frequency domain by dealing with it in the time domain! The snag is, of that it is not easy to estimate the pitch frequency: some techniques for doing so are discussed in the next main section. Another method is to remove the pitch ripple from the frequency spectrum directly. This will be discussed next, in an intuitive rather than a theoretical way.

Cepstral processing of speech. Suppose the rippled frequency spectrum were actually a time waveform. To remove the high-frequency pitch ripple is easy: just filter it out! However, filtering removes additive ripples, whereas this is a multiplicative ripple. To turn multiplication into addition, take logarithms. Then the procedure would be

– compute the DFT of the speech waveform (windowed, overlapped);
– take the logarithm of the transform;
– filter out the high-frequency part, corresponding to pitch ripple.

Filtering is often best done using the DFT. If the rippled waveform is transformed, a strong component could be expected at the ripple frequency, with weaker ones at its harmonics. These components can be simply removed by setting them to zero, and inverse-transforming the result to give a smoothed version of the original frequency spectrum. A spectrum of the logarithm of a frequency spectrum is often called a cepstrum – a sort of backwards spectrum. The horizontal axis of the cepstrum, having the dimension of time, is called "quefrency"! Note that high-frequency signals have low quefrencies and vice versa. In practice, because the pitch ripple is usually well above the quefrency of interest for formants, the upper end of the cepstrum is often simply cut off from a fixed quefrency which corresponds to the maximum pitch expected. However, identifying the pitch peaks of the cepstrum has the useful byproduct of giving the pitch period of the original speech.

To summarize, then, the procedure for spectral smoothing by the cepstral method is

– compute the DFT of the speech waveform (windowed, overlapped);
– take the logarithm of the transform;
– take the DFT of this log-transform, calling it the cepstrum;
– identify the lowest-quefrency peak in the spectrum as the pitch, confirming it by examining its harmonics, which should be equally spaced at the pitch quefrency;
– remove pitch effects from the cepstrum by cutting off its high-quefrency part above either the pitch frequency or some constant representing the maximum expected pitch (i.e. minimum expected pitch frequency);
– inverse DFT the resulting cepstrum to give a smoothed spectrum.

Estimating formants from smoothed spectra. The difficulties of formant extraction are not over even when a smooth frequency spectrum has been obtained. A simple peak-picking algorithm which identifies a peak at the k'th frequency component whenever

$$X(k-1)<X(k) \text{ and } X(k)<X(k+1)$$

will quite often identify formants incorrectly. It helps to specify in advance minimum and maximum formant frequencies – say 100 Hz and 3 kHz for three-formant identification, and ignore peaks lying outside these limits. It helps to estimate the bandwidth of the peaks and reject those with bandwidths greater than 500 Hz – for real formants are never this wide. However, if two formants are very close, then they may appear as a single, wide, peak and be rejected by this criterion. It is usual to take account of formant positions identified in previous frames under these conditions.

There are several estimation algorithms. The simplest uses the number of peaks identified in the raw spectrum (under 3 kHz, and with bandwidths greater than 500 Hz), to determine what to do. If exactly three peaks are found, they are used as the formant positions It is claimed that this happens about 85% to 90% of the time. If only one peak is found, the present frame is ignored and the previously-identified formant positions are used (this hap-

pens less than 1% of the time). The remaining cases are two peaks – corresponding to omission of one formant – and four peaks – corresponding to an extra formant being included. (More than four peaks do not normally occur.) Under these conditions, a nearest-neighbour measure can be used for identification. A suitable measure is

$$v_{ij} = |F^*_i(k) - F_j(k-1)|,$$

where $F_j(k-1)$ is the j'th formant frequency defined in the previous frame $k-1$ and $F^*_i(k)$ is the i'th raw data frequency estimate for frame k. If two peaks only are found, this measure is used to identify the closest peaks in the previous frame; and then the third peak of that frame is taken to be the missing formant position. If four peaks are found, the measure is used to determine which of them is furthest from the previous formant values, and this one is discarded.

This procedure works forwards, using the previous frame to distinguish peaks given in the current one. More sophisticated algorithms work backwards as well, identifying anchor points in the data which have clearly defined formant positions, and moving in both directions from these to identify neighbouring frames of data. Finally, absolute limits can be imposed upon the magnitude of formant movements between frames to give an overall smoothing to the formant tracks.

Very often, people will refine the result of such automatic formant estimation procedures by hand, looking at the tracks, knowing what was said, and making adjustments in the light of their experience of how formants move in speech. Unfortunately, it is difficult to obtain high-quality formant tracks by completely automatic means.

One of the most difficult cases in formant estimation is where two formants are so close together that the individual peaks cannot be resolved. One simple solution to this problem is to employ "analysis-by-synthesis", whereby once a formant is identified, a standard formant shape at this position is synthesized and subtracted from the logarithmic spectrum. Then, even if two formants are right on top of each other, the second is not missed because it remains after the first one has been subtracted.

Unfortunately, however, the single peak which appears when two formants are close together usually does not correspond exactly with the position of either one. There is one rather advanced signal-processing technique that can help in this case. The frequency spectrum of speech is determined by *poles* which lie in the complex z-plane inside the unit circle. (They must be inside the unit circle if the sytem is stable. Those familiar with Laplace analysis of analogue systems may like to note that the left half of the s-plane corresponds with the inside of the unit circle in the z-plane.) As shown earlier, computing a DFT is tantamount to evaluating the z-transform at equally-spaced points around the unit circle. However, better resolution is obtained by evaluating around a circle which lies inside the unit circle, but outside the outermost pole position. Such a circle is sketched in Fig. 21.

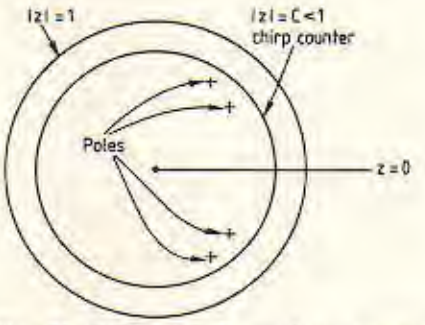


Fig. 21. Evaluating Z-transform outside outer pole but inside unit circle.

Recall that the FFT is a fast way of calculating the DFT of a sequence. Is there a similarly fast way of evaluating the z-transform inside the unit circle? The answer is yes, and the technique is known as the "chirp z-transform", because it involves considering a signal whose frequency increases linearly – just like a radar chirp signal. The chirp method allows the z-transform to be computed quickly at equally-spaced points along spirally-shaped contours around the origin of the z-plane – corresponding to signals of linearly incresing complex frequency. The spiral nature of these curves is not of particular interest in speech processing. What is of interest, though, is that the spiral can begin at any point on the $z=0$ axis, and its pitch can be set arbitrarily. If we begin spiralling at $z=0.9$, say, and set the pitch to zero, the contour becomes a circle inside the unit one, with radius 0.9. Such a circle is exactly what is needed to refine formant resolution.

To be continued

Literature Received

Catalogue of passive and active electronic components, hardware and tools, which includes a greater number of optoelectronic devices than usual, can be obtained by writing to HB Electronics, Norfolk House, Wellesley Road, Croydon CR0 0YF on company notepaper.

A variety of noise sources, from basic diodes to programmable generators is produced by Micronetics, who offer a catalogue through distributors March Microwave Ltd, 112 South Street, Braintree, Essex. WW401

Crow of Reading's capabilities in the design and construction of broadcast television equipment, from single instruments to large stations, is briefly described in a colour brochure which can be obtained from Crow of Reading Ltd, PO Box 36, Reading, Berks. WW402

A range of seven microterminals made by Burr-Brown are illustrated and shortly specified in a brochure, available from Burr-Brown International Ltd, Cassiobury House, 11-19 Station Road, Watford, Hertfordshire WD1 1EA. WW403

Catalogue of small tools and a selection of hardware is produced by Electroware, who describe their range in a new catalogue, which is obtainable from Dutton Lane, Eastleigh SO5 4SL. WW404

Voltage regulator i.cs to provide current up to 8A positive and 1.5A negative, and a range of switching power supplies for up to 50A are made by Lambda. Brochures can be had on application to Lambda Electronics Co., Abbey Barn Road, High Wycombe, Bucks. WW405

Production equipment for the electronics industry (cutting, stripping, bending and cleaning) is described in a leaflet produced by Eraser International Ltd, Unit M, Portway Industrial Estate, Andover SP10 3LU. WW406

P.r.o.ms and programmable logic devices from many makers are detailed in a new wall-chart from Microsystem Services, Duke Street, High Wycombe, Bucks. HP13 6EE. WW407

Booklet from the Electric Cable Makers' Confederation lists the member companies by name and by product, and includes a short resumé of each company's activities. The confederation's address is 56 Palace Road, East Molesey, Surrey KT8 9DW. WW408

Catalogue of general electronic components and tools, including a wide range of semiconductors and the well-known audio modules, can be obtained from Bi-Pak, the Maltings, 63a High Street, Ware, Herts. SG12 9AD. WW409

Link Electronics have produced a guide to their closed-circuit television equipment and systems, including cameras, studio equipment, complete studios and mobile units. It is available from Link Electronics Ltd, North Way, Andover SP10 5AJ. WW410

The Acron 505 Sync. pulse generator, for PAL, NTSC and PAL-M is described in a colour leaflet from Acron Video, Unit 3, Lovelace Road, Bracknell RG12 4YT. WW411

Communications, logic and memory devices in ISO-CMOS technology, which confers high speed at low power, will shortly be introduced by GTE, who have sent us a short description of the new devices. Copies can be obtained from GTE Microcircuits, 2000 W. 14th Street, Tempe, Ariz., 85281. WW412

Zilog's newsletter Z-Bits is now in its second issue, the latest one including details of the Z-Lab development system for sixteen users, new peripheral devices, Z8003 and 8004 c.p.us, a cross-assembler for Intel dev. systems and a new 4K by 8-bit quasi-static r.a.m. Copies can be had from Zilog (UK) Ltd, Babbage House, King Street, Maidenhead, Berks. SL6 1DU. WW413

A vast range of test and measuring instruments is fully described in the 1981/82 catalogue of instruments from Electroplan Ltd, PO Box 19, Orchard Road, Royston, Herts. SG8 5HH.