COMPUTATION OF THE COMPLEX CEPSTRUM BY FACTORIZATION OF THE Z-TRANSFORM*

Kenneth Steiglitz and Bradley Dickinson

Department of
Electrical Engineering and Computer Science
Princeton University
Princeton, N. J. 08540

Abstract

We compute the complex cepstrum of a finite duration signal by completely factoring its z-transform and summing the easily obtained complex cepstra of the factors. The method requires no phase unwrapping and introduces no aliasing. The only approximation is the unavoidable finite time window. Thus the factorization method provides a means for evaluating other methods for computing the complex cepstrum.

I. Introduction

The main difficulty in the application of complex cepstral analysis is the computation of the phase of the transform (z-transform on the unit circle) of the signal. The usual method requires phase unwrapping; that is, the phase of the transform is computed modulo 2π, using the FFT for example, and the multiples of 2π required to render the phase a continuous function of frequency are determined by successive refinement of the grid of frequency points [1]. Recently a new algorithm has been proposed by Tribolet [2] in which the derivative of the transform phase is adaptively integrated. For signals with transforms having zeros very close to the unit circle, these methods tend to be unreliable and/or computationally expensive as we will illustrate.

If one is willing to factor the z-transform of the signal, direct calculation of the complex cepstrum is possible, with the following three advantages:

1) No phase unwrapping is necessary.
2) No aliasing effects are present in the complex cepstrum due to using an inverse DFT of a sample transform.
3) A part of the complex cepstrum can be computed without necessarily computing any other part.

*This work was supported by NSF Grants GK-42048 and ENG75-10533, and by U.S. Army Research Office - Durham Grant DAAG29-75-G-0192.

At first, one might think it impractical to factor the z-transform of a signal, which is typically a polynomial of 255-th degree. For sampled voiced speech signals, however, it turns out that the polynomials are easily factored; empirically we have found the roots to be nicely spread out and near the unit circle. The Newton-Raphson method [3] has been found to be very effective in obtaining the roots in all the synthetic and natural speech signals we have examined.

In this paper, we will give the details of the complex cepstrum computation, discuss the polynomial factorization algorithm, and compare this method with Tribolet's algorithm on example analyses of synthetic and natural speech signals.

II. Details of Complex Cepstrum Computation by Factorization

The following discussion follows Oppenheim and Schafer [1]. Assume we are given a finite sample of a signal, f_0, $f_1,\ldots,f_{n-1}$; where, without loss of generality, we assume $f_0=1$. Its z-transform may be written

$$F(z) = \sum_{i=0}^{n-1} f_i z^{-1} = \prod_{i=0}^{n-1} (1 - z_i z^{-1}) \qquad (1)$$

The zeros z_i are obtained by numerical factorization of the (n-1)st degree polynomial F. Further assume that no zero z_i is precisely on the unit circle, and define the index sets I_1 and I_2 to correspond to zeros inside and outside the unit circle, respectively:

$$I_1 = \{i \,|\, |z_i| < 1\}$$
$$I_2 = \{i \,|\, |z_i| > 1\} \qquad (2)$$

then

$$F(z) = \prod_{i \in I_1} (1 - z_i z^{-1}) \prod_{i \in I_2} (1 - z z_i^{-1}) \prod_{i \in I_2} (-z z_i^{-1}) \qquad (3)$$

and

$$\log F(z) = \sum_{i \in I_1} \log(1 - z_i z^{-1}) + \sum_{i \in I_2} \log(1 - z z_i^{-1}) + \log \prod_{i \in I_2} (-z_i) + \log \prod_{i \in I_2} z^{-1} \qquad (4)$$

The first two terms can be expanded in power series in z, as described in [1]:

$$\log(1 - z_i a^{-1}) = -\sum_{k=1}^{\infty} \frac{(z_i z^{-1})^k}{k} \qquad |z| > |z_i|, i \in I_1 \qquad (5)$$

$$\log(1 - z z_i^{-1}) = -\sum_{k=1}^{\infty} \frac{(z z_i^{-1})^k}{k} \qquad |z| < |z_i|, i \in I_2 \qquad (6)$$

The coefficient of z^{-k} represents the value of the complex cepstrum at sample number k, apart from the last two terms in (4), which are linear phase contributions to the transform of the complex cepstrum. Define

$$c_{pos}(k) = \begin{cases} -\sum_{i \in I_1} \frac{z_i^k}{k} & k \geq 1 \\ 0 & k \leq 0 \end{cases} \qquad (7)$$

$$c_{neg}(k) = \begin{cases} 0 & k \geq 0 \\ \sum_{i \in I_2} \frac{z_i^k}{k} & k \leq -1 \end{cases} \qquad (8)$$

Notice that this determines c_{pos} and c_{neg} for all k and no aliasing is present. The transform of the complex cepstrum can be written as

$$C(z) = C_{pos}(z) + C_{neg}(z) + C_L(z) \qquad (9)$$

Where C_{pos} and C_{neg} are the z-transforms of c_{pos} and c_{neg}, respectively; and C_L corresponds to the last two terms in (4). C_L can be simplified as follows:

$$\log \prod_{i \in I_2} (-z_i) = \log \prod_{i \in I_2} R_i + \log \prod_{i \in I_2} (-e^{j\theta_i})$$
$$= P + \log(-1)^{n_p} \qquad (10)$$

$$\log \prod_{i \in I_2} z^{-1} = \log z^{-n_2} \qquad (11)$$

where $z_i = R_i e^{j\theta_i}$, n_p = number of positive real $z_i \in I_2$, n_2 = number of $z_i \in I_2$. The transform of the complex cepstrum is therefore

$$\log F(z) = C_{pos}(z) + C_{neg}(z) + P + \log(-1)^{n_p} + \log z^{-n_2} \qquad (12)$$

In practice, the last three terms are kept as adjustment terms and are not included when the complex cepstrum is manipulated. They represent only a constant factor and a time-shift of the original signal.

III. Polynomial Factorization Algorithm

A polynomial factorization program written by W.H. Surber, Jr. [3] was found very effective in this work. It uses the Newton-Raphson method [4] and first searches for a root inside the circle $|z| < 2^{120/n}$, where n is the degree of the polynomial. When no further roots can be found in this region, the roots of the deflated polynomial are inverted and the process repeated. Three normal starting points inside the unit circle are used. If the boundary of the region should be hit, special, pseudorandom starting points are used. The program uses double-precision arithmetic (64 bits) on the IBM 360/91 using FORTRAN H, and factors 255th degree polynomials in about 4.3 seconds with a convergence tolerance of 10^{-7}. Such a polynomial evaluated at the roots generally has a value less than 10^{-10}, except at the one or two roots that appear relatively far outside the unit circle (say $|z| \approx 1.3$). At these points the derivative of the polynomial is so large that a value of 10^{15} can be considered a zero!

IV. Computational Examples

Figures 1-3 pertain to the first example, in which a synthetic speech-like signal was generated by linear filtering of a pulse train by a 10-pole, 2-zero IIR filter. The poles correspond to the vowel /IY/, and there is an additional complex zero pair on the unit circle at 1875 Hz. (at a sampling rate of 15000 Hz). The pulse train consists of ideal pulses 80 samples apart. Fig. 1 shows the original signal sample of 256 points and Fig.

2 shows the locations of the zeros in the complex z-plane. The zero closest to the unit circle is at a radius 1.000279 and an angle of 2943.3 Hz. Fig. 3 shows the phase of the transform in fractions of π, without the linear phase component due to $C_L(z)$.

Figures 4-6 are analogous to Figs. 1-3, but refer to a second example, which uses a sample of natural speech. The sample consists of 256 points of a male utterance of the nasal consonant /m/ and, in contrast to the first example, it is windowed with a Hamming window. The zero closest to the unit circle is at a radius 0.999709 and an angle of 1756.5 Hz.

For the purposes of comparison, the phase unwrapping program of Tribolet [2] was applied to the same examples. In no case tried was the method able to correctly obtain the phase of the signal DFT, even when 2048-point transforms were used. A closer analysis of the results showed that every phase discontinuity obtained was located close to a zero of the signal, both in radius and angle. For the synthetic speech example, eight phase discontinuities were present; their locations were within 1 Hz. of a zero of the signal transform and all but one of these zeros was within 0.001 of the unit circle. Similar results were obtained for the natural speech example. This suggests an inherent difficulty with this method of phase unwrapping because increasing the number of DFT points in order to minimize aliasing of the complex cepstrum makes it more likely to require the evaluation of the phase derivative near a transform zero,where its value is very large.

A more sophisticated approach might be based on a combination of Tribolet's algorithm with a test for the number of zeros of the signal transform lying in a given sector [5]. This test might be applied whenever a change of more than π radians occurs between successive DFT points in order to decide upon grid refinement and phase interpolation. Practically, this amounts to locating the zeros of a polynomial by grid search, an approach much less efficient in general than second order convergent iterative methods. Manual interpolation of the phase generated by Tribolet's program for our examples was successfully performed.

References

1. A.V. Oppenheim and R.W. Schafer, Digital Signal Processing, Prentice-Hall, Englewood Cliffs, N.J., 1975. See Chapter 10.

2. J.M. Tribolet, "A New Phase Unwrapping Algorithm", unpublished manuscript, March 1976, submitted to IEEE Trans. on Acoust. Speech and Signal Processing.

3. W.H. Surber, Jr., "DPROOT", a FORTRAN IV program.

4. See, for example, A. Ralston, A First Course in Numerical Analysis, McGraw-Hill, N.Y., 1965.

5. M. Marden, The Geometry of the Zeros of a Polynomial in a Complex Variable, American Mathematical Society, New York, 1949. See Chapter 9.

Figure Captions

Fig. 1	A synthetic signal of length 256; amplitude vs. sample number: Example 1.
Fig. 2	The transform zero locations in the complex plane: Example 1.
Fig. 3	The phase component of the transform without the linear phase compoment; phase in fractions of π vs. DFT sample number: Example 1.
Figs. 4-6	Same as Figs. 1-3 for Example 2.

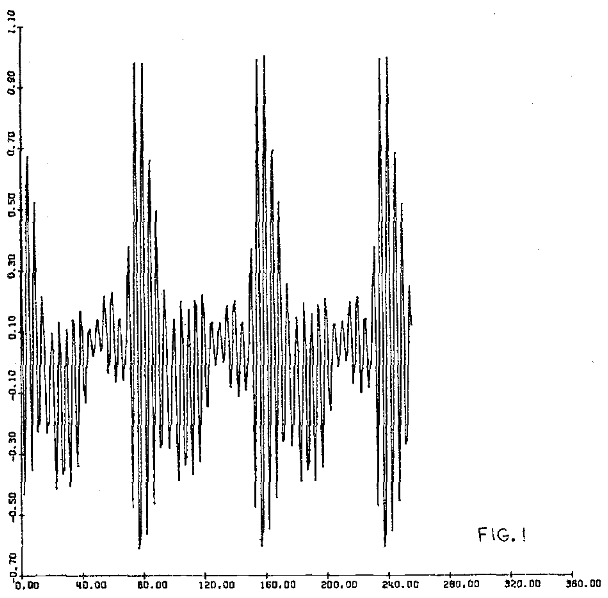

FIG. 1

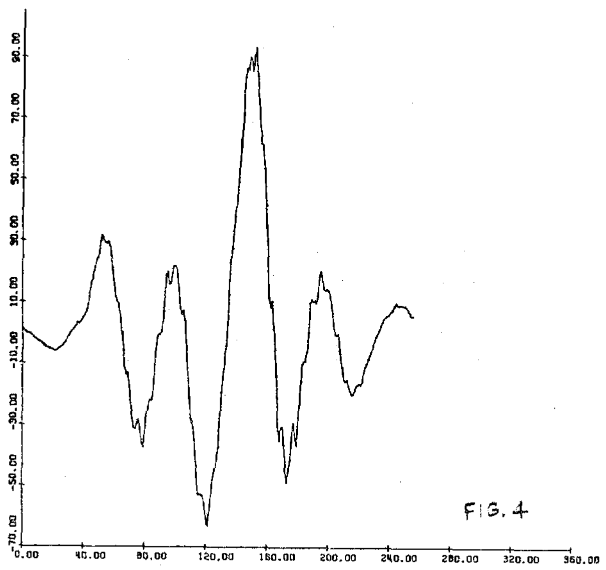

FIG. 4

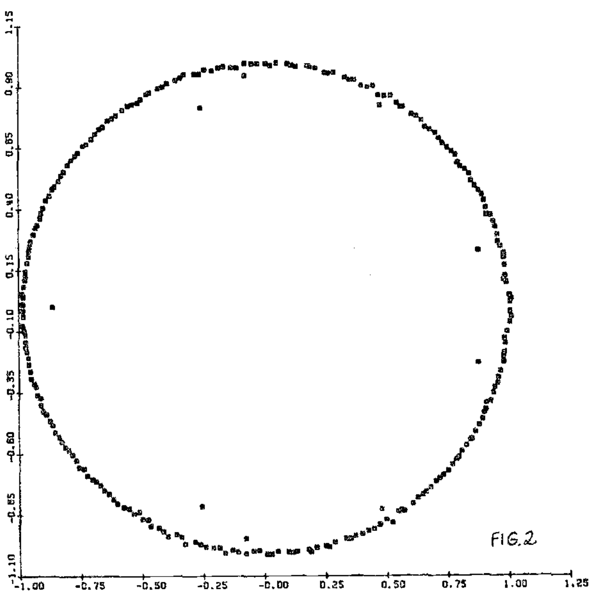

FIG. 2

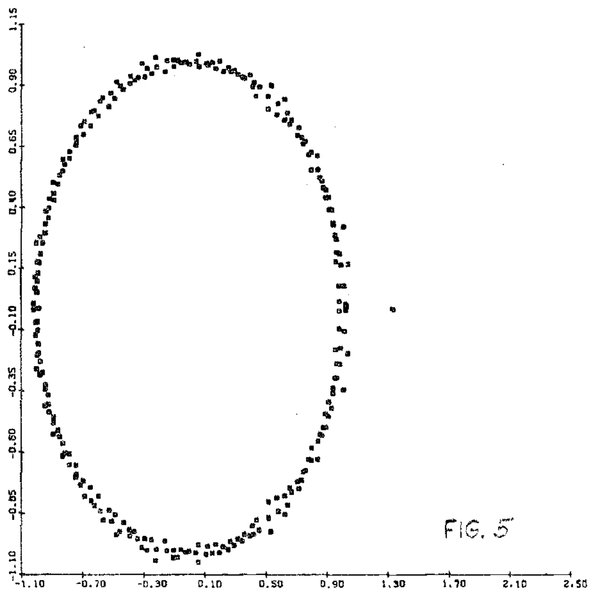

FIG. 5

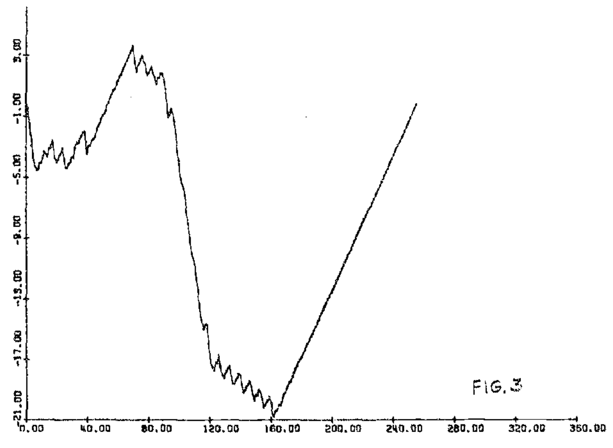

FIG. 3

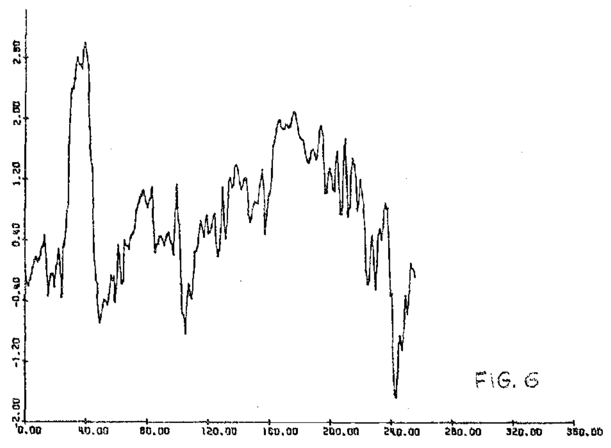

FIG. 6