A Fast Method of Channel Equalisation for Speech Signals and its Implementation on a DSP

Barry Theobald[1], Stephen Cox[1], Gavin Cawley[1] and Ben Milner[2]

[1]School of Information Systems, University of East Anglia, Norwich NR4 7TJ, U.K.

[2]British Telecom Research Laboratories, Martlesham Heath, Ipswich IP5 7RE.

Abstract

Blind equalisation of a speech signal that has been passed over a linear filter can be achieved by estimating the poles of the signal and separating the stationary poles due to the filter from the time-varying poles due to the speech. However, identification of the position of the stationary poles, conventionally done by pole clustering, is unreliable and slow. A new algorithm for identification of stationary poles is presented which is more accurate and faster than clustering.

The problem of recovering a signal passed over a linear time-invariant channel of unknown transfer function (”blind equalisation”) occurs in many areas (for a review, see [1]). One important example is that of recovering a voice signal passed through an unknown filter. This was first studied in [2] and is important in telephony services both for enhancement of speech quality and for use prior to automatic speech recognition. In these situations, it is impracticable to estimate the frequency-response of the channel using a test-signal because equalisation must be immediate. This letter describes an equalisation method based on an algorithm first proposed by Spencer and Rayner [3]. The method relies on clustering poles in the z-plane and uses a technique for pole identification that is faster, more accurate and more robust than conventional methods.

The effectiveness of this technique has enabled us to implement a practical, real-time equaliser on a TMS320C30 DSP system and this is described.

The system model is as follows. The speech signal is conventionally modelled as an all-pole time-varying filter of transfer function *A(z)* (due to the vocal tract) excited by a signal *e(n)* [3]. The speech signal *s(n)* is passed over a channel with a stationary transfer-function *B(z)* (also assumed to be all-pole) to give an output signal *d(n)*. The instantaneous transfer-function of the cascaded system at any time is then *C(z) = A(z)B(z)*. Since *A(z)* and *B(z)* are both all-pole systems, they may be described by

$$A(z) = \frac{G_a}{1-\sum_{k=1}^{P} a_k z^{-k}} = \frac{G_a}{\prod_{k=1}^{P}(1-\alpha_k z^{-k})} \quad \text{and} \quad B(z) = \frac{G_b}{1-\sum_{k=1}^{Q} b_k z^{-k}} = \frac{G_b}{\prod_{k=1}^{Q}(1-\beta_k z^{-k})}$$

so that the cascaded system is described by

$$C(z) = \frac{G_a G_b}{1-\sum_{k=1}^{P+Q} c_k z^{-k}} = \frac{G_a G_b}{\prod_{k=1}^{P+Q}(1-\gamma_k z^{-k})}$$

where γ_k is the k^{th} pole of *C(z)*. The poles of *C(z)* can be estimated using linear prediction [4]. The speech signal *s(n)* is assumed to be stationary over intervals of less than about 32 ms. Hence the signal *d(n)* is blocked into frames of length 32 ms segments and for each frame, the c_k are calculated using the Levinson-Durbin algorithm [5] and the γ_k using a root-finding algorithm. The task of the equaliser is to use these sets of γ_k to design an inverse filter that compensates for the long-term stationary (channel) poles.

Figure 1 plots on the *z*-plane the values of γ_k from a simulation using a fourth-order system as the channel filter. The poles of the channel filter system are located at

0.29±0.92j and 0.64±0.74j. Because of the finite frame length and of windowing, the poles associated with these stationary poles cluster around the true positions rather than lying precisely on these points. However, the poles due to the speech signal are scattered over the plane. Since the actual number of poles present in the signal ($P+Q$) is unknown, it is important to choice a prediction order M that gives a useful set of poles. If M is too low, there is a danger that the transfer function of the channel will not be well-modelled; if too high, the z-plane becomes overcrowded with poles and it becomes difficult to separate stationary channel poles from speech poles.

Identifying the channel poles from the data shown in Figure 1 is not straightforward. We ran several conventional clustering algorithms on data from simulations of speech passed over various channels, and found that all of them performed poorly both in terms of accuracy of identification of the channel poles and speed of convergence. The reasons for this poor performance are as follows. Firstly, the speech poles act as "noise" amongst the channel poles and conventional clustering algorithms are not good at dealing with noise—the clusters they find are very wide and the cluster centroids are often poor estimates of the position of the actual poles. Secondly, the amount of data available for estimating the channel poles increases linearly with time, and we would like to use all the available data to obtain the most accurate estimate of the poles. Unfortunately, since each data point must be re-assigned to a cluster (using a minimum distance criterion) during each iteration, the processing time also increases linearly with time, so that it quickly becomes impossible to run the algorithm in real-time even if channel pole estimation is only performed occasionally. Furthermore, if estimation is to be done over a long period, storage of the complete history of signal poles may become a problem for implementation on a DSP system with limited memory.

Because of these problems with conventional clustering algorithms, we devised an alternative algorithm that is based on direct estimation of the density function of the poles. Firstly, the section of the *z*-plane corresponding to $0.8 < |r| < 1.0$ is quantised into small regions. The centres of these regions are shown in Figure 2 (note that poles in only the top half of the *z*-plane need be processed because of symmetry about the real axis). During signal acquisition, each pole within this band is quantised to a region by separately quantising its magnitude and its angle (we ignore poles whose magnitude < 0.8 as these have negligible effect on the signal). The total number of poles n_i that have fallen inside the *i*'th quantisation region is updated after each frame has been processed as is the fraction $\rho_i = n_i / N$ of the total *N* poles (computed over the entire history of the signal) that are resident in region *i*. The channel pole positions are estimated as the centroids of the quantisation regions whose pole occupation density is above a threshold ρ_T. These high-density regions are easily identified by simply sorting in descending order the array holding the values of ρ_i and thresholding to remove values below ρ_T. In many cases, the candidates for channel pole positions are in adjacent quantisation regions. When this occurs, the algorithm assigns a single channel pole located at the weighted mean (weighted by region occupancy) of these centroid positions

This algorithm offers advantages of both accuracy and speed over conventional clustering techniques. Its accuracy is limited by the error in quantising the *z*-plane rather than by the number of poles that can be processed in the time available for channel pole estimation. Our system used only 150 quantisation regions and we have observed that it is significantly more accurate than *k*-means clustering. It is much faster, because the only processing required to estimate the channel pole positions is a sorting algorithm followed by a thresholding operation. Also, the time taken to implement this algorithm is independent of the number of poles in the channel filter, whereas a conventional clustering algorithm requires more processing time as the number of clusters (i.e. channel poles) increases. Table 1 shows the time taken for each process in the full channel equalisation algorithm when implemented on a TMS320C30 DSP system. Here, the speech sampling-rate was 8 kHz, the frame-rate was 32 ms, $M = 10$, the channel filter was re-estimated every frame and the speech continuously equalised. It can be seen that the full equalisation process was implemented in well under the time for a single frame, and most of the processing time is, in fact, taken with finding the signal poles.

We have established that the technique proposed here produces an excellent method of implementing a real-time equaliser. We are now concentrating on the following ways of developing and improving our system:

1. A current limitation of the system is that it assumes that the channel can be represented as an all-pole filter with simple poles; we aim to extend it to equalise channels that also have zeros, and also to channels with multiple poles.

2. We are implementing an adaptive equaliser that will be useful for tracking changes in the channel transfer function (caused, for instance, by movements of the head when using a mobile telephone).

3. A non-linear quantisation of the *z*-plane should be useful for providing higher estimation accuracy for poles whose magnitude is close to the unit circle.

References

[1] S. Haykin, *Adaptive Filter Theory*, Prentice-Hall, 1991

[2] T.G. Stockham and T.M. Cannon, "*Blind Deconvolution Through Digital Signal Processing*", Proc. IEEE Trans. on Acoustics Speech and Signal Processing, Vol. 63 no. 4, Apr. 1975

[3] P.S. Spencer and P.J.W. Rayner, "*Separation Of Stationary And Time-Varying Systems And Its Application To The Restoration Of Gramophone Recordings*", Proc. IEEE International Symposium on Circuits and Systems, 1989, pages 292–295.

[4] B.S. Atal and L.S. Hanauer, "*Speech Analysis and Synthesis by Linear Prediction of the Speech Wave*", Journal of the Acoustical Society of America, 1971, vol 50, pages 637–655.

[5] L.R. Rabiner and R.W. Schafer, *Digital Processing Of Speech Signals*, Prentice Hall, 1978

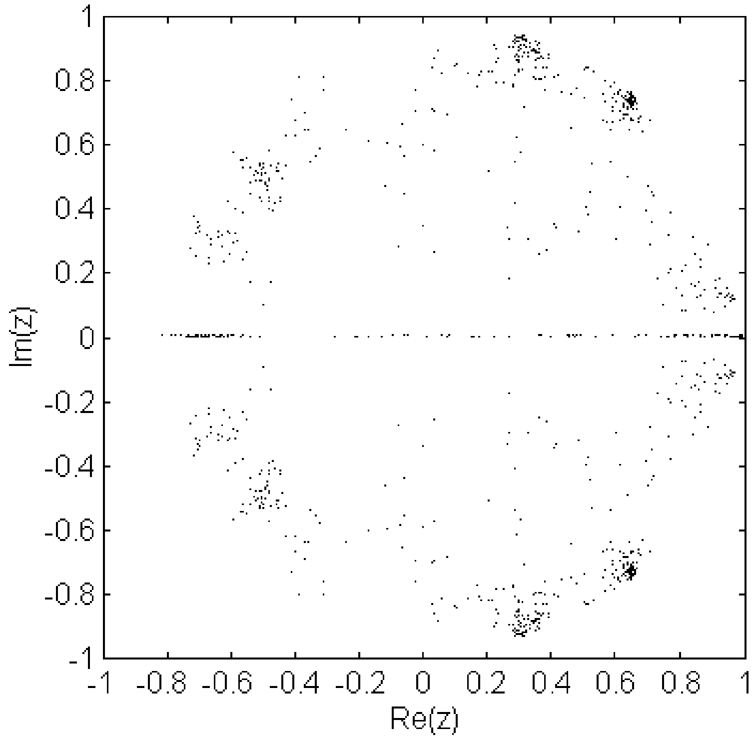

Figure 1: Pole positions for a speech waveform sampled at 16 kHz passed over a fourth order channel. Predictor order $M = 10$.

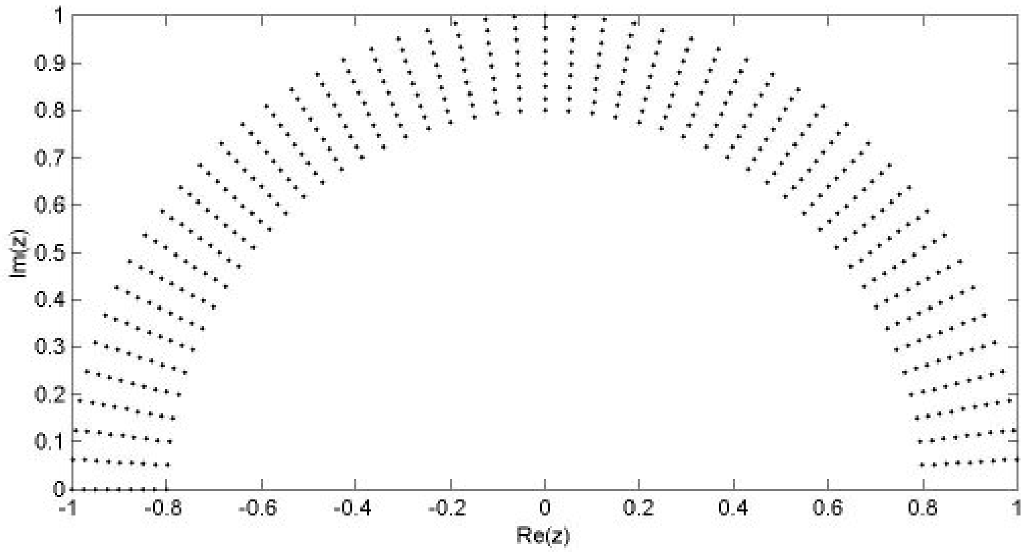

Figure 2: The centres of the regions to which the upper-half of the *z*-plane is quantised.

Routine executed	**Execution time (ms)**
Autocorrelation	1.60
Linear prediction	0.23
Calculating the roots of the predictor	12.0
Quantising the roots	0.10
Sorting the quantisation array	9.72
Cluster analysis	0.0068
Designing the inverse filter	0.0049
Total execution time	**23.7**

Table 1: The time taken to perform each process of the equalisation algorithm