# Implementation of recovery of speech with missing samples on a DSP chip

M. Nafie and F. Marvasti

*Indexing terms: Speech analysis, Signal processing, Digital signal processing*

FFT block codes have been proposed as an error correction code similar to RS codes. In the Letter this novel technique is applied to speech signals with erasures. A convolutional method is also implemented which is capable of reconstructing oversampled speech signals with a certain percentage of lost samples, given that the average sampling rate remains higher than the Nyquist rate. The convolutional method uses FIR digital filters in the time domain as opposed to lowpass filtering the FFT block in the block code. The system is implemented in real time on the floating point DSP processor AT&T DSP32C. The quality of speech for the various techniques is evaluated both subjectively and objectively.

*Introduction:* Speech signals transmitted through imperfect channels are susceptible to erasures. Packets may be lost in a congested network, or samples may be lost due to fading channels. Linear time-invariant systems cannot be used to recover such nonuniformly sampled signals. In this Letter a number of iterative interpolation algorithms are used to recover these signals. A nonlinear and a linear iterative method have been tested. It was noticed that although the nonlinear method converges faster, a smaller number of iterations can be implemented in real time.

*Iterative methods:* A block diagram of the iterative method used is shown in Fig. 1. The linear and the nonlinear modules are shown in Figs. 2 and 3, respectively. This iterative technique can be represented by the following equation [1]:

$$x_{k+1}(t) = PSPx + (I - PS)Px_k(t)$$

where $P$ is the lowpass filtering operator, $S$ is the sampling operator, and $x_k(t)$ is the output signal after $k$ iterations.

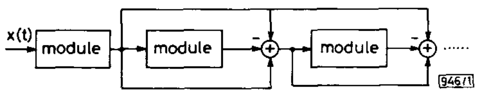


**Fig. 1** *Block diagram of iterative technique*

A number of iterative methods were used. Oversampling the speech signal at a rate higher than the Nyquist leads to the use of convolutional methods in recovery. A convolutional iteration employs convolving the time domain signal with the impulse response of a lowpass filter. The output from each iteration approaches the original signal. The convolutional iteration can be implemented linearly or nonlinearly, with faster convergence in the nonlinear case. Also, block methods can be used. Here, oversampling is achieved by padding zeros in a transform domain. This occurs after the speech samples have been grouped into blocks, hence the name block methods. Different algorithms may differ in the amount of enhancement of the signal they can achieve in one iteration. However, they are also different in the computational complexity and the number of iterations that can be implemented in real time. Hence the best criterion for comparing them is the quality of the output speech they can achieve in real time.

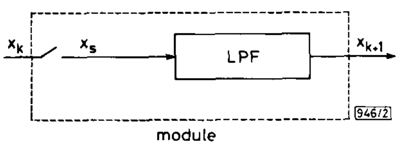


**Fig. 2** *Linear module*

The linear convolutional method was used in [2] as a method to recover signals from their averaged pulsed nonuniform samples. This was extended to impulses in [3]. Each module consists of a sampler followed by a lowpass filter. The sampler keeps the values at the positions of the original samples and inserts zeros where samples are lost. For a proof see [1].

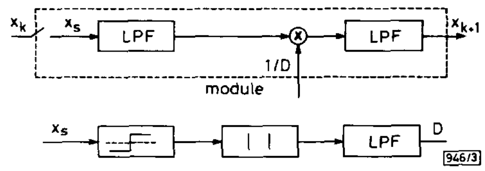


**Fig. 3** *Nonlinear module*

In the block methods the lowpass filtering is implemented by taking the FFT of the received block of samples, inserting zeros in the frequency domain and then obtaining the IFFT. This block method is in fact an error correction code similar to RS codes. Although here we only present its erasure recovery capability it can also be used in recovery from impulsive noise [4]. This method is to be compared to RS codes in typical application fields of RS codes, such as in compact disks, to further explore the error detection and correction capabilities of this new code.

In the nonlinear convolutional method [5] the sampling instances are hardlimited, rectified and then lowpass filtered. The result is then used to divide the lowpass filtered signal.

It is to be noted that perfect recovery can be obtained using the block code if Lagrange interpolation was used in solving for the missing samples [4] but this is very computationally intensive especially for large block sizes.

*Implementation and results:* The different algorithms were implemented in real time on the DSP processor AT&T DSP32C to compare their performance.

Convolutional methods were implemented on a speech signal sampled at 16kHz, and an FIR filter of 31 taps was used for lowpass filtering. The filter was implemented using a Kaiser window. In the block methods, a block of 64 samples which has 32 padded zeros at the end was used. To use a real FFT, which is faster than a complex FFT, 33 zeros were added to 31 samples to form the 64 sample block.

Fig. 4 compares the best results obtained for the block methods and the convolutional methods using the SNR criteria. The SNR was calculated using an overall SNR. To 30% loss, the block method implemented using real FFT is the best; then the hybrid convolutional method performs better. In this hybrid method the initial condition is obtained using the nonlinear method then the remaining iterations are linear.

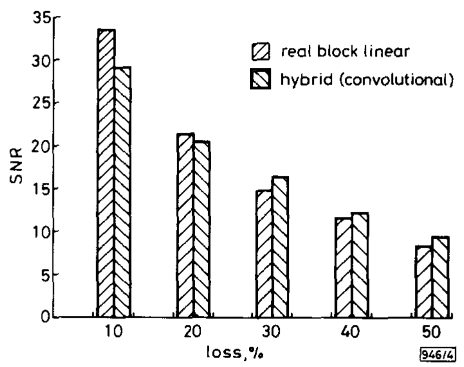


**Fig. 4** *Comparison between SNR of block and convolutional methods*

Several subjects were asked to listen to the speech output from the system and give it a score on a 1 to 5 scale. Fig. 5 shows these results. Again the real block method and the hybrid convolutional method gave the best performance.

*Conclusion:* The capability of iterative methods to recover speech with erasures in real time was demonstrated. Several methods were implemented on the DSP processor DS32C and their performance was compared both subjectively and objectively. The methods were tested at an effective sampling rate of 16kHz to clearly demonstrate their capabilities. These methods can be extended to speech sampled at 8 and 9kHz.

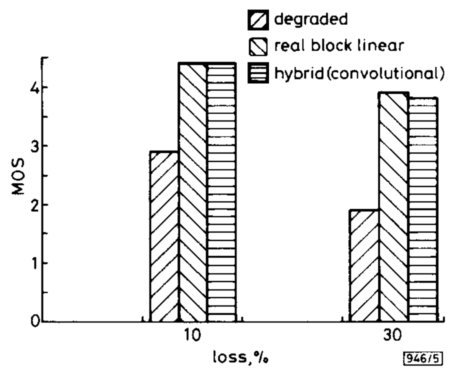


**Fig. 5** *Comparison between MOS of block and convolutional methods*

At twice the Nyquist rate, the MOS test shows that both the real block linear method and the hybrid convolutional method produce the best results at 10 and 30% loss rate. The SNR results show that the real block linear method is the best at 10% sample losses, and that this method and the hybrid convolutional method gave the best results at 20 and 30% losses.


1 November 1993
*Electronics Letters Online No: 19940056*

M. Nafie and F. Marvasti (*Dept. of Electronic and Electrical Engineering King's College London Strand, London WC2R 2LS, United Kingdom*)

**References**

1 MARVASTI, F., ANALOUI, M., and GAMSHADZAHI, M.: 'Recovery of signals using iterative methods', *IEEE Trans. Signal Process.*, 1991, **39**, pp. 872–878
2 WILEY, R.: 'Recovery of bandlimited signals from unequally spaced samples', *IEEE Trans.*, 1978, **COM-26**, pp. 135–137
3 MARVASTI, F.: 'Spectral analysis of random sampling and error free recovery by an iterative method', *Trans. IECE Jpn.*, 1986, **E-69**, (2), pp. 79–82
4 MARVASTI, F., and NAFIE, M.: 'Sampling theorem: A unified outlook on information theory, block and convolutional codes', *Trans. IEICE, Special Section on Information Theory and Its Applications*, September 1993,
5 MARVASTI, F., CLARKSON, P., DOKIC, M., GOENCHANART, U., and LIU, C.: 'Reconstruction of speech signals with lost samples', *IEEE Trans. Signal Process.*, 1992, **40**, pp. 2897–2903

## Product form distributions from decompositions of Markov chains

M.E. Woodward

*Indexing terms: Markov processes, Queueing theory, Telecommunications*

Necessary and sufficient conditions are given for the decomposition of a discrete-time, discrete-state Markov chain into two Markov chains, each having an independent state behaviour. The independence means that the equilibrium probabilities of the original Markov chain can be calculated as a product of those of the Markov chains resulting from the decomposition. This can greatly reduce the computation required to calculate such probabilities. The technique has applications in computer and telecommunications performance modelling and other disciplines involving Markovian models.

*Introduction:* Computer and telecommunication systems lend themselves in a natural way to modelling using the theory of queues and queueing networks [1–4]. The exact analysis of such models usually implies the application of Markovian assumptions and the subsequent calculation of the equilibrium probabilities of the resulting Markov chain. Performance measures for the system can be obtained either directly or indirectly from the equilibrium probabilities. If, however, the state space is extremely large, an exact solution can become intractable due to the vast amount of computation required.

In the following, necessary and sufficient conditions are given for decomposing a model (Markov chain) into a stochastically equivalent realisation by two independent Markov chains, each having a smaller state space than the original. The problem of calculating the equilibrium probabilities can then be reduced to the often much simpler task of calculating the equilibrium probabilities of the Markov chains resulting from the decomposition.

*Terminology and definitions:* In the following any references to a Markov chain will imply a discrete-time, discrete-state Markov chain which is irreducible, aperiodic and has a finite state space represented by a subset of the non-negative integers. The standard terminology relating to partitions will be used (see, for example, [5]).

*Definition 1:* A partition $\pi$ on the state space $S$ of a Markov chain $M$ has the *substitution property* if and only if each state transition associated with the same transition probability maps blocks of $\pi$ into blocks of $\pi$. That is, such transitions from states within a block of $\pi$ are to states all of which are also within a block of $\pi$. Such partitions will be called SP partitions.

*Definition 2:* A trivial partition of the state space $S$ of a Markov chain $M$ is either one that has a single block, or one that has as many blocks as states. The latter will be denoted by a (0). A nontrivial partition then has its obvious meaning.

*Definition 3:* The state behaviour decomposition of a Markov chain $M$ into two Markov chains $M_1$ and $M_2$, each having an independent state behaviour, is a nontrivial decomposition if and only if $M_1$ and $M_2$ each have fewer states than M.

*Decomposition theorem:* A Markov chain $M$ has a nontrivial decomposition of its state behaviour into two Markov chains, $M_1$ and $M_2$, each with an independent state behaviour, if and only if there exist two nontrivial SP partitions $\pi_1$ and $\pi_2$ on the states of $M$ such that $\pi_1 . \pi_2 = \pi(0)$.

In the above, the dot operator generates a new partition whose blocks are all the possible intersections of the blocks of the operands.

The theorem can be proved by making use of the decomposition results for finite state machines in [5]. These results however must first be recast within a stochastic framework by interpreting the finite state machines in [5] as Markov chains. This can be done by associating with each machine input symbol a Markov chain transition probability. For example, a finite state machine with four input symbols can be interpreted as a Markov chain by associating with these inputs the transition probabilities $a$, $b$, $c$ and $d$, where $a + b + c + d = 1$, and some permutation of $a$, $b$, $c$ and $d$ appear in each row of the transition matrix of the Markov chain. Then with these modifications, the proof of the decomposition theorem parallels that of theorem 2.5 in [5]. A direct consequence of this proof is that the Markov chains resulting from the decomposition are irreducible, aperiodic and recurrent non-null and so have a unique equilibrium distribution.

To clarify the above idea of interpreting finite state machines as Markov chains, consider a finite state machine with input set {0,1} and state set {0, 1, 2, 3, 4, 5}, and a state table as in Table 1.

**Table 1:** State table of finite state machine

| | 0 | 1 |
|---|---|---|
| 0 | 3 | 2 |
| 1 | 5 | 2 |
| 2 | 4 | 1 |
| 3 | 1 | 4 |
| 4 | 0 | 3 |
| 5 | 2 | 3 |