IMPROVING THE STABILITY OF THE DFT ERROR RECOVERY CODES BY USING THE VANDERMONDE FAST DECODING ALGORITHM

Abdolali Momenai, Siamak Talebi, Member, IEEE

University of Kerman, Kerman, Iran
momenai@gmail.com
siamak.talebi@mail.uk.ac.ir

ABSTRACT

Discrete Fourier Transform (DFT) error recovery codes have been extended from the Galois field of numbers to the real and complex fields of numbers. The arithmetic in the complex field is much easier compared to the Galois field. But the computational error always exists in every calculation that is performed on the real and complex fields of numbers. This paper proposes a new algorithm for decoding DFT error recovery codes on the real and complex fields of numbers. It is shown that the proposed algorithm has lower computational complexity compared to the other DFT decoding algorithms. The proposed method is also more stable than the current methods of decoding DFT codes.

1. INTRODUCTION

Error recovery techniques have been extended from the binary to the real and complex fields of numbers [1], [2]. Parts of these recovery techniques use the Discrete Fourier Transform (DFT) on the complex field of numbers [2]. One of their advantages to the binary recovery method is that they can be easily implemented on DSP cards. Also they perform well under sparse error condition. But their stability falls when there are a large number of consecutive errors [2]-[6]. This paper proposes a new algorithm for decoding DFT codes on the real and complex fields of number. This new method accelerates the decoding procedure by exploiting the Vandermonde properties of the recovery matrix decomposition. It is also shown that this method is more stable than the current methods for decoding DFT codes.

This paper is organized as follows. Section 2 describes the DFT codes. The proposed method is described in Section 3. The computational complexity of the proposed algorithm is studied in Section 4 with the experimental results in Section 5

2. DFT CODES

Assume that there is an information source that produces x_i for $0 \le i < m$ which is denoted by the vector $x_{m\times 1}$. The DFT of x is denoted by $\hat{x} = F_m x$ in which F_m is the Fourier transform matrix of size m.

$$j = \sqrt{-1}\ ,\ \omega_m = exp(\ -j\frac{2\pi}{m})$$

$$F_m = [\,F_{m_{i,k}}\,] = [\,\omega_m^{ik}\,]\ ,\ 0 \le i,k < m \qquad (1)$$

Now $\hat{x}_{m\times 1}$ is oversampled to form $\hat{y}_{n\times 1}$ by padding $Z = n - m$ consecutive zeroes in the middle of $\hat{x}_{m\times 1}$ in a way that the Hermitian symmetry is preserved. The position of these inserted zeroes is denoted by z_i. Because we have Z consecutive zeroes starting at index α of the $\hat{y}_{n\times 1}$, z_i can be shown by

$$z_i = (\alpha + i)\,mod\ n\ ,\ 0 \le i < Z\ . \qquad (2)$$

The *mod n* shows that there can be circularly consecutive zeroes in $\hat{y}_{n\times 1}$. A more general form of the zero positions is

$$z_i = (\alpha + \beta i)\,mod\ n \quad 0 \le i < Z\ . \qquad (3)$$

In this formula, α indicates the position of the first inserted zero and β is the circular distance between any two consecutive zeroes. Equation (3) reduces to (2) if the circular distance is 1, $\beta = 1$.

The vector $y_{n\times 1}$ is the inverse Fourier transform of $\hat{y}_{n\times 1}$ and it is transmitted through the channel. At the other side of the channel, $y'_{n\times 1}$ is received which is $y_{n\times 1}$ with L errors. It is assumed that the position of the errors is known by the receiver. It does not affect the generality of the algorithm, because an error locating algorithm can be performed first on the received signal to find the position of the errors as it was studied in the previous works [1], [8]. The position of the errors is denoted by l_k for $0 \le k < L$. The error samples at the position l_k in $y'_{n\times 1}$ are set to zero and the error signal, e is defined as follows

$$y = y' + e\ ,\ y'_i = \begin{cases} 0 & i = l_k \\ y_i & i \ne l_k \end{cases} \Rightarrow e_i = \begin{cases} y_i & i = l_k \\ 0 & i \ne l_k \end{cases}. \qquad (4)$$

Because there are Z padded zeroes in the positions z_i of

$\hat{y}_{n\times 1}$ it follows

$$\hat{y} = \hat{y}' + \hat{e} \Rightarrow \hat{e} = \hat{y} - \hat{y}' , \hat{y}_{z_i} = 0 \Rightarrow \hat{e}_{z_i} = -\hat{y}'_{z_i} . \tag{5}$$

Furthermore it is known that $\hat{e} = F_n e$, so

$$\hat{e} = F_n e \Rightarrow \hat{e}_i = \sum_{k=0}^{n-1} \omega_n^{ik} e_k \overset{(4)}{\Rightarrow} \hat{e}_i = \sum_{k=0}^{L-1} \omega_n^{il_k} y_{l_k} . \tag{6}$$

Equation (6) can be written at the inserted zero positions, i.e. $i = z_i$, to obtain

$$\hat{e}_{z_i} = \sum_{k=0}^{L-1} \omega_n^{z_i l_k} y_{l_k} \overset{(5)}{\Rightarrow} -\hat{y}'_{z_i} = \sum_{k=0}^{L-1} \omega_n^{z_i l_k} y_{l_k} . \tag{7}$$

Equation (7) is a linear equation between the lost samples and the values of $\hat{y}'$ at the inserted zero positions. It can be rewritten in the matrix product form as

$$S = [S_i] = [\hat{e}_{z_i}] = [-\hat{y}'_{z_i}] \ , \ 0 \le i < Z \tag{8}$$

$$P = [P_k] = [y_{l_k}] \ , \ 0 \le k < L \tag{9}$$

$$F_R = [F_{n_{z_i . l_k}}] = [\omega_n^{z_i l_k}] \ , \ 0 \le i < Z \ , \ 0 \le k < L \tag{10}$$

$$S = F_R P . \tag{11}$$

S contains the syndromes (value of $\hat{y}'$ at the inserted zero positions), P denotes the lost samples, and F_R is a $Z \times L$ matrix that shows the coefficient of the linear equation (7). Equation (11) is an overdetermined system of equations. It was also proved that F_R has full column rank L . So there is a unique least square solution that is

$$P = (F_R^T F_R)^{-1} F_R^T S . \tag{12}$$

Therefore the linear system of (11) can be solved and the error signal can be found and substituted in (4) to compute y. Finally the original signal $x_{m\times 1}$ is calculated by removing the inserted zeroes from $\hat{y}_{n\times 1}$ and performing an inverse DFT.

The error in the computation of this solution is in direct proportion with $\kappa(F_R)$, [11], which is the condition number of F_R. References [2]-[6] have studied the effects of the different patterns of errors (different values of l_k) on the stability of the DFT codes. They have proved that the DFT codes are vulnerable to the burst of errors which is the consecutive form of l_k. This case results in a large condition number for F_R which makes the computation of the solution for (7) very unstable. Next section proposes a fast decoding method which reduces this unstability problem.

Solving (11) by using matrix operations needs high computational complexity. The matrix algorithms needs about L^3 operations [12] which is not applicable if the number of the lost samples is high. So there have been extensive studies to find better algorithms for solving (11). Some of the efficient algorithms like recursive extension [2] and Forney algorithm [7] solve (14) by introducing some auxiliary polynomials. The computational complexity of these algorithms is greatly reduced by using these auxiliary polynomials. This makes them efficient for decoding long DFT codes with high number of errors when the matrix inversion is not applicable. The next section proposes a matrix based method that needs lower computational complexity than the currently used methods.

3. DFT Decoding Using Matrix Decomposition

There are many different methods in the linear algebra that can accelerate the computation of the solution for a linear system of equations [12]. Most of these methods are developed for specially structured matrices like Toeplitz, Circular and Vandermonde matrices. These methods can still be used when the matrix of the coefficients of the linear system can be decomposed into the product of some structured matrices. Equation (10) shows the system of the linear equations in the case of the DFT decoding. Equations (7) and (10) show that F_R is a submatrix of F_n, i.e. F_R contains the rows z_i and the columns l_k of the matrix F_n. By considering

$$\omega_n^{\beta . l_k} = \varphi_{l_k} \tag{13}$$

$$\omega_n^{z_i l_k} = \omega_n^{[(\alpha+\beta i) \bmod n] l_k} = \omega_n^{\alpha l_k + \beta i l_k} = \omega_n^{\alpha l_k} (\varphi_{l_k})^i \tag{14}$$

and substituting (14) in (10), F_R can be rewritten as

$$F_R = [F_{n_{z_i . l_k}}] = [\omega_n^{\alpha l_k} (\varphi_{l_k})^i] . \tag{15}$$

Therefore F_R can be decomposed into the product of a Vandermonde matrix F_V and a diagonal matrix F_D as

$$F_R = F_V F_D \tag{16}$$

$$F_V = [F_{V_{i,k}}] = [(\varphi_{l_k})^i] \ , \ 0 \le i < Z \ , \ 0 \le k < L \tag{17}$$

$$F_D = Diag(\omega_n^{\alpha l_k}) \ , \ 0 \le k < L \tag{18}$$

$$F_V = \begin{bmatrix} \varphi_{l_0}^0 & \varphi_{l_1}^0 & \cdots & \varphi_{l_{L-1}}^0 \\ \varphi_{l_0}^1 & \varphi_{l_1}^1 & \cdots & \varphi_{l_{L-1}}^1 \\ \vdots & \vdots & \ddots & \vdots \\ \varphi_{l_0}^{Z-1} & \varphi_{l_1}^{Z-1} & \cdots & \varphi_{l_{L-1}}^{Z-1} \end{bmatrix}_{Z\times L} \quad F_D = \begin{bmatrix} \omega_n^{\alpha l_0} & 0 & \cdots & 0 \\ 0 & \omega_n^{\alpha l_1} & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & \omega_n^{\alpha l_{L-1}} \end{bmatrix}_{L\times L}$$

F_D is an $L \times L$ diagonal matrix that has nonzero diagonal entries and F_V is a $Z \times L$ Vandermonde matrix. The main point of the above paragraph is that by substituting (16) in (11) it follows

$$S = F_V F_D P . \tag{19}$$

Equation (19) is a linear system of equations that is the product of two very structured matrices, a Vandermonde matrix and a diagonal matrix. If we define

$$\overline{P} = F_D P \tag{20}$$

and substitute (20) in (19) it follows

$$S = F_V \overline{P} . \tag{21}$$

Equation (21) is a Vandermonde system of equations. There are some fast algorithms for solving a Vandermonde system [12]. These algorithms are almost extensions and improvements of Bjorck and Pereyra algorithm [9]. Fig. 1 shows and implementation of the Bjorck and Pereyra algorithm for solving the Vandermonde system (21).

Higham [10] has extensively analyzed the computational error and stability of these Vandermonde system algorithms. He has proved that these algorithms give surprisingly accurate

```
for $i = 0 : L-1$
    $\overline{P}_i = S'_i$
end
for $k = 0 : L-2$
    for $i = L-1 : -1 : k+1$
        $\overline{P}_i = \overline{P}_i - \varphi_{l_k} \overline{P}_{i-1}$
    end
end
for $k = L-2 : -1 : 0$
    for $i = k+1 : L-1$
        $\overline{P}_i = \overline{P}_i / (\varphi_{l_i} - \varphi_{l_{i-k-1}})$
    end
    for $i = k : L-2$
        $\overline{P}_i = \overline{P}_i - \overline{P}_{i+1}$
    end
end
```

Fig. 1. Bjorck and Pereyra algorithm for computing the solution of the Vandermonde system.

results which improve the stability of the system. The experimental results in the next section also show that the stability of the proposed DFT error recovery algorithm is improved by using these Vandermonde system algorithms.

After computing $\overline{P}$ in (21) by using the algorithm of Fig. 1, equation (20) should be solved to find P that is the vector of the lost samples. It should be considered that

$$P = F_D^{-1}\overline{P} \tag{22}$$

$$F_D^{-1} = Diag(1/\omega_n^{\alpha l_k}) = Diag(\omega_n^{-\alpha l_k}) \ , \ 0 \le k < L \tag{23}$$

$$P_k = \omega_n^{-\alpha l_k}\overline{P}_k = exp(+j\frac{2\pi\alpha l_k}{n})\overline{P}_k \ , \ 0 \le k < L \ . \tag{24}$$

Parameter α determines the first position of the inserted zeroes in $\hat{y}_{n\times 1}$ that is the Fourier transform of $y_{n\times 1}$. So by changing α the position of the inserted zeroes in the Fourier domain of $y_{n\times 1}$ is shifted. Shifting α terms in the Fourier domain and multiplying the time domain by $\omega_n^{-\alpha i}$ is one of the properties of the Fourier transform [13]. So the effect of this shift in the Fourier domain of $y_{n\times 1}$ can be seen in (24) when the vector $\overline{P}$ is multiplied by $\omega_n^{-\alpha l_k}$ to find P that is the vector of the lost samples of $y_{n\times 1}$. Then $\hat{y}_{n\times 1}$ that is the Fourier transform of $y_{n\times 1}$ should be calculated. The inserted zeroes at the positions z_i should be removed to find $\hat{x}_{m\times 1}$. Finally another inverse DFT is needed to find the original data $x_{m\times 1}$ which completes the recovery process.

4. Computational Complexity

This section analyses the computational complexity of the proposed algorithm and compares it with the recursive extension [2] and Forney algorithm [7] which are good decoding methods of the DFT codes. The computational

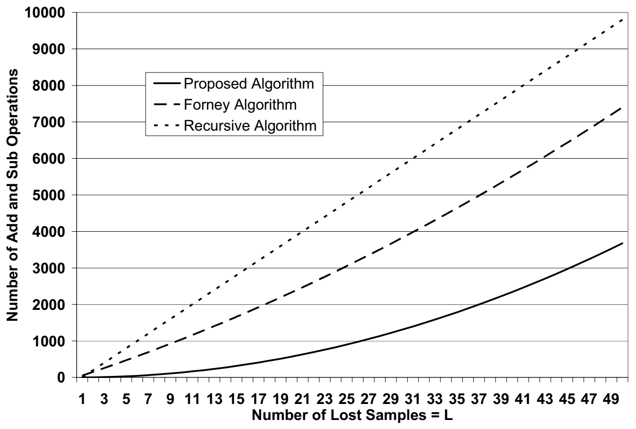

Fig. 2. Number of addition and subtraction operations for the algorithms.

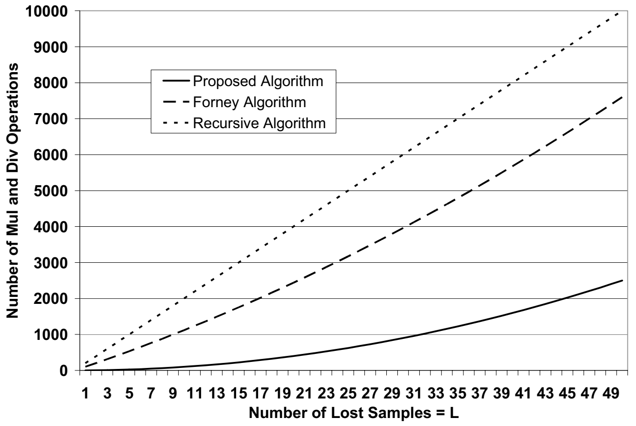

Fig. 3. Number of multiplication and division operations for the algorithms.

complexity for the calculation of $\omega_n^{\alpha l_k}$ and $\omega_n^{-\alpha l_k}$ is not considered since all of these algorithms share this computation.

The proposed decoding algorithm by using the algorithm of Fig. 1 needs $3L.(L-1)/2$ subtractions and $L.(L-1)$ multiplications and divisions for the computation of $\overline{P}$. If $\alpha = 0$ then $P = \overline{P}$ and the lost samples are found. But if $\alpha \ne 0$, then another L multiplications are needed to find the vector of the lost samples P from $\overline{P}$.

The recursive extension algorithm and the Forney algorithm use the error locator polynomial $\Lambda(x)$ to accelerate the decoding procedure. Because it was assumed that an error locating algorithm is performed on the received signal to find the position of the errors, this polynomial is already calculated in the error locating step. The recursive extension algorithm [2] needs NL multiplications and $N(L-1)$ additions to calculate the Fourier of the error signal by using the recursive formula. Then it is substituted in (4) to find the original signal.

According to [1], [7], the Forney algorithm needs $Z.(L-1)$ additions and $Z.L$ multiplications for the calculation of the error evaluator $\Gamma(x)$ polynomial and $L-1$ multiplications to compute $\Lambda'(x)$ from the error locator polynomial $\Lambda(x)$.

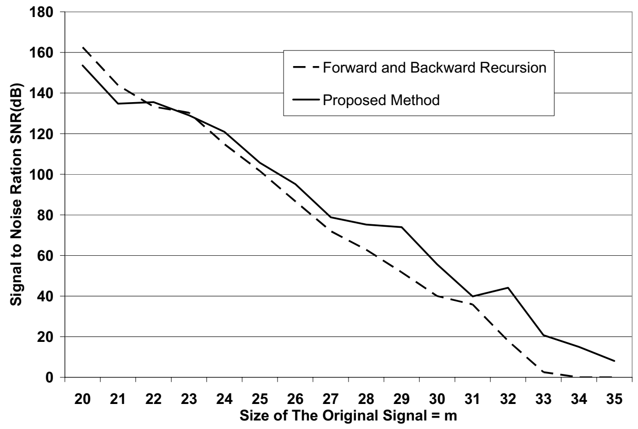

Fig. 4. Signal to Noise Ratio, SNR (dB) of the proposed method and the recursive method.

Then by using these polynomials, another *L.(Z+L-1)* additions and *L.(Z+L+1)* multiplications and divisions are needed to find the values of the lost samples.

Fig. 2 shows the number of addition and subtraction operations of these algorithms when *N=200*, *Z=N/4=50* and the number of the lost samples *L* is increased from *1* to *50*. Fig. 3 shows the number of multiplication division operations in the same conditions of Fig. 1. It can be seen that the proposed method needs much less computations than the other algorithms.

5. Experimental Results

This section studies the stability and the accuracy of the proposed method compared with the recursive extension algorithm. Marvasti *et al.* [2] have compared the stability of the recursive extension algorithm with some other methods like Forney algorithm. They have shown that the recursive extension is better than the other algorithms in the case of the stability and the accuracy of the decoding procedure. Therefore we only compare the stability of the proposed method with the recursive extension method.

A normalized random number generator with uniform distribution generated the signal vector $x_{m\times 1}$. The DFT of x was oversampled with Z consecutive zeroes to make $y_{n\times 1}$. The size of the original signal, *m*, was increased from *1* to *40* and the number of the inserted zeroes was *Z=m+1* in each step. Then *L=Z* consecutive samples of $y_{n\times 1}$ were erased. The proposed algorithm and the recursive extension algorithm were used to recover the original signal. Then the Signal to Noise Ratio (SNR) of the recovered signals were calculated. When the size of the original signal was below *20*, these algorithms recovered the original signal with very high accuracy and they didn't have significant difference in their SNR values. When the size of the original signal was above *35* both algorithms failed to recover the original signal. The main difference of these algorithms was when the size of the original signal was between *20* and *35*. Fig. 4 compares the SNR(dB) values of the recovered signal when the size of the original signal, *m*, was between 20 to 35. It should be noted that the SNR values in this paper are the average of the SNR values for 100 recovered signals. As the figure indicates, the proposed method recovers the original signal with up to 20 dB higher accuracy compared to the recursive method. This is mainly because the proposed algorithm is based on the fast and accurate Bjorck-Pereyra [9] Vandermonde system algorithm. Therefore the experimental result agrees with the theoretical result that the proposed method by using the Vandermonde system algorithm is more stable than the recursive extension algorithm.

6. References

[1] R. E. Blahut, *Algebraic Codes for Data Transmission*. Cambridge: Cambridge University Press, 2003.

[2] F. Marvasti, M. Hasan, M. Echhart and S. Talebi, “Efficient algorithms for burst error recoery using FFT and other transform kernels,” *IEEE Trans. Signal Processing*, vol 47, pp. 1065-1075, Apr 1999.

[3] G. Rath and C. Guillemot, “Frame-theoretic analysis of DFT codes with erasures,” *IEEE Trans. Signal Processing*, vol. 52, pp. 447-460, Feb 2004.

[4] G. Rath and C. Guillemot, “Subspace algorithms for error localization with quantized DFT codes,” *IEEE Trans Communication*, vol. 52, pp. 2115-2124, Dec. 2004.

[5] P. J. S. G. Ferreira, “The Stability of a procedure for the recovery of lost samples in band-limited signals,” *Signal Processing*, vol. 40, pp. 195-205, Dec. 1994.

[6] P. J. S. G. Ferreira, “The eigenvalues of matrices which occur in certain interpolation problems,” *IEEE Trans. Signal Processing*, vol 45, pp. 2115-2120, Aug. 1997.

[7] G. D. Forney, Jr., “On decoding BCH codes,” *IEEE Trans. Inform. Theory*, vol. IT-11, pp. 549-557, 1965.

[8] P. J. S. G. Ferreira and J. M. N. Vieira, “Locating and correcting errors in images,” *in Proc ICIP*, vol. I, Santa Barbara, CA, Oct. 1997, pp. 691-694.

[9] A. Bjorck and V. Pereyra, “Solution of Vandermonde systems of equations,” *Mathematical Computation.*, vol. 24, pp. 893-903, 1970.

[10] N.J Higham, “Error analysis of the Bjorck-Pereyra algorithms for solving Vandermonde systems,” *Numer. Math.*, vol. 50, pp. 613-632., 1987.

[11] R. A. Horn and C. R. Johnson, *Matrix Analysis*. Cambridge: Cambridge University Press, 1990.

[12] G. H. Golub and C. F. Van Loan, *Matrix Computations*. 3rd ed., London: The Johns Hopkins University Press, 1996.

[13] A. V. Oppenheim and R. W. Schafer, *Discrete-Time Signal Processing*. Englewood Cliffs, NJ: Prentice-Hall, 1989.