Correspondence

On Computing the Inverse DFT

P. DUHAMEL, B. PIRON, AND J. M. ETCHETO

***Abstract*—This correspondence indicates a (possibly) new method for computing an inverse discrete Fourier transform (IDFT) through the use of a forward DFT program.**

We point out that, in many cases, this is obtained without any additional cost, either in terms of program length or in terms of computational time.

I. Forward and Inverse DFT [1]–[3]

The forward DFT is usually defined as follows:

$$X_k = \sum_{n=0}^{N-1} x_n W_N^{nk} = \text{DFT}_k\{x_n\} \tag{1}$$

with

$$W_N \triangleq \exp\left(-j\frac{2\pi}{N}\right)$$

and the inverse DFT has nearly the same expression, except for a normalizing factor $1/N$ and a conjugation of the Nth root of unity:

$$x_n = \frac{1}{N}\sum_{k=0}^{N-1} X_k W_N^{-nk}. \tag{2}$$

As we are very often interested only in relative values of the sequence, the scaling factor is generally omitted:

$$x_n' = \sum_{k=0}^{N-1} X_k W_N^{-nk} = \text{IDFT}_n\{X_k\} \tag{3}$$

or sometimes distributed between the two expressions to increase their symmetry:

$$\begin{cases} X_k' = \dfrac{1}{\sqrt{N}}\displaystyle\sum_{n=0}^{N-1} x_n W_N^{nk} \\ x_n = \dfrac{1}{\sqrt{N}}\displaystyle\sum_{k=0}^{N-1} X_k' W_N^{-nk}. \end{cases} \tag{4}$$

This correspondence first recalls the usual algorithms used to compute the IDFT, and then proposes a new one, which turns out to have some advantages compared to the classical ones.

II. Usual Algorithms for Computing the IDFT [1]–[3]

Since both expressions (1) and (3) for the forward and inverse DFT are nearly symmetrical, a trivial solution for computing the IDFT is to conjugate every multiplying constant (including multiplications by j in radix-4 or split-radix programs [3], [4]) inside the forward FFT algorithm. This takes into account the conjugation of W_N in (3). A multiplication by $1/N$ may be needed to obtain (2) if necessary.

But this solution requires two different subprograms for computing the forward and inverse DFT. Therefore, when the scaling factor $1/N$ is omitted or when (4) is used, it is often preferred to use the forward DFT as an intermediate step.

Manuscript received November 1, 1986; revised August 24, 1987.

The authors are with CNET/PAB/RPE, 92131 Issy-les-Moulineaux, France.

IEEE Log Number 8718023.

In fact, it is easily seen that

$$\text{IDFT}_n\{X_k\} = \sum_{k=0}^{N-1} X_k W_N^{-nk} = \text{DFT}_{-n}\{X_k\} \tag{5}$$

which means that the unscaled inverse DFT of $\{X_k\}$ is equivalent to a forward DFT followed by a permutation:

$$x_n' = x''_{\langle -n \rangle_N}. \tag{6}$$

This permutation has to be performed after FFT computation, which is time consuming or has to be included in the subsequent calculation, which gives rather intricate programs.

III. The New Algorithm

The new method is based on the remark that conjugating x_n' in (3) gives an expression which is already in the form of a DFT:

$$x_n'^* = \sum_{k=0}^{N-1} X_k^* W_N^{nk}. \tag{7}$$

A second remark is that

$$x_n = a_n + j \cdot b_n \Rightarrow j \cdot x_n^* = b_n + j \cdot a_n. \tag{8}$$

Hence, if we multiply (7) by j, we get

$$j \cdot x_n'^* = \sum_{k=0}^{N-1} j \cdot X_k^* W_N^{nk} \tag{9}$$

and

$$x_n' = j\left[\sum_{k=0}^{N-1} (jX_k^*)\, W_N^{nk}\right]^*. \tag{10}$$

In other words, (10) means that the unscaled IDFT of a sequence can be obtained by the following procedure:

- exchange the real and imaginary parts of the initial sequence,
- perform a forward DFT,
- exchange the real and imaginary parts of the result.

At first glance, the above procedure seems to be more complicated than the usual one, but we shall see that in many circumstances, this procedure is absolutely costless when compared to a forward FFT.

In fact, FFT programs are generally written in real rather than complex arithmetic (to increase execution speed). Hence, if written in Fortran, FFT subroutines are generally of the type

SUBROUTINE FFT (*XR*, *XI*, *N*)

where *XR* (respectively, *XI*) is the real (respectively, imaginary) part of the sequence to be transformed on input, and of the result on output.

In such cases, an unscaled IDFT can be obtained by exchanging the real and imaginary parts in the call of the subroutine.

If the IDFT of the sequence $YR(I) + j \cdot YI(I)$ is to be computed, (10) tells us that it will be obtained in the following manner:

CALL FFT (*YI*, *YR*, *N*)

which will perform the three steps described above.

It is easily seen that the same kind of procedure will provide the desired IDFT in any of the programming languages for which subprogram's arguments are transmitted by means of their address.

Let us also point out that, obviously, this method also holds for the more symmetrical definition of the DFT and IDFT given in (4) as well as for multidimensional DFT's and IDFT's.

References

[1] L. R. Rabiner and B. Gold, *Theory and Application of Digital Signal Processing.* Englewood Cliffs, NJ: Prentice-Hall, 1975.
[2] E. O. Brigham, *The Fast Fourier Transform.* Englewood Cliffs, NJ: Prentice-Hall, 1974.
[3] C. S. Burrus and T. W. Parks, *DFT/FFT and Convolution Algorithms.* New York: Wiley, 1985.
[4] P. Duhamel, "Implementation of split-radix FFT algorithms for complex, real, and real-symmetric data," *IEEE Trans. Acoust., Speech, Signal Processing*, vol. ASSP-34, pp. 285-295, Apr. 1986.

An Adaptive IIR Algorithm with Unimodal Performance Surfaces

SHLOMO KARNI AND GENGSHENG ZENG

Abstract—**In this correspondence, we develop an adaptive IIR algorithm by means of least squares inverses; the algorithm also guarantees the resulting IIR system to be stable.**

I. Introduction

Current methods in adaptive system identification can be roughly divided into two categories: adaptive FIR (finite impulse response) systems and adaptive IIR (infinite impulse response) systems. If an IIR system is to be identified by an adaptive FIR system, the size of the FIR system is usually fairly large, even though the adaptive procedure converges faithfully. However, the current adaptive IIR systems have two disadvantages that are not found in the adaptive FIR systems [1]. 1) They become unstable if the poles move outside the unit circle during the adaptive process. 2) Their performance surfaces are generally nonquadratic and may even have local minima.

In 1982, Chui and Chan [2] presented an IIR filter design technique by approximation theory methods. They showed that if $H(z) \in H^2$ where H^2 is the usual Hardy-Hilbert space of analytic functions in $|z| < 1$ and $H(0) \neq 0$, then there exists a unique $Q_N \in \pi_N$ solving the extremal problem

$$\|1 - Q_N(z)\,H(z)\|_2 = \inf\left\{\|1 - A(z)\,H(z)\|_2 : A \in \pi_N\right\} \tag{1}$$

where π_N denotes the collection of all polynomials of degree not exceeding N. The polynomial Q_N is called a least squares inverse of $H(z)$ from π_N, and has the important property

$$Q_N(z) \neq 0 \quad \text{for } |z| < 1.$$

That is, we have a *stable* all-pole filter

$$H_A(z) = \frac{1}{Q_N(z)}$$

approximating $H(z)$.

Based on Chui and Chan's results, we derive the adaptive IIR algorithm which overcomes the two mentioned disadvantages of the current adaptive IIR algorithm. In fact, the new algorithm presented here is of quadratic performance surfaces and results in a stable IIR system. We first give an adaptive algorithm for the all-pole system, and the IIR algorithm follows immediately.

Manuscript received April 30, 1987; revised September 22, 1987.
The authors are with the Department of Electrical and Computer Engineering, University of New Mexico, Albuquerque, NM 87131.
IEEE Log Number 8718439.

II. The Adaptive All-Pole System

Let

$$Q_N(z) = q_0 + q_1 z^{-1} + \cdots + q_N z^{-N}$$

and

$$H(z) = \sum_{k=0}^{\infty} h_k z^{-k}.$$

Chui and Chan showed that $Q_N(z)$ in (1) can be obtained by solving the following linear system of equations:

$$\begin{bmatrix} r_0 & r_1 & \cdots & r_N \\ r_1 & r_0 & \cdots & r_{N-1} \\ \vdots & \vdots & \vdots & \vdots \\ r_N & r_{N-1} & \cdots & r_0 \end{bmatrix} \begin{bmatrix} q_0 \\ q_1 \\ \vdots \\ q_N \end{bmatrix} = \begin{bmatrix} h_0 \\ 0 \\ \vdots \\ 0 \end{bmatrix} \tag{2}$$

where $\{r_k\}$ is the serial correlation sequence of $\{h_k\}$:

$$r_k = \sum_{n=0}^{\infty} h_{k+n} h_n.$$

We now transform (2) into an adaptive algorithm.

Suppose the system $H(z)$ is fed with the white noise $\{x_n\}$ of variance σ^2, and we denote the output as $\{d_n\}$. It is well known [3] that the autocorrelation sequence of $\{d_n\}$ is given by

$$\{\sigma^2 r_k\}$$

and

$$\sigma^2 h_0 = E[x_n d_n].$$

Hence, we can rewrite (2) as follows:

$$E\{D_N D_N^T\}\,Q_N = \begin{bmatrix} E\{x_n d_n\} \\ 0 \\ \vdots \\ 0 \end{bmatrix} \tag{3}$$

where

$$X_N = [x_n, x_{n-1}, \cdots, x_{n-N}]^T$$
$$D_N = [d_n, d_{n-1}, \cdots, d_{n-N}]^T$$
$$Q_N = [q_0, q_1, \cdots, q_N]^T.$$

We now introduce an auxiliary quadratic function

$$F(Q_N) = \tfrac{1}{2} Q_N^T R_N Q_N - \Phi_N^T Q_N + \text{constant} \tag{4}$$

where R_N is the correlation matrix in (2) and Φ_N is the column vector on the right-hand side of (2). Since R_N is positive definite, the minimum of $F(Q_N)$ exists. We find the gradient of F with respect to the vector Q_N:

$$\nabla_{Q_N} F = R_N Q_N - \Phi_N. \tag{5}$$

The equality

$$\nabla_{Q_N} F = \begin{bmatrix} 0 \\ 0 \\ \vdots \\ 0 \end{bmatrix} \tag{5a}$$

is then exactly (2). Thus, solving (2) is equivalent to finding the minimum of (4). We next find the minimum of F by means of the steepest descent technique. The nth iteration of the algorithm is

$$Q_N(n+1) = Q_N(n) - \mu \nabla_{Q_N(n)} F$$