計測コラム　　**emm140 号用**

基礎からの周波数分析（9）－「高速フーリエ変換（FFT）」

前回、**DFT（離散フーリエ変換）**の導出の説明をしましたが、この DFT は数値計算でフーリエ変換をするときに、基本的で非常に有用な式です。しかし、これを式通りに計算しようとすると、膨大な計算量が必要となりなかなか実用的には利用されていませんでしたが、1965 年に Cooley と Turkey によって高速フーリエ変換（Fast Fourier Transform、FFT）の算法が開発されて以来、非常にポピュラーに利用されるようになり、今ではディジタル信号処理の最も有用な技術となりました。最近では、TV 放送もディジタルとなっていて、OFDM と呼ばれるディジタル変調技術に FFT 技術が使われているくらいです。

今回は、**FFT（高速フーリエ変換）**の計算手法を簡単に説明します。

前回説明したように、DFT により、N 点の時間信号 x_n（n=0、.....、N−1）から N 点の周波数スペクトル X_k（k=0、.....、N−1）が得られます。すなわち；

$$X_k = \sum_{n=0}^{N-1} x_n W^{nk} \qquad \text{.....（1）}$$

ここで、

$$W = e^{-\frac{2\pi}{N}i} \qquad \text{.....（2）}$$

通常、式（1）の DFT は複素数で計算され、N 点の X_k を求めるためには N^2 回の複素数乗算が必要となります。今 N が複数の整数の積で表せる場合は、式（1）の演算をうまく小グループに分解してかつ W の周期性（周期 N）を使ってなるべく乗算の回数を減らして DFT 演算を高速に行う算法が FFT です。

FFT の計算点数 N（時間関数のサンプル点数）は、通常は演算分割の単純さとディジタルコンピュータのプログラムの実装の相性などから、2 のべき乗値（N=2^m）を選びます。例えば、N=1024（m=10）、N=2048（m=11）などです。これを**２基底 FFT** と呼びます。

以下の説明では、N=8（m=3）として説明します。N=8 を式（1）に代入すると；

$$X_k = \sum_{n=0}^{7} x_n W^{nk} \qquad \cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots (3)$$

$$W = e^{-\frac{\pi}{4}i} \qquad \cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots (4)$$

となります。ここで、W は単位円を負の角度方向（時計回り）に π/4（45°）だけ回転する機能を持つので、**回転因子**と呼びます。この回転因子 W は、1 回転すると同じ値を、また半回転では符号が逆の値となります。N=8 の場合での具体的な単位円上の点は図 1 となります。

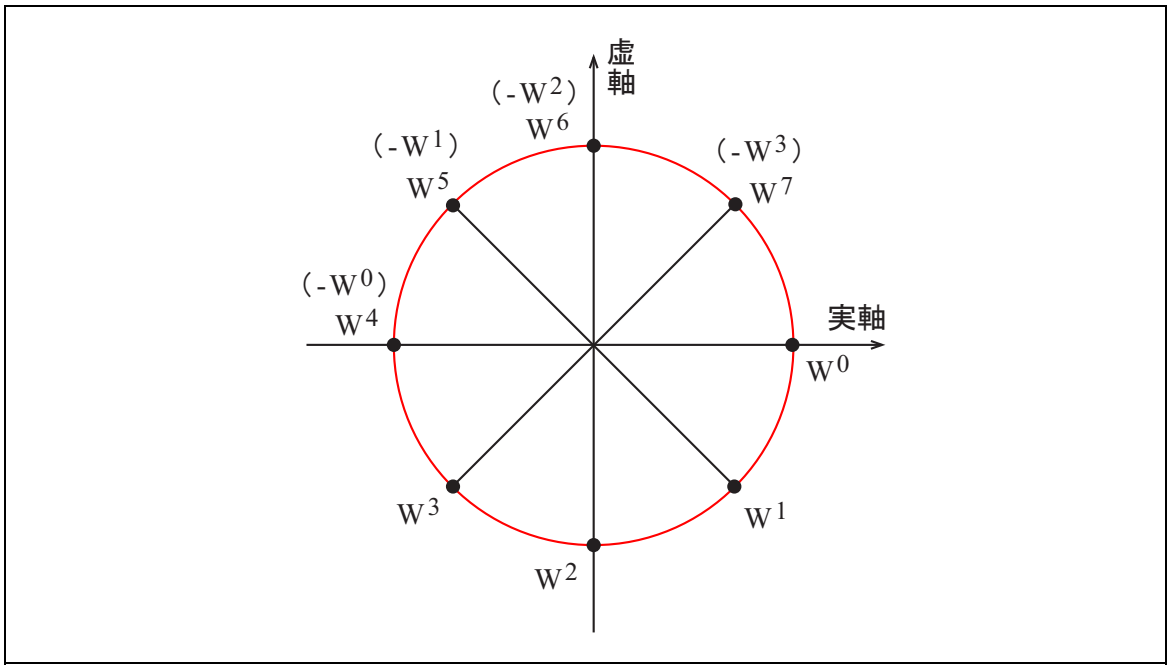

図 1　単位円上での N=8 の場合の回転因子 W

次に、時間信号 x_n を偶数列と奇数列に分割すると、式（3）は；

$$\begin{aligned} X_k &= \sum_{r=0}^{3} x_{2r} W^{2rk} + \sum_{r=0}^{3} x_{2r+1} W^{(2r+1)k} \\ &= \sum_{r=0}^{3} x_{2r} W^{2rk} + W^k \sum_{r=0}^{3} x_{2r+1} W^{2rk} \end{aligned} \qquad \cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots (5)$$

ここで、r = 0、…、3 とします。

式（5）は、偶数項 4 点 DFT と奇数項 4 点 DFT に回転因子 W^k を掛けたものとの和となっていることを示しています。$W^k = -W^{k-4}$ となる（図 1）ことに注意してこの演算の流れ図は図 2 のようになります。

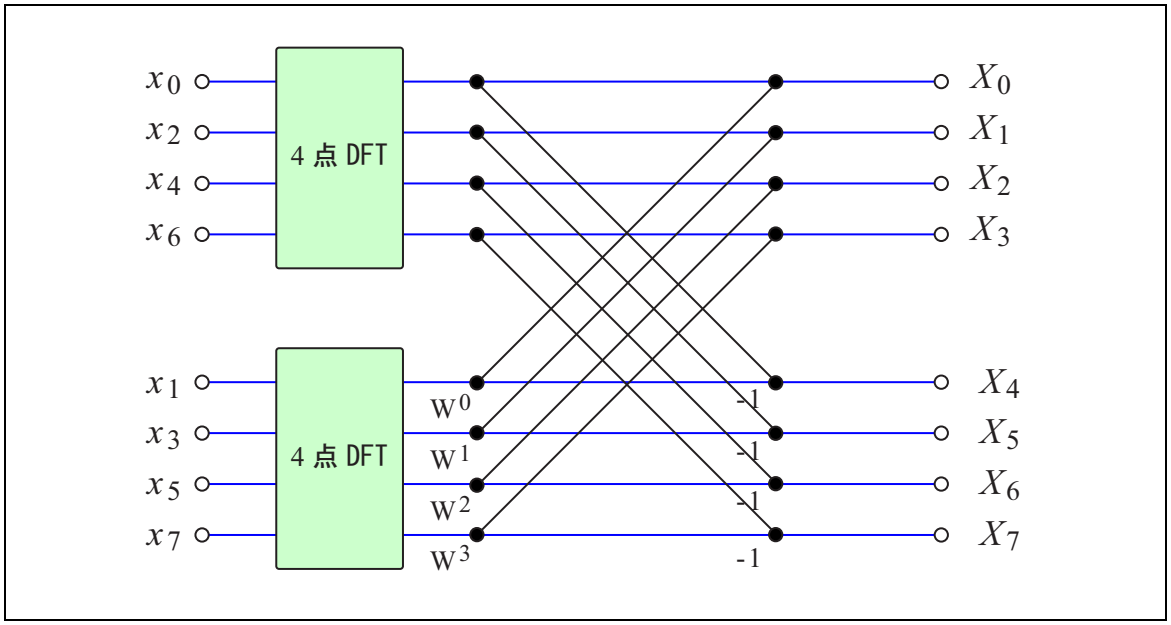

図 2　偶数項と奇数項それぞれ 2 つの 4 点 DFT に分解したときの式（5）の信号流れ図

全く同様にして、4 点 DFT を 2 つの 2 点 DFT に分解するなどして、最終的に 8 点 DFT の FFT は、図 3 のような信号流れ図となります。なお、本手法は時間信号を分割して求める手法なので、**時間間引き法**と呼ばれます。

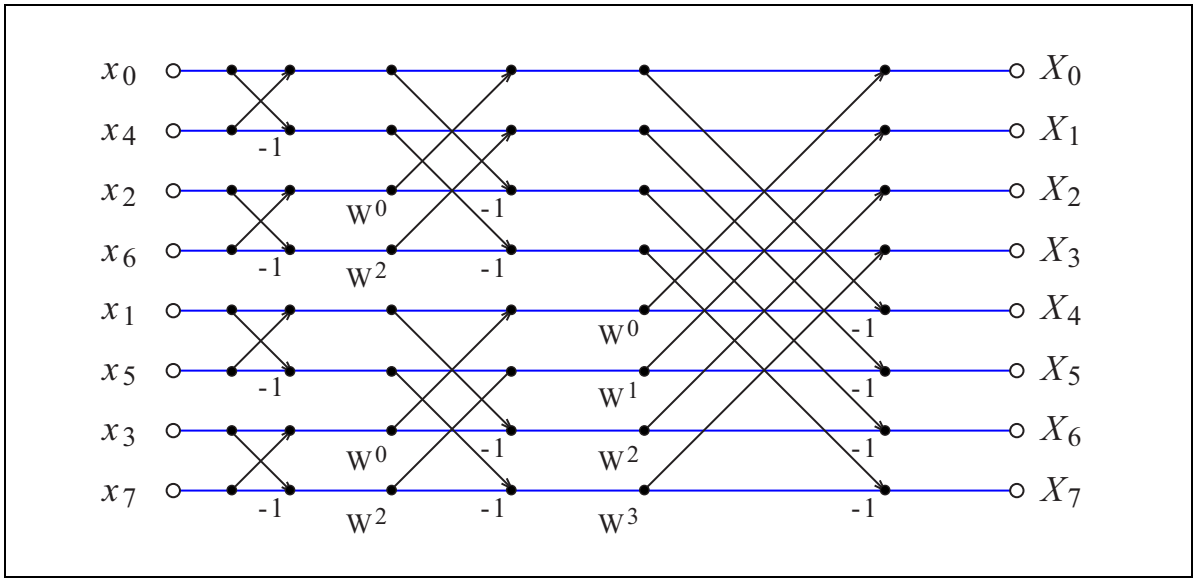

図 3　時間間引き法による 8 点 FFT 演算の信号流れ図

図 3 の流れ図をよく見ると、どの計算列も前の列と同じ一対の節点からのみの計算となっているのがわかります。すなわち、FFT 演算は式（6）の形の積和対を基本の演算としています。

$$\left.\begin{aligned} x_\alpha &= x_\alpha + x_\beta W^p \\ x_\beta &= x_\alpha - x_\beta W^p \end{aligned}\right\} \qquad \cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots (6)$$

この演算を流れ図にすると、図 4 のようになり、蝶の羽のような形をしているので**バタフライ演算**と呼ばれます。

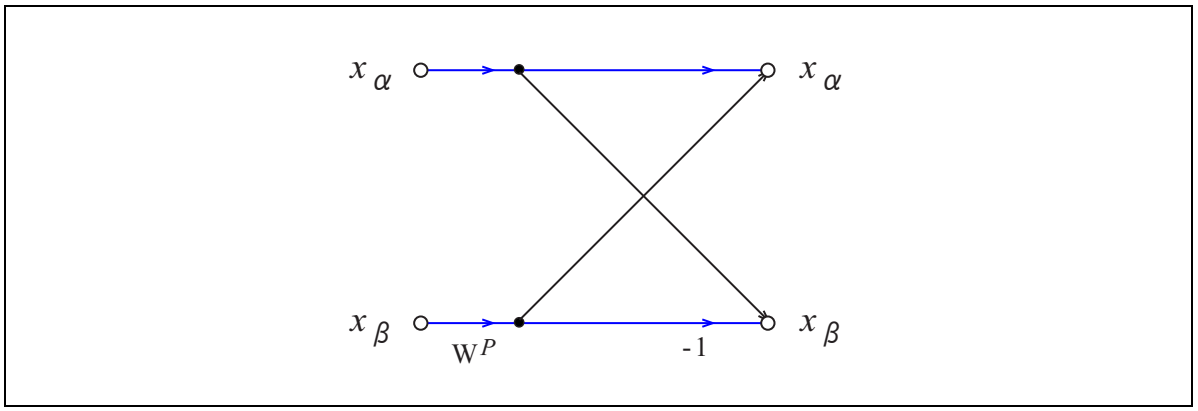

図 4　時間間引き法によるバタフライ演算

式（6）の意味するところは、一対の節点データを x_α、x_β とすると、このデータを時間信号（複素数）のメモリーから呼び出して右辺の計算を行いその結果を同じ x_α、x_β のメモリーに格納するという操作となります。このように、FFT の基礎的な演算であるバタフライ演算は、一対の節点データだけで行えるので、**In-Place メモリー演算**と呼ばれ、メモリー量を少なくかつメモリーアクセス回数も少なくなるので非常に効率が良いというメリットがあります。

図 3 の流れ図に関して、もう 1 点注目すべき点があります。時間信号 x_n の数列がサンプルした通常の順序でなくなっていることです。そのため、FFT 演算を実行する前に、図 3 のような並びに並び替える必要があります。これは、数列の順序数値を 2 進数にしてみると、簡単に並び替えることができます。

N=8 の場合の例が表 1 です。このように、元の数列の順序番号を 2 進数に書き換え、そのビット列をひっくり返すことにより、図 3 のような並びを導き出すことが可能です。この操作を**ビット逆順**（Bit reverse、BTR）と呼びます。この操作も通常のコンピュータで簡単に行うことができます。

表 1　N=8 の場合のビット逆順による並び替え

原順序		ビット逆順序	
10 進数	2 進数	2 進数	10 進数
0	0 0 0	0 0 0	0
1	0 0 1	1 0 0	4
2	0 1 0	0 1 0	2
3	0 1 1	1 1 0	6
4	1 0 0	0 0 1	1
5	1 0 1	1 0 1	5
6	1 1 0	0 1 1	3
7	1 1 1	1 1 1	7

今度は、DFT 演算結果である X_k（k=0、.....、$N-1$）を偶数と奇数とに分割していくことにより、時間間引き法と同様にして、**周波数間引き法**を導き出すことが出来ます。図 5 が、N=8 の場合の周波数間引き法による 8 点 FFT の信号流れ図です。

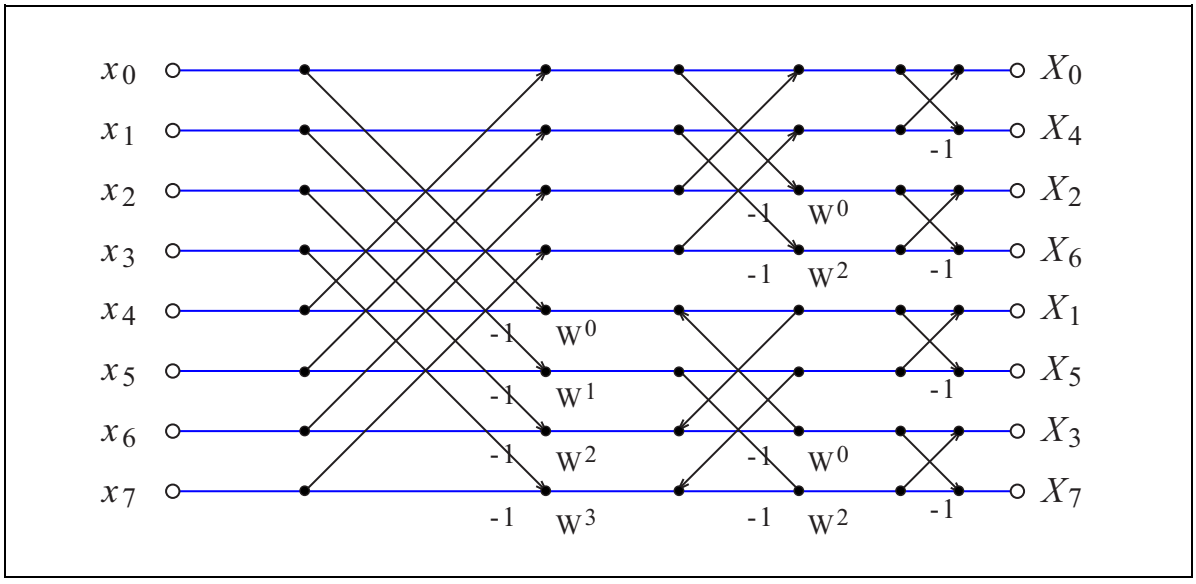

図 5　周波数間引き法による 8 点 FFT 演算の信号流れ図

この場合のバタフライ演算は、式（7）となり、その信号流れ図は、図 6 となります。

$$\left.\begin{aligned} x_\alpha &= x_\alpha + x_\beta \\ x_\beta &= (x_\alpha - x_\beta)W^p \end{aligned}\right\} \quad \cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots (7)$$

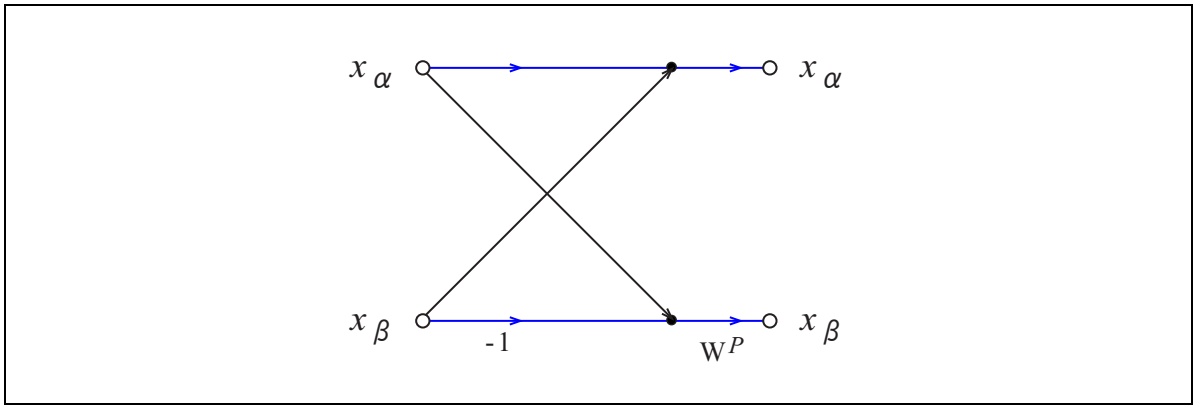

図 6　周波数間引き法によるバタフライ演算

図 5 にあるように、周波数間引き法では通常演算結果の X_k が周波数順とならず、BTR 処理が必要となります。表 1 の逆の処理をすることにより、正しい順序列とすることが出来ます。

その他いろいろな演算方法があり、時間間引き法で結果に BTR する場合や、周波数間引き法で最初に BTR する方法も考えられますが、これらの方法では、回転因子 W をアクセスするときに、BTR 処理が必要となるため、あまり使われません。

この FFT 演算により、通常の DFT と比較してどれくらい演算時間が短縮できるか考えます。式（6）のバタフライ演算で 1 回の複素数乗算を行うとして N 点の FFT で $\frac{N}{2}\log_2 N$ のバタフライ演算を行いますから、DFT の乗算回数 N^2 と比較すると、N=1024（=2^{10}）の場合で約 1/200、N=16384（=2^{14}）の場合では、約 1/2300 と驚異的に短縮されます。
また、乗算回数が減少することにより丸め誤差による計算誤差の低減にもつながり、FFT 算法の大きなメリットと言われております。

最後に、まとめです。

(1) FFT は DFT を高速に演算するアルゴルリズムです。
(2) FFT 計算点数 *N* は、通常 1024 や 2048 のような 2 のべき乗値を採用して、これを 2 基底 FFT と呼びます。
(3) FFT 算法は、大きく時間間引き法と周波数間引き法があります。
(4) FFT の基本演算はバタフライ演算と言われ、In-Place メモリー演算なので、メモリーを節約できます。
(5) FFT は通常の DFT と比較して乗算回数を大幅に減らすことが出来るので、演算スピードアップだけでなく、計算誤差低減効果も期待できます。

【キーワード】
DFT（離散フーリエ変換）、FFT（高速フーリエ変換）、2 基底 FFT、回転因子、時間間引き法、バタフライ演算、ビット逆順（BTR）、In-Place メモリー演算、周波数間引き法

【参考資料】

1. 「高速フーリエ変換」E. ORAN BRIGHAM 著　科学技術出版社
2. 「ディジタルフーリエ解析（I）－基礎編－」城戸健一著　コロナ社
3. 「信号処理」森下巖/小畑秀文 共著　朝倉書店

以上
（Hima）