2011 年度秋学期　解析応用　第６回

第１部・フーリエ解析／ 離散フーリエ変換と高速フーリエ変換

離散フーリエ変換

ここまで，フーリエ解析についていろいろと説明してきましたが，今回はフーリエ変換をコンピュータで実際に計算することを考えます。ここまでの説明では，実数値で定義された関数のフーリエ変換を考えてきましたが，コンピュータで扱える計算は離散的なものだけです。そこで，実数値で定義された関数の値を一定間隔で取り出す「サンプリング」を行って関数を離散化し，その関数のフーリエ変換を離散的なまま計算する方法について考えてみましょう。

実数値で定義された関数 $f(x)$ を間隔 T でサンプリングした関数 $f_T(x)$ は，デルタ関数が等間隔に並んだくし形関数 $\mathrm{comb}_T(x)$

$$\mathrm{comb}_T(x) = \sum_{n=-\infty}^{\infty} \delta(x - nT) \tag{1}$$

を用いて

$$f_T(x) = f(x)\mathrm{comb}_T(x). \tag{2}$$

と表されます。

サンプリングされた関数 $f_T(x)$ のフーリエ変換は，

$$\begin{aligned} FT[f_T(x)](\nu) &= FT[f(x)\mathrm{comb}_T(x)](\nu) \\ &= FT[f(x)](\nu) * FT[\mathrm{comb}_T(x)](\nu) \end{aligned} \tag{3}$$

というコンヴォリューションで表されます。

ここで，$FT[\mathrm{comb}_T(x)](\nu)$ がどうなるか考えてみましょう。$\mathrm{comb}_T(x)$ は周期 T の周期関数ですから，フーリエ級数を考えてみます。周期 T とはすなわち周波数が $1/T$ ですから，$\mathrm{comb}_T(x)$ は周波数が $1/T$ の整数倍である正弦波の級数，すなわち

$$\mathrm{comb}_T(x) = \sum_{n=-\infty}^{\infty} a_n \exp(i2\pi\frac{n}{T}x) \tag{4}$$

で表されます（n は整数)。フーリエ係数 a_n は，

$$\begin{aligned} a_n &= \frac{1}{T}\int_{-\frac{T}{2}}^{\frac{T}{2}} \mathrm{comb}_T(x)\exp(-i2\pi\frac{n}{T}x)dx \\ &= \frac{1}{T}\int_{-\frac{T}{2}}^{\frac{T}{2}} \delta(x)\exp(-i2\pi\frac{n}{T}x)dx \\ &= \frac{1}{T}\exp(-i2\pi\frac{n}{T}\cdot 0) = \frac{1}{T} \end{aligned} \tag{5}$$

となります。よって，フーリエ級数は

$$\mathrm{comb}_T(x) = \frac{1}{T}\sum_{n=-\infty}^{\infty} \exp(i2\pi\frac{n}{T}x) \tag{6}$$

となります。一方，前回説明したとおり

$$FT[\exp(i2\pi\frac{n}{T}x)](\nu) = \delta(\nu - \frac{n}{T}) \tag{7}$$

ですから，

$$\begin{aligned} FT[\mathrm{comb}_T(x)](\nu) &= \frac{1}{T}\sum_{n=-\infty}^{\infty}\delta(\nu - \frac{n}{T}) \\ &= \frac{1}{T}\mathrm{comb}_{\frac{1}{T}}(\nu) \end{aligned} \tag{8}$$

となります。すなわち，周期 T のくし形関数のフーリエ変換は，周期 $1/T$ のくし形関数となります。

以上から，

$$\begin{aligned} FT[f_T(x)](\nu) &= FT[f(x)](\nu) * FT[\mathrm{comb}_T(x)](\nu) \\ &= FT[f(x)](\nu) * \mathrm{comb}_{\frac{1}{T}}(\nu) \end{aligned} \tag{9}$$

となります。このことは，周波数空間においては，もとの関数 $f(x)$ のフーリエ変換 $FT[f_T(x)](\nu)$ が $1/T$ 間隔で繰り返し現れることを意味しています。

この状態では，実空間の関数 $g(t)$ はサンプリングによって離散化しましたが，周波数空間では離散的にはなっておらず，まだ「コンピュータで計算するために離散化する」という目的は達せられていません。そこで，周波数空間のほうもサンプリングすることにします。周波数空間での 1 周期 $1/T$ の間に N 回のサンプリングをすることにすると，サンプリング間隔は $1/NT$ になります。

このような周波数空間でのサンプリングは，実空間では何をしていることになるのでしょうか？ 周波数空間での間隔 $1/NT$ のサンプリングは，周波数空間で間隔 $1/NT$ のくし形関数 $\mathrm{comb}_{1/(NT)}$ をかけ算することに相当します。フーリエ変換したときにそのように「とびとび」になるのは，実空間では周期 NT の周期関数のはずです。

実空間の関数 $f_T(x)$ は間隔 T でサンプリングされていました。ということは，周波数空間でサンプリングされた関数は，「実空間の $f_T(x)$ のうち N 個の一続きのサンプルだけを取り出し，この N 個のサンプルを無限に繰り返してコピーして，周期 NT の周期関数を作ったもの」をフーリエ変換したものになっているわけです[1]。

さて，ここまでの説明では，$f(\nu)$ が先にあって，$f_T(x)$ は $f(x)$ をサンプリングしたものと考えてきました。ここで，発想を転換して，$f(x)$ がどんなものかは不明で，$f_T(x)$ だけが与えられていると考えます。そうすると，$f_T(x)$ は単なる数値の列となりますから，これを $u(n)$ と表すことにします。$u(n)$ は，$f(x)$ やそれをサンプリングした $f_T(x)$ のように，連続的に変化する x にしたがって「測る」ものではなく，$n = 1, 2, \ldots$ と「数える」ものとなります。このような表現は，$u(n)$ で n が 1 だけちがうことは，$f_T(x)$ のほうでは長さ（あるいは時間）T だけ離れていることに対応していますから，サンプリング間隔 T を 1([単位長さ（あるいは単位時間）]) とする新しい単位で時間を表現していると考えることもできます。このようにして，

$$u(n) = f_T(nT) \tag{10}$$

と表します。

[1] 一部ごまかしがあります。詳しくは付録 I をみてください。

ここまでの説明で述べたように，$f(x)$ は，間隔 T でサンプリングされ，周期 NT の周期関数になっているとみなしていますから，(3) 式のフーリエ変換の実数全体での積分のかわりに，$u(n)$ の 1 周期分（N 項）だけを計算することにします。

また，(3) 式は，デルタ関数の並びとある関数との積を実数全体で積分する形になっています。一方，デルタ関数は「インパルスのある場所以外では 0，積分すると 1」ですから，ある関数とデルタ関数との積を実数全体で積分すると，その関数からインパルスのある位置での値だけを切り出したものになります。したがって，離散的に計算する場合は，積分の代わりに，インパルスの位置でのその関数の値を合計することになります。

以上から，(3) 式のフーリエ変換のかわりに

$$U(k) = \sum_{n=0}^{N-1} u(n) \exp\left(-i2\pi \frac{k}{N} n\right) \quad (k = 0, 1, \ldots, N-1) \tag{11}$$

を計算します。(11) 式を**離散フーリエ変換** (discrete Fourier transformation, DFT) といいます。計算機で行うフーリエ変換は，すべてこの離散フーリエ変換であり，本来のフーリエ変換とは異なることに注意してください。

同様に，周波数空間でも，周期関数がサンプリングされた形になっており，1 周期分が N 項からなりますから，

$$u(j) = \frac{1}{N} \sum_{k=0}^{N-1} U(k) \exp\left(i2\pi \frac{j}{N} k\right) \quad (j = 0, 1, \ldots, N-1) \tag{12}$$

を計算し，これを**離散逆フーリエ変換**といいます。(12) 式の右辺に (11) 式を代入すると，

$$\begin{aligned} &\frac{1}{N} \sum_{k=0}^{N-1} \left(\sum_{n=0}^{N-1} u(n) \exp\left(-i2\pi \frac{k}{N} n\right) \right) \exp\left(i2\pi \frac{j}{N} k\right) \\ &= \frac{1}{N} \sum_{n=0}^{N-1} u(n) \sum_{k=0}^{N-1} \exp\left(i2\pi \frac{j-n}{N} k\right) \end{aligned} \tag{13}$$

となります。後ろ側の総和は，$n \neq j$ のとき，公比 $\exp\left(i2\pi \frac{j-n}{N}\right)$，項数 N の等比数列の和なので，

$$\frac{1 - \{\exp(i2\pi)\}^{(j-n)k}}{1 - \exp\left(i2\pi \frac{j-n}{N}\right)} = \frac{1 - 1^{(j-n)k}}{1 - \exp\left(i2\pi \frac{j-n}{N}\right)} = 0 \tag{14}$$

となり，$j = n$ のときは $\sum_{k=0}^{N-1} 1 = N$ となります。よって，(13) 式は $\frac{1}{N} N \cdot u(j) = u(j)$ となり，(11) 式の $u(n)$ と (11) 式の $U(k)$ が，確かにフーリエ変換対になっていることがわかります[2]。

例として，サンプリング間隔 $T = 1$([秒]) でサンプリングされ，$N = 256$ 点からなる信号 $u(n)$ があるとします。この信号を離散フーリエ変換して $U(k)$ を得たとき，k の 1 刻みは，もとの信号における周波数空間では何 ([1/秒]) の周波数に相当するかを考えてみましょう。ここまでの説明の通り，周波数空間においては，サンプリングの間隔は $1/NT$([1/秒]) です。この例では $T = 1$[秒]，$N = 256$(刻み) なので，1 刻みは 1/256([1/秒]) に相当します。

---

[2]離散逆フーリエ変換の係数 $1/N$ は，この関係がなりたつようにつけられています。

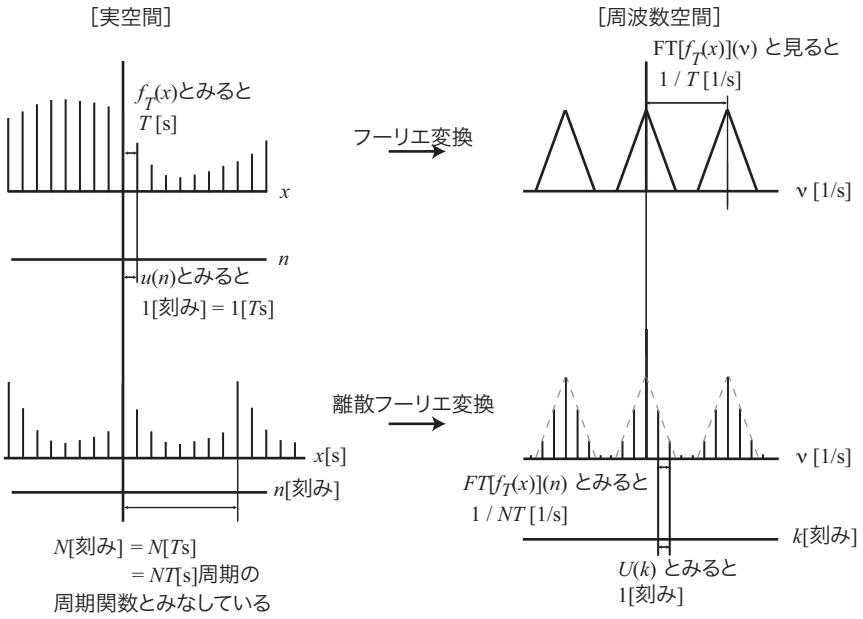

図 1: 離散フーリエ変換.

高速フーリエ変換

高速フーリエ変換 (Fast Fourier Transformation, FFT) は，離散フーリエ変換の計算における指数関数の対称性を利用して，計算を分割してまとめることで，計算に含まれるかけ算の回数を減らす工夫です。ここでは，もっとも有名な Cooley and Tukey の高速フーリエ変換を説明します。

高速フーリエ変換は，計算する点の数（(11) 式の N）が 2 の累乗のときに，もっとも効果を発揮します。ここでは，簡単のために $N = 8$ としましょう。すると，(11) 式は

$$U(k) = \sum_{n=0}^{7} u(n) \exp\left(-i2\pi \frac{k}{8} n\right) \quad (k = 0, 1, \ldots, 7) \tag{15}$$

となります。この計算は，8^2 回のかけ算でできています。

この計算を，行列を用いて表してみます。ここで，

$$W \equiv \exp\left(-i\frac{2\pi}{8}\right) \tag{16}$$

とおくと，

$$\begin{pmatrix} U(0) \\ U(1) \\ U(2) \\ U(3) \\ U(4) \\ U(5) \\ U(6) \\ U(7) \end{pmatrix} = \begin{pmatrix} W^{0\cdot0} & W^{0\cdot1} & W^{0\cdot2} & W^{0\cdot3} & W^{0\cdot4} & W^{0\cdot5} & W^{0\cdot6} & W^{0\cdot7} \\ W^{1\cdot0} & W^{1\cdot1} & W^{1\cdot2} & W^{1\cdot3} & W^{1\cdot4} & W^{1\cdot5} & W^{1\cdot6} & W^{1\cdot7} \\ W^{2\cdot0} & W^{2\cdot1} & W^{2\cdot2} & W^{2\cdot3} & W^{2\cdot4} & W^{2\cdot5} & W^{2\cdot6} & W^{2\cdot7} \\ W^{3\cdot0} & W^{3\cdot1} & W^{3\cdot2} & W^{3\cdot3} & W^{3\cdot4} & W^{3\cdot5} & W^{3\cdot6} & W^{3\cdot7} \\ W^{4\cdot0} & W^{4\cdot1} & W^{4\cdot2} & W^{4\cdot3} & W^{4\cdot4} & W^{4\cdot5} & W^{4\cdot6} & W^{4\cdot7} \\ W^{5\cdot0} & W^{5\cdot1} & W^{5\cdot2} & W^{5\cdot3} & W^{5\cdot4} & W^{5\cdot5} & W^{5\cdot6} & W^{5\cdot7} \\ W^{6\cdot0} & W^{6\cdot1} & W^{6\cdot2} & W^{6\cdot3} & W^{6\cdot4} & W^{6\cdot5} & W^{6\cdot6} & W^{6\cdot7} \\ W^{7\cdot0} & W^{7\cdot1} & W^{7\cdot2} & W^{7\cdot3} & W^{7\cdot4} & W^{7\cdot5} & W^{7\cdot6} & W^{7\cdot7} \end{pmatrix} \begin{pmatrix} u(0) \\ u(1) \\ u(2) \\ u(3) \\ u(4) \\ u(5) \\ u(6) \\ u(7) \end{pmatrix} \tag{17}$$

と表すことができます。この計算をそのまま行うと，8^2 回のかけ算を行う必要があります。

ここで，

$$W^8 = \exp\left(-i2\pi\frac{8}{8}\right) = 1 = W^0 \tag{18}$$

ですから，(17) 式は

$$\begin{pmatrix} U(0) \\ U(1) \\ U(2) \\ U(3) \\ U(4) \\ U(5) \\ U(6) \\ U(7) \end{pmatrix} = \begin{pmatrix} W^0 & W^0 & W^0 & W^0 & W^0 & W^0 & W^0 & W^0 \\ W^0 & W^1 & W^2 & W^3 & W^4 & W^5 & W^6 & W^7 \\ W^0 & W^2 & W^4 & W^6 & W^0 & W^2 & W^4 & W^6 \\ W^0 & W^3 & W^6 & W^1 & W^4 & W^7 & W^2 & W^5 \\ W^0 & W^4 & W^0 & W^4 & W^0 & W^4 & W^0 & W^4 \\ W^0 & W^5 & W^2 & W^7 & W^4 & W^1 & W^6 & W^3 \\ W^0 & W^6 & W^4 & W^2 & W^0 & W^6 & W^4 & W^2 \\ W^0 & W^7 & W^6 & W^5 & W^4 & W^3 & W^2 & W^1 \end{pmatrix} \begin{pmatrix} u(0) \\ u(1) \\ u(2) \\ u(3) \\ u(4) \\ u(5) \\ u(6) \\ u(7) \end{pmatrix} \tag{19}$$

となります。さらに

$$W^{m-4} = \exp\left(-i2\pi\frac{-4}{8}\right)\exp\left(-i2\pi\frac{m}{8}\right) = \exp(i\pi)\,W^m = -W^m, \tag{20}$$

すなわち $W^m = -W^{m-4}$ ですから，$m = 4, 5, 6, 7$ についてこれを適用すると

$$\begin{pmatrix} U(0) \\ U(1) \\ U(2) \\ U(3) \\ U(4) \\ U(5) \\ U(6) \\ U(7) \end{pmatrix} = \begin{pmatrix} W^0 & W^0 & W^0 & W^0 & W^0 & W^0 & W^0 & W^0 \\ W^0 & W^1 & W^2 & W^3 & -W^0 & -W^1 & -W^2 & -W^3 \\ W^0 & W^2 & -W^0 & -W^2 & W^0 & W^2 & -W^0 & -W^2 \\ W^0 & W^3 & -W^2 & W^1 & -W^0 & -W^3 & W^2 & -W^1 \\ W^0 & -W^0 & W^0 & -W^0 & W^0 & -W^0 & W^0 & -W^0 \\ W^0 & -W^1 & W^2 & -W^3 & -W^0 & W^1 & -W^2 & W^3 \\ W^0 & -W^2 & -W^0 & W^2 & W^0 & -W^2 & -W^0 & W^2 \\ W^0 & -W^3 & -W^2 & -W^1 & -W^0 & W^3 & W^2 & W^1 \end{pmatrix} \begin{pmatrix} u(0) \\ u(1) \\ u(2) \\ u(3) \\ u(4) \\ u(5) \\ u(6) \\ u(7) \end{pmatrix} \tag{21}$$

となります。ここで，$U(2)$ に対応する行を見ると，左半分と右半分が同じになっています。したがって，

$$U(2) = \begin{pmatrix} W^0 & W^2 & -W^0 & -W^2 \end{pmatrix} \begin{pmatrix} u(0)+u(4) \\ u(1)+u(5) \\ u(2)+u(6) \\ u(3)+u(7) \end{pmatrix} \tag{22}$$

とまとめることができます。このようにまとめることで，元は 8 回だったかけ算の回数が，4 回に減っています。$U(0), U(4), U(6)$ も同じです。

また，$U(1)$ に対応する行をみると，左半分と右半分は符号だけが違っています。そこで，

$$U(1) = \begin{pmatrix} W^0 & W^1 & W^2 & W^3 \end{pmatrix} \begin{pmatrix} u(0)-u(4) \\ u(1)-u(5) \\ u(2)-u(6) \\ u(3)-u(7) \end{pmatrix} \tag{23}$$

とまとめることができます。$U(3), U(5), U(7)$ も同じです。

以上から，

$$\begin{aligned}
\begin{pmatrix} U(0) \\ U(2) \\ U(4) \\ U(6) \end{pmatrix} &= \begin{pmatrix} W^0 & W^0 & W^0 & W^0 \\ W^0 & W^2 & -W^0 & -W^2 \\ W^0 & -W^0 & W^0 & -W^0 \\ W^0 & -W^2 & -W^0 & W^2 \end{pmatrix} \begin{pmatrix} u(0)+u(4) \\ u(1)+u(5) \\ u(2)+u(6) \\ u(3)+u(7) \end{pmatrix} \\
\begin{pmatrix} U(1) \\ U(3) \\ U(5) \\ U(7) \end{pmatrix} &= \begin{pmatrix} W^0 & W^1 & W^2 & W^3 \\ W^0 & W^3 & -W^2 & W^1 \\ W^0 & -W^1 & W^2 & -W^3 \\ W^0 & -W^3 & -W^2 & -W^1 \end{pmatrix} \begin{pmatrix} u(0)-u(4) \\ u(1)-u(5) \\ u(2)-u(6) \\ u(3)-u(7) \end{pmatrix}
\end{aligned} \tag{24}$$

となります。上の計算について，さらに

$$W^{m-2} = \exp\left(-i2\pi\frac{-2}{8}\right)\exp\left(-i2\pi\frac{m}{8}\right) = \exp\left(i\frac{\pi}{2}\right)W^m = iW^m, \tag{25}$$

すなわち $W^m = -iW^{m-2}$ ですから，(24) 式の下半分は

$$\begin{pmatrix} U(1) \\ U(3) \\ U(5) \\ U(7) \end{pmatrix} = \begin{pmatrix} W^0 & W^1 & -iW^0 & -iW^1 \\ W^0 & -iW^1 & iW^0 & W^1 \\ W^0 & -W^1 & -iW^0 & iW^1 \\ W^0 & iW^1 & iW^0 & -W^1 \end{pmatrix} \begin{pmatrix} u(0)-u(4) \\ u(1)-u(5) \\ u(2)-u(6) \\ u(3)-u(7) \end{pmatrix} \tag{26}$$

となります。

これを使って，再び同様の操作をすることができ，

$$\begin{aligned}
\begin{pmatrix} U(0) \\ U(4) \end{pmatrix} &= \begin{pmatrix} W^0 & W^0 \\ W^0 & -W^0 \end{pmatrix} \begin{pmatrix} (u(0)+u(4))+(u(2)+u(6)) \\ (u(1)+u(5))+(u(3)+u(7)) \end{pmatrix} \\
\begin{pmatrix} U(2) \\ U(6) \end{pmatrix} &= \begin{pmatrix} W^0 & W^2 \\ W^0 & -W^2 \end{pmatrix} \begin{pmatrix} (u(0)+u(4))-(u(2)+u(6)) \\ (u(1)+u(5))-(u(3)+u(7)) \end{pmatrix} \\
\begin{pmatrix} U(1) \\ U(5) \end{pmatrix} &= \begin{pmatrix} W^0 & W^1 \\ W^0 & -W^1 \end{pmatrix} \begin{pmatrix} (u(0)-u(4))-i\,(u(2)-u(6)) \\ (u(1)-u(5))-i\,(u(3)-u(7)) \end{pmatrix} \\
\begin{pmatrix} U(3) \\ U(7) \end{pmatrix} &= \begin{pmatrix} W^0 & -iW^1 \\ W^0 & iW^1 \end{pmatrix} \begin{pmatrix} (u(0)-u(4))+i\,(u(2)-u(6)) \\ (u(1)-u(5))+i\,(u(3)-u(7)) \end{pmatrix}
\end{aligned} \tag{27}$$

とすることができます。この段階に達した時，$U(k)$ のおのおのについて，行列の要素のかけ算を 2 回と，± 1 や $\pm i$ をかける計算を 3 回，すなわち $\log_2 8$ 回行っています。一般に，点数が N のとき，N^2 回必要だったかけ算を $Nlog_2N$ に比例した回数に減らすことができます。

このように，問題を半分，さらに半分，さらに半分…と分けていく方法は「分割統治 (divide and conquer) 法」とよばれ，アルゴリズム開発ではよく用いられるものです。いわゆる「クイックソート」も，同様の考えにもとづいています。

付録１．周波数空間でのサンプリングと，実空間での周期関数の関係

実空間でサンプリングされた関数 $f_T(x)$ から，幅 NT に入る N 個のサンプルを切り出したものは，本当は

$$f_T(x) \times \mathrm{rect}(\frac{x}{NT}) \tag{A1}$$

になっているはずです。(A1) 式の関数を，さらに周期 NT で繰り返したものは，くし形関数を使うと

$$\Big(f_T(x) \times \mathrm{rect}(\frac{x}{NT})\Big) * \mathrm{comb}_{NT}(x) \tag{A2}$$

となります。(A2) 式の関数のフーリエ変換は，

$$\begin{aligned} & FT[\Big(f(x)\mathrm{comb}_T(x) \times \mathrm{rect}(\frac{x}{NT})\Big) * \mathrm{comb}_{NT}(x)] \\ &= \Big(FT[f(x)\mathrm{comb}_T(x)] * FT[\mathrm{rect}(\frac{x}{NT})]\Big) \times FT[\mathrm{comb}_{NT}(x)] \\ &= \Big(FT[f_T(x)] \times \mathrm{comb}_{\frac{1}{NT}}(x\nu)\Big) * \mathrm{sinc}(\frac{\nu}{1/(NT)}) \end{aligned} \tag{A3}$$

と表すことができます。

この式の前半のかけ算は，実空間でサンプリングされた $f_T(x)$ のフーリエ変換を，さらに間隔 NT でサンプリングすることを示しています。サンプリングされた結果は，デルタ関数の並びになっています。前回説明したように，ある関数とデルタ関数とのたたみ込みは，その関数自身になります。したがって，(A3) 式の後半のたたみ込みにより，sinc 関数が間隔 $1/(NT)$ で並ぶことになります。しかし，関数 $\mathrm{sinc}(\frac{\nu}{1/(NT)})$ は，間隔 $1/(2NT)$ ごとにゼロになりますので，あるデルタ関数のところに位置する sinc 関数の影響は，他のデルタ関数の位置には及びません。したがって，(A2) 式で表される周期関数のフーリエ変換が，確かに $f_T(\nu)$ をフーリエ変換し，周波数空間でさらにサンプリングしたものに相当しています。