# テーマD　線形システムと画像処理への応用

## 1　はじめに

### 1.1　連絡事項

- **プログラム等の事前配布のサイト：**
  `http://www.ide.titech.ac.jp/~yamasita/Exp/index.html`
  を必ず確認しておくこと。
- 上記 Web 上で案内した，eclipse を使って Hello World を表示するまでの基本的な使い方 (youtube.com へのリンク) を，見ておくこと。

### 1.2　目的

本実験の目的は,「線形システム論 (国際開発工学)」などにおいて学習した，フーリエ級数，フーリエ変換を具体的に実現するための，離散フーリエ変換 (DFT) を学習し,「情報処理実習」で学習した Java 言語を用いて，プログラムで実現する。また，JMF (Java Media Framework) を使って，Web カメラから画像を取得し，DFT を使った畳み込み計算やウィーナーフィルタを，画像処理へ応用する実験を行う。

## 2　離散フーリエ変換

「線形システム論 (国際開発工学)」において，時間信号を様々な周期をもつ正弦波などの和に分解する，フーリエ級数展開とフーリエ変換を勉強した。前者の定義域は有限区間あるいはその繰り返し，後者の定義域は無限区間であるという違いがあった。しかしながら，両者は連続変数の関数を扱うものであった。

一方，コンピュータでデータを処理する場合は，連続関数 $f(t)$ を直接的に扱うことが難しい。そのため，離散フーリエ変換 (Discrete Fourier Transform, DFT) を使う。本節では，まず，フーリエ級数展開とフーリエ変換を復習したのち，この離散フーリエ変換を説明する。さらに，画像を扱うための 2 次元離散フーリエ変換について説明する。

### 2.1　フーリエ級数展開

0 から $T$ までの関数 $g(t)$ に対して，そのフーリエ級数展開係数 $c_n$ は,

$$c_n = \frac{1}{T}\int_0^T g(\tau)e^{-\frac{2\pi i}{T}n\tau}d\tau$$

によって求まる。この展開係数によって，$g(t)$ をフーリエ級数展開

$$g(t) = \sum_{n=-\infty}^{\infty} c_n e^{\frac{2\pi i}{T}nt}$$

によって表すことができる。オイラーの定理から

$$e^{\frac{2\pi i}{T}nt} = \cos\frac{2\pi i}{T}nt + i\ \sin\frac{2\pi i}{T}nt$$

が成立する。従って，上式は $g(t)$ を重みをかけた三角関数の和で表していることになる。そして，$c_n$ が $g(t)$ の周波数成分の強さを表している。ここで注意すべきことは，$g(t)$ が実数関数であっても，一般には $c_m$ は複素数になるということである。

$g(t)$ のフーリエ変換 $G(f)$ は,

$$G(f) = \int_{-\infty}^{\infty} g(\tau) e^{-2\pi i f \tau} d\tau$$

によって求めることができる。また，$G(f)$ のフーリエ逆変換

$$g(t) = \int_{-\infty}^{\infty} G(f) e^{2\pi i f t} df$$

によって，$G(f)$ から $g(t)$ を求めることができる。この式でも，$g(t)$ が $G(f)$ を重みとした三角関数の和 (積分) によって表され，$G(f)$ が $g(t)$ の周波数成分の強さを表している。

数値計算をコンピュータで行う現代では，連続変数関数 $g(t)$ を直接的に扱うことが難しい。そのため，離散フーリエ変換 (Discrete Fourier Transform, DFT) が使われる。DFT では，データは有限個の離散時間上の値である。いま，$N$ 個のデータを $g_n$ $(n = 0, 1, 2, \ldots, N-1)$ で表す。このデータを DFT した結果 $G_m$ $(m = 0, 1, 2, \ldots, N-1)$ は,

$$G_m = \sum_{n=0}^{N-1} g_m e^{-\frac{2\pi i}{N} nm}$$

で与えられる。$G_m$ の逆離散フーリエ変換 (Inverse DFT, IDFT, 逆 DFT) は,

$$g_n = \frac{1}{N} \sum_{m=0}^{N-1} G_m e^{\frac{2\pi i}{N} nm}$$

で与えられる。

この 2 つの式は，$g_n$ と $G_m$ をすべての整数 $n, m$ 上で定義された周期 $N$ の周期関数に拡張しても成立する。**以下ではすべて，このような周期関数として考えることにする。**

フーリエ級数展開同様に，$g_n$ が実数であっても，一般には $G_m$ は複素数になる。ただし，$g_n$ が実数の場合には，次式が成立する。

$$G_{N-m} = \overline{G_m}$$

従って，$g_n$ が実数の場合，$g_m$ の実数としての自由度は $N$ であるが，上式から $G_m$ の実数としての自由度も $N$ ということになる。

周期関数を考えているので,

$$G_{-m} = G_{N-m} = \overline{G_m}$$

と書くこともできる。これは，実数値関数 $g(t)$ をフーリエ変換したときに,

$$G(-f) = \overline{G(f)}$$

が成立することに相当する。

この関係を証明する。$\overline{g_n} = g_n$，$\overline{e^{i\theta}} = e^{-i\theta}$ であるから，次のようにして証明できる。

$$\begin{aligned} G_{N-m} &= \sum_{n=0}^{N-1} g_n e^{-\frac{2\pi i}{N} n(N-m)} = \sum_{n=0}^{N-1} g_n e^{-2\pi i n + \frac{2\pi i}{N} nm} = \sum_{n=0}^{N-1} g_n e^{\frac{2\pi i}{N} nm} \\ &= \sum_{n=0}^{N-1} \overline{g_n} \overline{e^{-\frac{2\pi i}{N} nm}} = \overline{\sum_{n=0}^{N-1} g_n e^{-\frac{2\pi i}{N} nm}} = \overline{G_m} \end{aligned}$$

## DFT の例

DFT の具体例を示す。はじめに，ある整数 $k$ $(0 \leq k \leq N-1)$ に対して,

$$g_n = \frac{1}{N} e^{\frac{2\pi i}{N} kn}$$

を考える (複素数を使った三角関数)。$m \neq k$ のときは,

$$\begin{aligned} G_m &= \sum_{n=0}^{N-1} \frac{1}{N} e^{\frac{2\pi i}{N}kn} e^{-\frac{2\pi i}{N}mn} = \frac{1}{N} \sum_{n=0}^{N-1} e^{\frac{2\pi i}{N}(k-m)n} = \frac{1}{N} \frac{1 - e^{\frac{2\pi i(k-m)N}{N}}}{1 - e^{\frac{2\pi i(k-m)}{N}}} \\ &= \frac{1}{N} \frac{1-1}{1 - e^{\frac{2\pi i(k-m)}{N}}} = 0 \end{aligned}$$

となり，$m = k$ のときは,

$$G_m = \sum_{m=0}^{N-1} \frac{1}{N} e^{\frac{2\pi i}{N}kn} e^{-\frac{2\pi i}{N}mn} = \frac{1}{N} \sum_{m=0}^{N-1} 1 = 1$$

となる。従って,

$$G_m = \begin{cases} 1 & (m = k) \\ 0 & (m \neq k) \end{cases}$$

となる。

次に，通常の三角関数の変換を考える。オイラーの公式を使って，上の結果から簡単に導くことができる。

$$\frac{1}{N} \cos \frac{2\pi}{N} kn = \frac{1}{N} \frac{e^{\frac{2\pi i}{N}kn} + e^{-\frac{2\pi i}{N}kn}}{2} = \frac{1}{N} \frac{e^{\frac{2\pi i}{N}kn} + e^{\frac{2\pi i}{N}(N-k)n}}{2}$$

であるから，$\frac{1}{N} \cos \frac{2\pi i}{N} kn$ の DFT は,

$$G_m = \begin{cases} \frac{1}{2} & (m = k) \\ \frac{1}{2} & (m = N - k) \\ 0 & (m \neq k) \end{cases}$$

となる。同様に，$\frac{1}{N} \sin \frac{2\pi i}{N} kn$ の DFT は,

$$\frac{1}{N} \sin \frac{2\pi}{N} kn = \frac{1}{N} \frac{e^{\frac{2\pi i}{N}kn} - e^{-\frac{2\pi i}{N}kn}}{2i} = \frac{1}{N} \frac{e^{\frac{2\pi i}{N}kn} - e^{\frac{2\pi i}{N}(N-k)n}}{2i}$$

より,

$$G_m = \begin{cases} \frac{1}{2i} & (m = k) \\ -\frac{1}{2i} & (m = N - k) \\ 0 & (m \neq k) \end{cases}$$

となる。このようにして，DFT によって，原信号の周波数成分を取り出すことができることがわかる。

## 2.2 DFT の畳み込み

フーリエ級数展開やフーリエ変換で，2 つの関数を変換したものの積は，もとの 2 つの関数の畳み込みをしたものを変換したものになることを，線形システムの講義で示した。DFT でも同様なことが成立する。ここで，$g_n$ と $h_n$ を DFT したものを，$G_m$ と $H_m$ とする。上では $g_n$ は，$0 \leq n \leq N-1$ で定義されていると考えたが，ここでは，$g_n$ や $h_n$ は全ての整数の集合上で定義された周期 $N$ の周期関数と考える。このとき，畳み込みした $d_n$ を次のように定義する。

$$d_n = \sum_{k=0}^{N-1} g_k h_{n-k} = \sum_{k=0}^{N-1} g_{n-k} h_k$$

(周期関数を仮定しているので，2 番目と 3 番目の式は同じものになる。)

この関係を，$d_n$ を DFT した $D_m$ が，$G_m H_m$ になることを示すことによって証明する。

$$D_m = \sum_{n=0}^{N-1} d_n e^{-\frac{2\pi i}{N}nm} = \sum_{n=0}^{N-1} \sum_{k=0}^{N-1} g_{n-k} h_k e^{-\frac{2\pi i}{N}nm}$$

となる。ここで，$k$ と $n$ の和の順番を交換したあと，$l = n - k$ とおき，$n$ に関する和を，$l$ に関する和に書き換える。

$$D_m = \sum_{k=0}^{N-1} \sum_{n=0}^{N-1} g_{n-k} h_k e^{-\frac{2\pi i}{N} nm} = \sum_{k=0}^{N-1} \sum_{l=-k}^{N-1-k} g_l h_k e^{-\frac{2\pi i}{N}(l+k)m}$$

さらに，$g_l$ も，$l$ に関して周期 $N$ の周期関数であるため，$g_l h_k e^{-\frac{2\pi i}{N}(l+k)m}$ は，$l$ に $l+N$ を代入しても値が変わらない。従って，$l$ に関する $-k$ から $N-1-k$ までの和を，0 から $N-1$ までの和とすることができる。以上より，

$$\begin{aligned} D_m &= \sum_{k=0}^{N-1} \sum_{l=0}^{N-1} g_l h_k e^{-\frac{2\pi i}{N}(l+k)m} \\ &= \left( \sum_{l=0}^{N-1} g_l e^{-\frac{2\pi i}{N} lm} \right) \left( \sum_{k=0}^{N-1} h_k e^{-\frac{2\pi i}{N} km} \right) \\ &= G_m H_m \end{aligned}$$

となり，DFT したものの積になることがわかる。

DFT の場合，周期関数としての畳み込みが得られるが，このような畳み込みを巡回型の畳み込みと呼ぶ。たとえば，$g_m$ をシステムへの入力と考え，$d_n$ をその出力と考えれば，$h_n$ はシステムのインパルス応答を表している。この場合インパルス $\delta_n$ は，

$$\delta_n = \begin{cases} 1 & (n = 0) \\ 0 & (n \neq 0) \end{cases}$$

である。このインパルスの定義で，$n = 0$ だけで 1 と書いてあるが，巡回型の畳み込みを考えているため，実際には $n = 0, \pm N, \pm 2N, \ldots$ において 1 になる。

## 2.3 パワースペクトル

DFT の結果 $G_m$ は周波数成分を表すが，その値は複素数であるため，

$$G_m = |G_m| e^{i\phi}$$

と書くことができる。このとき，絶対値 $|G_m|$ がその周波数成分の大きさを表し，$\phi$ がその周波数での位相を表すことになる。また，$|G_m|^2$ をパワースペクトルと呼ぶ。

$g_{-n} = g_{N-n}$ を DFT すると，$\overline{G_m}$ が得られる。パワースペクトル $|G_m|^2$ を IDFT したものを $h_l$ とおく。$|G_m|^2 = G_m \overline{G_m}$ であり，DFT したものの積は畳み込みしたものを DFT したものであることから，$h_l$ は $g_l$ と $g_{N-l}$ を畳み込みしたものになる。従って，

$$h_l = \sum_{k=0}^{N-1} g_k g_{N-(l-k)} = \sum_{k=0}^{N-1} g_k g_{k-l}$$

となる。左辺は $g_n$ を $l$ だけずらしたものと，もとの $g_n$ と乗算して，0 から $N-1$ までたし合わせたものである。このようなものを，自己相関関数呼ぶ。すなわち，自己相関関数を DFT したものが，パワースペクトルとなる。

信号が確率分布することを考える。すわなち，確率的な構造は同じであるが，様々な信号が現れる場合である。このような場合，この確率によるパワースペクトルの平均を考えることができる。これは，自己相関関数の平均を DFT したものに一致する。**実際の信号処理装置はこのような確率的な構造を使って設計する場合が多い。**例えば，パワースペクトル (自己相関関数) の平均が与えられるものとして，それを使って処理装置を設計する。

## 2.4　高速フーリエ変換

$N$ 点の値からなるデータの DFT を計算するための計算量を考える。$N$ 個の $G_m$ を計算するために，それぞれ，およそ $N$ 回の複素数の乗算と加算が必要であるので，DFT を直接計算する計算量は $N^2$ のオーダーとなる。

ソートの場合と同様に，この計算量はデータ数が多くなると問題となる。計算量を削減する方法として，高速フーリエ変換 (fast Fourier transform, FFT) が開発されている。考え方としては，選択ソートに対するマージソートと同じであり，問題を半分ずつに分割していく方法である。

データ点数が 2 のべき乗であるときを考える。すなわち，$N = 2^M$ となる場合である。このとき，FFT の計算量は $N \log_2 N$ になる。その原理を簡単に説明する。

まず，入力データのインデックスを奇数の場合と偶数の場合に分けることを考える。

$$
\begin{aligned}
G_m &= \sum_{n=0}^{N-1} g_m e^{-\frac{-2\pi i}{N}mn} \\
&= \sum_{l=0}^{N/2-1} g_{2l} e^{-\frac{2\pi i}{N}m(2l)} + \sum_{l=0}^{N/2-1} g_{2l+1} e^{-\frac{2\pi i}{N}m(2l+1)} \\
&= \sum_{l=0}^{N/2-1} g_{2l} e^{-\frac{2\pi i}{N}m(2l)} + e^{\frac{-2\pi i}{N}m} \sum_{l=0}^{N/2-1} g_{2l+1} e^{\frac{2\pi i}{N}m(2l)} \\
&= \sum_{l=0}^{N/2-1} g_{2l} e^{-\frac{2\pi i}{N/2}ml} + e^{-\frac{2\pi i}{N}m} \sum_{l=0}^{N/2-1} g_{2l+1} e^{-\frac{2\pi i}{N/2}ml}
\end{aligned}
$$

と書くことができる。この式は，DFT の結果 $G_m$ をすべて求めるためには，インデックスが偶数のデータに対する DFT と奇数のデータに対する DFT を計算した後，$N$ 回程度の乗算と加算で計算できることを示している。すなわち，$N/2$ 点のデータの DFT が 2 回と，$N$ のオーダの計算量が必要になる。同様に，$N/2$ 点の DFT の計算には，$N/4$ 点のデータの DFT が 2 回と，$N/2$ のオーダの計算量が必要になる。また，データ数が 1 である DFT はデータの値そのものが DFT の結果であるため，計算量は 0 である。従って，この分割を繰り返せば，

$$
\begin{aligned}
& (N \text{ 点の DFT の計算量}) \\
= \;& 2 \times (N/2 \text{ 点の DFT の計算量}) + O(N) \\
= \;& 4 \times (N/4 \text{ 点の DFT の計算量}) + 2 \times O(N/2) + O(N) \\
= \;& \cdots \\
= \;& (N/2) \times (2 \text{ 点の DFT の計算量}) + N/4 \times O(4) + \cdots + 2 \times O(N/2) + O(N) \\
= \;& N \times (1 \text{ 点の DFT の計算量}) + N/2 \times O(2) + N/4 \times O(4) + \cdots + 2 \times O(N/2) + O(N) \\
= \;& O(NM) = O(N \log_2 N)
\end{aligned}
$$

となり，高速に DFT が計算できることがわかる。

## 2.5　2 次元フーリエ変換

今まで，フーリエ変換で扱う関数は時間軸上で定義されているものとして扱ってきたが，もっと一般の量を扱うことができる。たとえば，$g(x)$ で，座標 $x$ における温度を表すとすれば，熱の分布を扱うことができる (フーリエ級数展開は，もともとは線状の金属における熱の伝導を解析するために，フーリエ先生が考え出したものである)。

画像を扱うためには，2 次元が定義域の関数を考える。すなわち，点 $(x, y)$ における明るさを $f(x, y)$ で表す。カラー画像を考える場合は，光の 3 原色である赤，緑，青（RGB) の画素値を考えることになるが，ここでは単純に白黒画像を考え，明るさだけを考えることにする。

さらに，コンピュータで扱うので，明るさは離散点で定義されることになる。画像の画素 $(m, n)$ でのこの明るさの値が，$g_{m,n}$ $(0 \leq m \leq M-1, 0 \leq n \leq N-1)$ で表されているものとする。慣例に従い，1 番目のインデックスが横方向，2 番目のインデックスが縦方向を表すものとする。すなわち，横 $M\times$ 縦 $N$ 画素の画像を考えていることになる。

2 次元の DFT は，2 次元データの縦と横をそれぞれに対して変換する形になる。$g_{m,n}$ を DFT したものを $G_{k,l}$ とおけば，

$$G_{k,l} = \sum_{m=0}^{M-1}\sum_{n=0}^{N-1} g_{m,n} e^{-\frac{2\pi i}{M}km - \frac{2\pi i}{N}ln}$$

となる。IDFT も同様に，

$$g_{m,n} = \frac{1}{MN}\sum_{k=0}^{M-1}\sum_{l=0}^{N-1} g_{m,n} e^{\frac{2\pi i}{M}km + \frac{2\pi i}{N}ln}$$

によって表される。

$g_{m,n}$ と $h_{m,n}$ を DFT したものを，それぞれ，$G_{k,l}$ と $H_{k,l}$ する。このとき，$G_{k,l}H_{k,l}$ は，$g_{m,n}$ と $h_{m,n}$ の畳み込み積分 (横の周期が $M$，縦の周期が $N$ の周期関数を考えている)

$$d_{m,n} = \sum_{k=0}^{M-1}\sum_{l=0}^{N-1} g_{k,l}h_{m-k,n-l} = \sum_{k=0}^{M-1}\sum_{l=0}^{N-1} g_{m-k,n-l}h_{k,l}$$

を，DFT したものになる。

## 2.6 ウィーナーフィルタ

最も基本的な画像観測のモデルを考える。まず，原画像 $f_{m,n}$ があり，それがブレたボケなどの変換を受け，雑音が加算されたものが観測される。このブレやボケを画像の劣化と呼ぶことにする。

劣化は，画像の位置によらず一定であるものとする。そのため，劣化を 2 次元インパルス応答を $a_{m,n}$ で表すことができる。このインパルス応答を，画像処理の分野では，Point Spread Function (点広がり関数) と呼ぶ。すなわち，原画像が原点だけで値が 1 で，それ以外では 0 の場合，$a_{m,n}$ が劣化画像となる。原画像が一般の画像 $f_{m,n}$ の場合，$a_{m,n}$ と $f_{m,n}$ の畳み込みが劣化画像となる。

この劣化画像に雑音 $n_{m,n}$ が加算されたものが，観測画像 $g_{m,n}$ になる。従って，画像の観測モデルは，

$$g_{m,n} = \sum_{k=0}^{M-1}\sum_{l=0}^{N-1} a_{k,l}f_{m-k,n-l} + n_{m,n}$$

で与えられる。ここでは，DFT を使うために，画像，劣化のインパルス応答，雑音ともに，周期が横 $M$，縦 $N$ の周期関数であるものとする。

$f_{m,j}$，$a_{m,n}$，$n_{m,n}$，$g_{m,n}$ を DFT したものを，$F_{i,j}$，$A_{i,j}$，$N_{i,j}$，$G_{i,j}$ で表す。このとき，

$$G_{i,j} = A_{i,j}F_{i,j} + N_{i,j}$$

が成立する。

この観測画像から復元画像を推定する問題を考える。ここでは，復元のためのインパルス応答 $b_{m,}$ を観測画像と畳み込むことによって，復元画像を得ることを考える。従って，復元画像 $\hat{f}_{m,n}$ は，

$$\hat{f}_{m,n} = \sum_{k=0}^{M-1}\sum_{l=0}^{N-1} b_{k,l}g_{m-k,n-l}$$

となる。$b_{m,n}$ と $\hat{f}_{m,n}$ を DFT したものを，$B_{i,j}$，$\hat{F}_{i,j}$ で表せば，

$$\hat{F}_{i,j} = B_{i,j}G_{i,j}$$

となる。

この $b_{m,n}$ を具体的に決める必要がある。本実験で用いるウィーナーフィルタは，平均 2 乗誤差を最小するものとして定義される。いま，原画像と雑音に関して，パワースペクトルが分かっているものとする。それぞれのパワースペクトルを $U_{i,j}$ と $V_{i,j}$ とすれば，ウィーナーフィルタを DFT を使って表現した $B_{i,j}$ は，

$$B_{i,j}=\frac{U_{i,j}\overline{A_{i,j}}}{A_{i,j}U_{i,j}\overline{A_{i,j}}+V_{i,j}}$$

となる。

実際の画像への適用することを考える。ピンボケ画像は，原画像の 1 点が円状に広がることになる。その半径を $r$ とし，原点を中心とした半径 $r$ の円内の格子点の数を $L$ とすれば，ボケのインパルス応答は $a_{i,j}$ は，

$$a_{i,j}=\begin{cases}\frac{1}{L} & (i^2+j^2\leq r^2)\\ 0 & \text{(else)}\end{cases} \tag{1}$$

となる。実際には，円周部分を線形補間し，滑らかにした関数を用いる。

画像は，本来は周期関数ではない。ただ，DFT を使うためには周期関数でなければいけない。そのため，画像の領域を拡張して大きな領域で DFT を行う。拡張した部分の画素の値を決めなくてはいけない。そのための最も簡単な方法は，画像の周辺の値を線形補間するものである。

# 3 実習内容

## 3.1 前試問

- DFT のプログラムを作成し，そのプログラムを説明する。

  FFT のプログラムは複雑なため，単なる DFT のプログラムで良い。具体的には，DFT を計算するために，複素数を計算するクラス `Complex` と，クラス `Complex` の 1 次元配列を入力として，DFT を計算するプログラムを作成する。

  クラス `Complex`，FFT を実行するクラス `FFT`，グラフを書く部分を除いたメインプログラムのためのクラス `MainWithouGraph` のファイルは，事前に Web サイトで配布するが，クラス `Complex` のプログラムは，できるだけ各自作成すること。クラス `Complex` とクラス `DFT` の仕様は以下の通りである。

**クラス `Complex` の仕様**

- フィールド
  - `double re`
    複素数の実部を格納する。
  - `double im`
    複素数の虚部を格納する。
  - `static final double EPS`
    絶対値がこれ以下のとき，0 と判断する値。
- コンストラクター
  - `Complex(double re, double im)`
    実部を `re`，虚部を `im` とする `Complex` のオブジェクトを作成する。
- メソッド

- `boolean equals(Complex c)`
  自身と c の実部，虚部の差の絶対値が両方とも EPS より小さければ，true を返し，そうでなければ false を返す。
- `double real()`
  自身の実部 re を返す。
- `double imag()`
  自身の虚部 im を返す。
- `Complex copy()`
  自身と同じ値を持つ Complex のオブジェクトを作成して，その参照情報を返す。
- `Complex set(double re, double im)`
  自身の実部に re を虚部に im をセットして，自身の参照情報を返す。
- `Complex set(Complex c)`
  自身に Complex のオブジェクト c の値をセットして，自身の参照情報を返す。
- `Complex setConj(Complex c)`
  自身に，自身の複素共役をセットして，自身の参照情報を返す。
- `Complex setZero()`
  自身の実部と虚部に 0 をセットして，自身の参照情報を返す。
- `boolean isZero()`
  自身の実部と虚部の絶対値が両方とも EPS より小さければ，true を返し，そうでなければ false を返す。
- `Complex addTo(Complex c)`
  自身に，自身と Complex のオブジェクト c の値を加算したものをセットして，自身の参照情報を返す。
- `Complex timesBy(Complex c)`
  自身に，自身と Complex のオブジェクト c の値を乗算したものをセットして，自身の参照情報を返す。
- `Complex divideBy(Complex c)`
  自身に，自身を Complex のオブジェクト c の値で割ったものをセットして，自身の参照情報を返す。
- `Complex negate()`
  自身に，自身の正負の符号を反転したものをセットして，自身の参照情報を返す。
- `Complex div(int n)`
  自身に，自身を int 型変数 n に格納されている整数値で割ったものをセットして，自身の参照情報を返す。

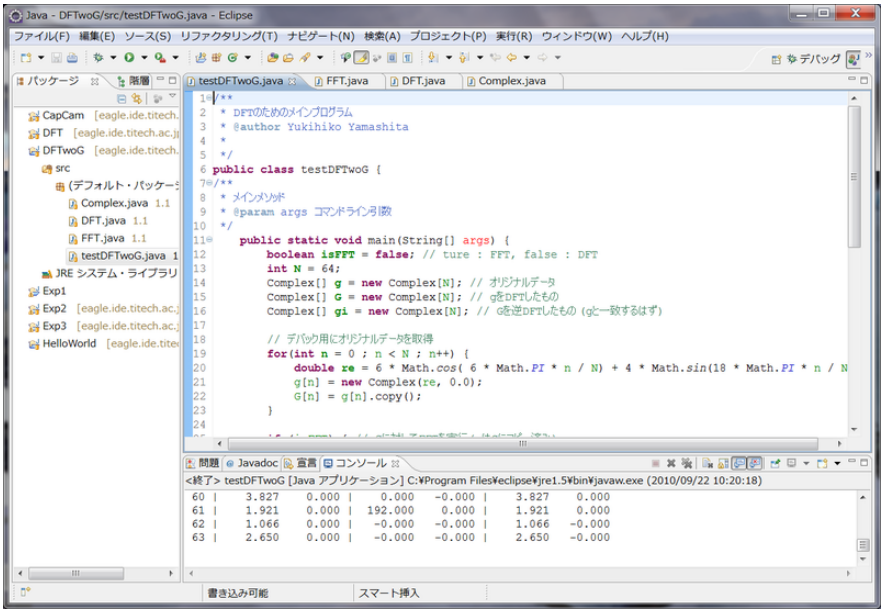
### クラス DFT の仕様

- フィールド
  - `int N`
    処理するデータ数
  - `Complex[] wTbl`
    高速化のための三角関数値のテーブル (高速化のために，DFT を実行する前に，必要な三角関数の値を計算し，配列に格納しておく)。三角関数のテーブルの利用が難しければ，利用しなくてよい。

- コンストラクタ
  - `DFT(int N)`
    要素数 N の DFT を計算するためのオブジェクトを作成する。
    フィールド N の設定と，三角関数値のテーブルの作成を行う。
- メソッド
  - `void dft(Complex[] data)`
    data の DFT を計算して `data` に格納する (メソッド内部に，長さ N の作業用配列を作成し，そこに DFT を計算したものを格納し，それを最後に `data` に書き戻す)。
  - `void idft(Complex[] data)`
    data の逆 DFT を計算して，`false` ならば，data の DFT を `data` に格納する

## 3.2 実習内容 (１日目)

開発環境として，eclipse を用いる (計算機室での eclipse の起動方法については当日説明する)。




最初の「連絡事項」に書いた通り，eclipse の基本的な使い方を見ておくこと。eclipse の表示として，主に Java パースペクティブを用いる。Java パースペクティブの中で，次の 3 つのウィンドウを表示する。

まず，左側にプロジェクトを管理する「パッケージ・エクスプローラ」を表示する。そして，右 (と中央) 上側にプログラムを編集するためのウィンドウを表示する。表示されてないときは，パッケージ・エクスプローラで，プログラムのファイルをダブルクリックすると，この場所にプログラムを編集するためのウィンドウが開く。右 (と中央) 下側に，エラーの場所などを表示する「問題」，あるいは `System.out.println()` の出力を表示する「コンソール」のウィンドウを表示する (両者は自動的に切り替わるが，そのウィンドウのタブを使って変更可能である)。

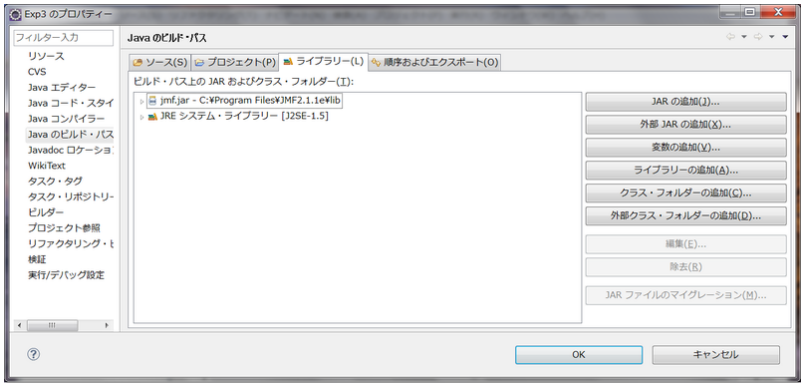
それ以外のウィンドウは，ウィンドウ右上の「最小化」のボタンをクリックして，非表示にする (メニューやツールボックスはそのまま)。

eclipse の機能でデバッグを行うと，Debug パースペクティブになる。デバッグの機能の使い方が分からない場合は，変数の値を表示する `System.out.println()` をプログラムの各所に配置し，変数の値の推移を調べてデバッグを行う。

- eclipse 上に，新しい Java プロジェクト「DFT」を作成する。「ファイル」→「新規作成」で，Java プロジェクトを選択し，名前を DFT として作成する。
- グラフを書くためのライブラリーを追加する。「パッケージ・エクスプローラ」のプロジェクト `DFT` の上で右クリックし,「プロパティ」をクリックする。現れたウィンドウの「Java のビルド・パス」をクリック,「ライブラリー」のタブをクリックして,「外部 JAR の追加」をクリックすれば，追加することができる。追加するライブラリーは,

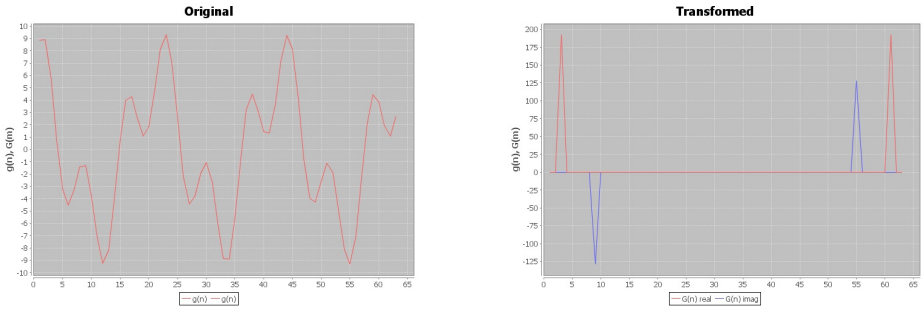
  `C:¥Experiment¥jar¥jcommon-1.0.16.jar`
  `C:¥Experiment¥jar¥jfreechart-1.0.13.jar`

  の 2 つである。




- 各自作成してきたプログラム，`DFT.java` と `Complex.java` をコピーする。コピーしたいファイルを,「パッケージ・エクスプローラ」のプロジェクト `DFT` の `src` に，ドラッグすれば，コピーが行われる。同様に,

  `C:¥Experiment¥DFT¥FFT.java`
  `C:¥Experiment¥DFT¥Plot.java`
  `C:¥Experiment¥DFT¥Main.java`
  をコピーする。
- DFT を実行し，その結果を記録せよ。




結果の例

- **データ数を変化させ**，`DFT` と `FFT` のプログラムの速度を比較せよ (データ数はかなり大きくしてみること)。

時間を測定するためのプログラムを次に示す。

ソースコード 1: 時間測定

```
double start = System.currentTimeMillis();

// 時間を測定したい処理

double end = System.currentTimeMillis();
System.out.println("Execute time = "
                + 0.001 * (end - start) + "[sec]");
```

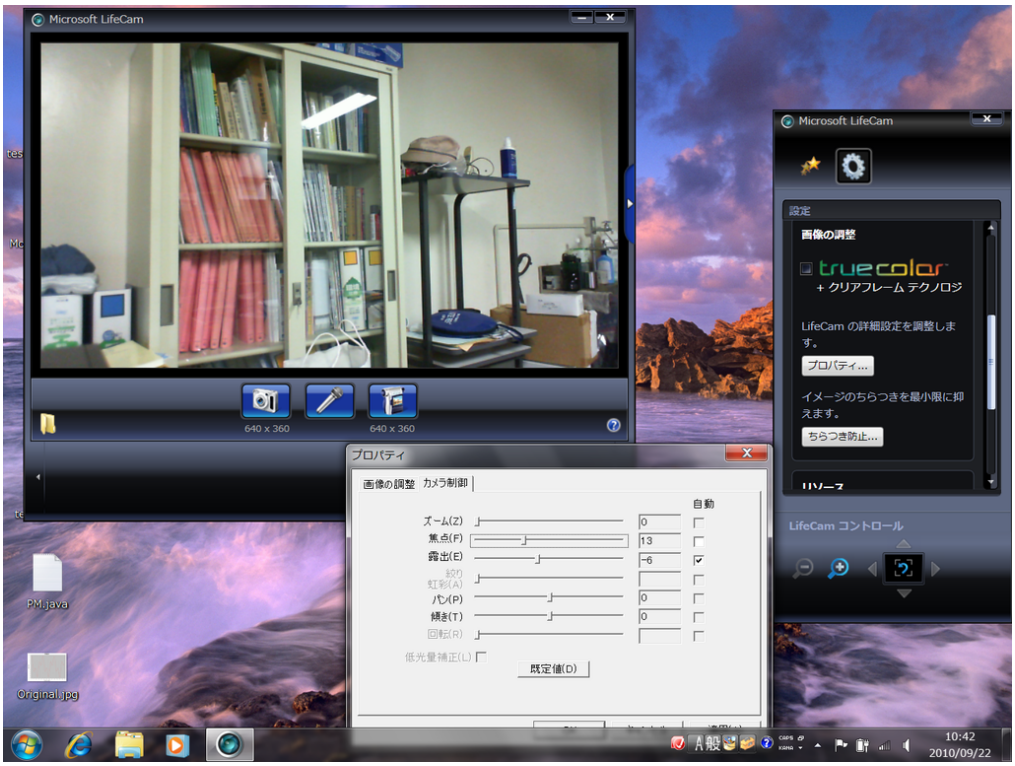
時間を計測する場合は，上記 2 つの関数の間に表示などの直接関係ない処理を挟まないようにすること。

- eclipse で，プロジェクト DFT をコピーして，畳み込み演算のためのプロジェクト Conv を作成する。
  プロジェクト DFT を選択して，メニューの「編集」→「コピー」をクリックする。そして，メニューの「編集」→「貼り付け」をクリックする (右クリックを使っても操作可能である)。
- 畳み込み積分を実行するクラス Conv をファイル Conv.java に作成する。
  プロジェクト Conv の src の上で右クリックし，メニューの「新規」→「クラス」をクリックする。メニュー上で名前 Conv を代入して「完了」をクリックする。Conv の仕様を下に示す。
- クラス Main のメソッド main() を次のように書き換える。畳み込む 2 つのデータを作成し (例えば，ある一定間隔の 1 になり他では 0 である信号と，指数関数 $e^{-an}$)，その畳み込み演算をメソッド directConv() および fftConv() を使って計算し，それぞれをグラフに表示し，記録する。

**クラス Conv の仕様**

- フィールド
  - int N
    処理するデータ数
  - Complex[] d1F, d2F
    受け取ったデータ df1 と df2 を FFT したものを格納する。
- コンストラクタ
  - Conv(int N)
    要素数 N の畳み込みを計算するためのオブジェクトを作成する。
    フィールド N の設定と，配列 d1F, d2F のための領域確保を行う。
- メソッド
  - void directConv(double[] d1, double[] d2, double[] d2, double[] result)
    d1 と d2 の (周期関数を仮定した) 畳み込みを定義式から直接計算し，result に格納する。
  - void fftConv(double[] d1, double[] d2, double[] d2, double[] result)
    d1 と d2 の (周期関数を仮定した) 畳み込みを計算し，result に格納する。計算は，d1 と d2 を d1F と d2F に格納し，DFT 係数を FFT を使って求め，その積を逆 FFT して実数部を result に格納する。

## 3.3 実習内容 (２日目)




- １日目の課題が終わっていない場合は，引き続きその課題を行う。
- 画像処理の実験を行う。
- Web カメラを接続して,「スタート」→「すべてのプログラム」などから「LifeCam」を実行する。LifeCam の設定ウィンドウが表示されない場合は，右端中央の右矢印をクリックする。設定ウィンドウで，解像度は 640×360 とし,「truecolor」のチェックを外し,「プロパティ」をクリックする。出てきたウィンドウの「カメラ制御」タブをクリックし，焦点の「自動」のチェックを外し，スライドバーを使って，わざと少し焦点が合っていない画像を取得するようにする。
  Lifecam のプログラムを修了する。
- eclipse を立ち上げ，新規プロジェクト `Wiener` を作成し，外部 JAR として
  `C:¥Program Files¥JMF2.1.1e¥lib¥jmf.jar`
  を追加する。また，ソースプログラムとして,「パッケージ・エクスプローラ」の `Wiener` の `src` に,
  `C:¥Experiment¥DFT¥Complex.java`
  `C:¥Experiment¥DFT¥FFT.java`
  `C:¥Experiment¥Wiener¥FFT2.java` (2 次元 FFT)
  `C:¥Experiment¥Wiener¥ImagePanel.java`
  (画像の表示と動作制御を行うプログラム)
  `C:¥Experiment¥Wiener¥ImagegProc.java`
  (ウィーナフィルタの計算などを行うプログラム)
  `C:¥Experiment¥Wiener¥Main.java`
  をコピーする。
- `Main.java` を開いて,「実行」をクリックすると，カメラから画像取得し，白黒画像を生成し，ウィーナーフィルタで処理するプログラムが立ち上がる。

このプログラムにおいて,「画像取得」をクリックすると，画像を取得し，白黒画像に変換する。「画像処理」をクリックすると，ウィーナフィルタによる処理を実行する。

「画像保存」をクリックすると，現在表示されている画像を JPEG ファイルとして保存する。また,「表示：」の右側の 3 つのボタン「カメラ画像」,「取得画像」,「処理済画像」のボタンをクリックすると，それぞれ，現在のカメラの画像，取得した白黒画像，ウィーナーフィルタで処理した画像を表示する。「ボケ半径：」右横のテキストボックスに数値を入力すると，ウィーナフィルタで使われるボケを表す式 (1 ) における $r$ の値 (ボケの半径) を設定できる。

- 画像を取得したあと，ボケ半径の値を変え，ウィーナーフィルタを使って処理を行い，画像が程度復元可能か調べよ。また，LifeCam の設定で画像のボケの度合いを変えて，同様に調べよ。
- 時間があれば，プロジェクト `Wiener` のソースプログラムを読んで (このソースプログラムも事前配布する)，画像や雑音のパワースペクトルを変更したり，画像に対して，2 次元畳み込みを行い表示するプログラムなどを作成せよ。

## 3.4 後試問

- 15:30 ごろにプログラムを作成する課題を出すので，そのプログラムを作成して，結果を教員または TA に見せる。

## 3.5 レポートに書く内容

- DFT の実験結果と，それに関して考察したことを述べよ。
- FFT の実験結果と，それに関して考察したことを述べよ。
- 畳み込み演算と，それに関して考察したことを述べよ。
- ウィーナーフィルタの実験結果と，それに関して考察したことを述べよ。