Table of Contents

FFT(Fast Fourier Transformation)

file:/Users/bob/Github/TeamNishitani/jupyter_num_calc/fft
https://github.com/daddygongon/jupyter_num_calc/tree/master/notebooks_p

直接積分によるフーリエ係数

対象とする関数をまず作る.

piecewise関数は階段関数で，振る舞いはコメント(#)を適当に外して確認せよ． 初期設定.

In [1]:
```
%matplotlib inline
import sympy as sym
import matplotlib.pyplot as plt
import numpy as np

def func(x):
    return (1)*sym.Piecewise((-1, (x<1/2).any() ), (1, True)) # step func
#    return 1
    # #F:=x->piecewise(x=0,1/2,x>0,x);
  #F:=x->piecewise(x<1/2,x,x>=1/2,1-x);
  #F:=x->piecewise(x<1/2,-1,x>1/2,1);
  #  Piecewise(x, [x < 1/2, x >= 1/2], [-1, 1])
  #F:=x->piecewise(x>0 and x<1/2,-1,x>1/2,1);
  #F:=x->x-1/2;

xdata = range(0, 1, 256)
```

```
plt.plot(xdata, func(xdata), 'r-', label='data')
plt.grid()
plt.show()
```

```
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
<ipython-input-1-3fe72b92062d> in <module>
     16 xdata = range(0, 1, 256)
     17
---> 18 plt.plot(xdata, func(xdata), 'r-', label='data')
     19 plt.grid()
     20 plt.show()

<ipython-input-1-3fe72b92062d> in func(x)
      5
      6 def func(x):
----> 7     return (1)*sym.Piecewise((-1, (x<1/2).any() ), (1, True)) # step func
      8 #    return 1
      9     # #F:=x->piecewise(x=0,1/2,x>0,x);

TypeError: '<' not supported between instances of 'range' and 'float'
```

上の通り，積分をPiecewise関数に施すのは難しそう．sympy.Piecewiseではarrayに対応してないし，一方でnumpyはintegrateが数値積分．

ここでは，Mapleの結果を残しておく．

```
maplerestart;
F:=x->piecewise(x=0,1/2,x>0,x);
#F:=x->piecewise(x<1/2,x,x>=1/2,1-x);
#F:=x->piecewise(x<1/2,-1,x>1/2,1);
F:=x->piecewise(x<1/2,-1,x>=1/2,1);
#F:=x->piecewise(x>0 and x<1/2,-1,x>1/2,1);
#F:=x->x-1/2;
plot([F(x)],x=0..1);
F := proc (x) options operator, arrow; piecewise(x = 0, 1/2, 0

   < x, x) end proc


 F := proc (x) options operator, arrow; piecewise(x < 1/2, -1,

    1/2 <= x, 1) end proc



KK:=3;
N:=2^KK;L:=1-0;
                              KK := 3

                               N := 8

                               L := 1
```

直接積分によるフーリエ級数

```
maple
2*Pi*1/L*x;
```

$$2\ Pi\ x$$

```
maple
> int(F(x)*cos(2*Pi*1/L*x),x=0..L);
```

$$0$$

```
maple
> for n from 0 to N do
    a[n]:=2/L*int(F(x)*cos(2*Pi*n/L*x),x=0..L);
  end do;
```

$$a_0 := 0$$
$$a_1 := 0$$
$$a_2 := 0$$
$$a_3 := 0$$
$$a_4 := 0$$
$$a_5 := 0$$
$$a_6 := 0$$
$$a_7 := 0$$
$$a_8 := 0$$

```
maple
> for n from 0 to N do
    b[n]:=2/L*int(F(x)*sin(2*Pi*n/L*x),x=0..L);
  end do;
```

$$b_0 := 0$$
$$b_1 := \frac{4}{\pi}$$
$$b_2 := 0$$
$$b_3 := \frac{4}{3\pi}$$
$$b_4 := 0$$
$$b_5 := \frac{4}{5\pi}$$
$$b_6 := 0$$
$$b_7 := \frac{4}{7\pi}$$
$$b_8 := 0$$

ここで，オイラーの関係

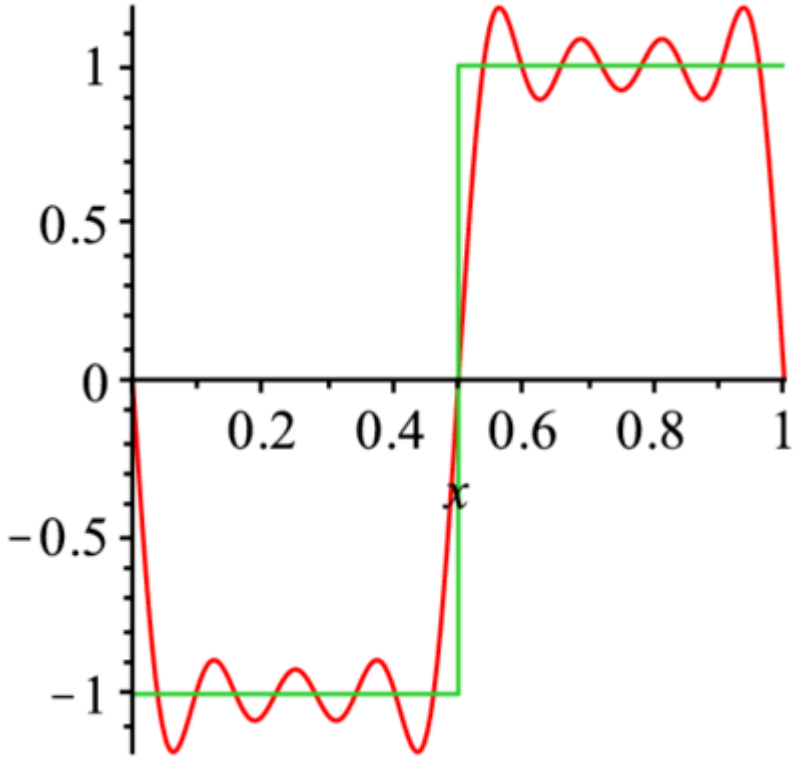
$$a[n] = c[n] + c[-n],\ b[n] = I(c[n] - c[-n])$$
$$c[-n] = \frac{1}{2}(a[n] + b[n]),\ c[n] = \frac{1}{2}(a[n] - Ib[n])$$

を使って，三角関数系からexpへ変換する.

```
maple
> for n from 0 to N do c[n]:=1/L*int(F(x)*exp(-
I*2*Pi*n/L*x),x=0..L); end do;
> for n from 1 to N do c[-
n]:=1/L*int(F(x)*exp(I*2*Pi*n/L*x),x=0..L); end do;
```

$$c_0 := 0$$
$$c_1 := \frac{2I}{\pi}$$
$$c_2 := 0$$
$$c_3 := \frac{2I}{3\pi}$$
$$c_4 := 0$$
$$c_5 := \frac{2I}{5\pi}$$
$$c_6 := 0$$
$$c_7 := \frac{2I}{7\pi}$$
$$c_8 := 0$$
$$c_{-1} := -\frac{2I}{\pi}$$
$$c_{-2} := 0$$
$$c_{-3} := -\frac{2I}{3\pi}$$
$$c_{-4} := 0$$
$$c_{-5} := -\frac{2I}{5\pi}$$
$$c_{-6} := 0$$
$$c_{-7} := -\frac{2I}{7\pi}$$
$$c_{-8} := 0$$

```
maple
> F1:=unapply(sum(evalf(c[i]*exp(I*2*Pi*i/L*x)),i=-(N-1)..(N-
1)),x):
> plot({Re(F1(x)),F(x)},x=0..1);
```




```
maple
> evalf(2/3/Pi);
```

$$0.2122065907$$

選点直交性による計算の簡素化

ところが，実際に積分していては，時間がかかりすぎる．直交関数系の選点直交性を使うとより簡単になる．

直交関数系の選点直交性

直交多項式は，

$$\varphi_n(x) = 0 \, at \, x_1, \, x_2, \, \cdots x_n$$

である．$n-1$以下の次数m, lでは，

$$\sum_{i=1}^{n} \phi_l(x_i) \, \varphi_m(x_i) = \delta_{ml} C_l$$

が成り立つ．これは，直交関係と違い積分でないことに注意．証明は略．

これを使えば，この先程の直交関数展開

$$F(x) = \sum_{l=1}^{N} a_l \varphi_l(x)$$

の両辺に$\varphi_m(x_i)$を掛けてiについて和をとれば，

$$\sum_{i=1}^{n} F(x_i)\,\phi_m(x_i) =$$

$$\sum_{i=1}^{n}\sum_{l=1}^{N} a_l \varphi_l(x_i)\,\varphi_m(x_i)$$

$$= \sum_{l=1}^{N} a_l \sum_{i=1}^{n} \varphi_l(x_i)\,\varphi_m(x_i)$$

$$= \sum_{l=1}^{N} a_l \delta_{ml} C_m = a_m C_m$$

となる．つまり，

$$a_m = \frac{1}{C_m}\sum_{i=1}^{n} F(x_i)\,\varphi_m(x_i)$$

となり，単純な関数の代入とかけ算で係数が決定される．

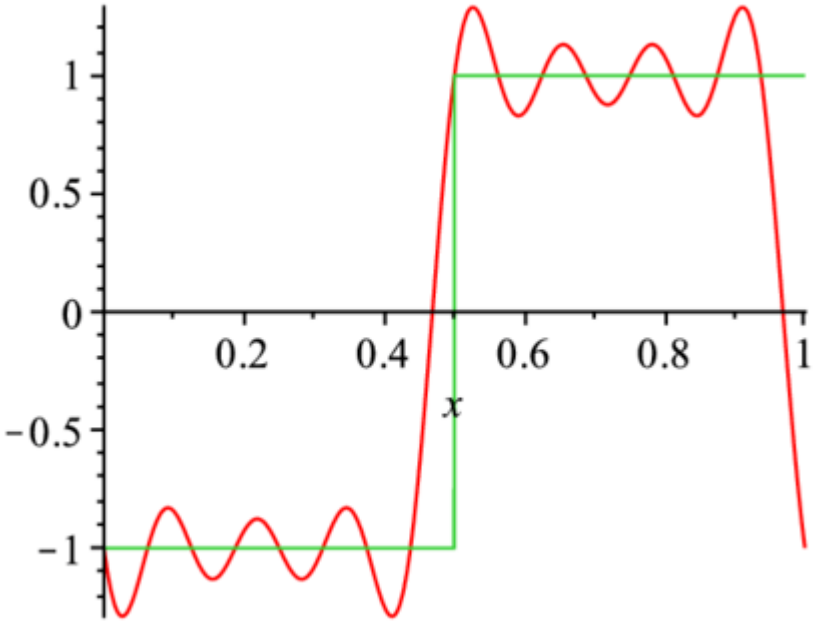
選点直交性を用いた結果

Pythonで扱うのを失敗しています．以下はMapleの出力．

```
maple
> KK:=4; N:=2^KK;L:=1-0;
> for k from 0 to N-1 do
    c[k]:=evalf(sum(F(i*L/N)*exp(-I*2*Pi*k*i/N),i=0..N-1));
  end do;

maple
c_0:=0.
c_1:=-2.000000000 + 10.05467898 I
c_2:=0.
c_3:=-2.000000000 + 2.993211524 I
c_4:=0.
c_5:=-2.000000001 + 1.336357276 I
c_6:=0.
c_7:=-2.000000001 + 0.3978247331 I
c_8:=0.
c_9:=-2.000000001 - 0.3978247331 I
c_10:=0.
c_11:=-2.000000001 - 1.336357276 I
c_12:=0.
c_13:=-2.000000000 - 2.993211524 I
c_14:=0.
c_15:=-2.000000000 - 10.05467898 I

maple
> F1:=unapply(sum(evalf(c[i]*exp(I*2*Pi*i/L*x)/N),i=0..(N/2-1))+
> sum(evalf(c[N-i]*exp(-I*2*Pi*i/L*x)/N),i=1..(N/2-1)),x):
> plot({Re(F1(x)),F(x)},x=0..1);
```



高速フーリエ変換アルゴリズムによる高速化

sin, cosとexp関数を結びつけるオイラーの関係を使うと,

と変換できる．これを使うと,

$$c_k = \frac{1}{C_m} \sum_{i=0}^{N-1} F(x_i) \exp\left(\frac{-2\pi I}{N}\right)$$

となる．$N = 8$の場合を実際に計算すると，$z = \exp(-\frac{2\pi}{8} I)$として，$z^8 = 1, z^9 = z, \cdots$を使うと,

$$\begin{bmatrix} c_0 \\ c_1 \\ c_2 \\ c_3 \\ c_4 \\ c_5 \\ c_6 \\ c_7 \end{bmatrix} = \begin{bmatrix} 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & z & z^2 & z^3 & z^4 & z^5 & z^6 & z^7 \\ 1 & z^2 & z^4 & z^6 & 1 & z^2 & z^4 & z^6 \\ 1 & z^3 & z^6 & z & z^4 & z^7 & z^2 & z^5 \\ 1 & z^4 & 1 & z^4 & 1 & z^4 & 1 & z^4 \\ 1 & z^5 & z^2 & z^7 & z^4 & z^1 & z^6 & z^3 \\ \vdots & & & & & & & \\ \vdots & & & & & & & \end{bmatrix} \begin{bmatrix} F_0 \\ F_1 \\ F_2 \\ F_3 \\ F_4 \\ F_5 \\ F_6 \\ F_7 \end{bmatrix}$$

となる．この行列計算を素直に実行すると，8×8=64回の演算が必要となる．これを減らせないかと考えたのが，高速フーリエ変換の始まりである． この行列をよく見ると同じ計算を重複しておこなっていることが分かる．そこで，行列の左側と右側で同じ計算をしている部分をまとめると,

$$\begin{bmatrix} c_0 \\ c_1 \\ c_2 \\ c_3 \\ c_4 \\ c_5 \\ c_6 \\ c_7 \end{bmatrix} = \begin{bmatrix} 1 & 1 & 1 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & z & z^2 & z^3 \\ 1 & z^2 & z^4 & z^6 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & z^3 & z^6 & z \\ 1 & z^4 & 1 & z^4 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & z^5 & z^2 & z^7 \\ 1 & z^6 & z^4 & z^2 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & z^7 & z^6 & z^5 \end{bmatrix} \begin{bmatrix} F_0 + F_4 \\ F_1 + F_5 \\ F_2 + F_6 \\ F_3 + F_7 \\ F_0 - F_4 \\ F_1 - F_5 \\ F_{[\,]} - F_6 \\ F_{[\,]} - F_7 \end{bmatrix}$$

とすることができる．ここで，$z^4 = -1$などを使っている．右側のベクトルの計算でロスするが，行列の中の計算の回数を半分に減らすことができる．再度できあがった行列を見れば，同じ計算をさらにまとめるこ とができそうである．こうして，次々と計算回数を減らしていくことが可能で，最終的に行列部分の計算が一切なくなる．残るのは，右側のベクトルの足し算引き算だけになる．

このベクトルの組み合わせは，一見相当複雑そうで，その条件分岐で時間がかかりそうに思われる．しかし，よく調べてみれば，単純なビット演算で処理することが可能であるこ とが判明した．こうして，2の整数乗のデータの組に対しては，極めて高速にフーリエ変換を実行することが可能となった． FFTでの演算回数は，データ数をNとすると

$$N \log_2 N$$

となる．単純な場合のN^2と比較すると，以下のようになり，どれだけ高速化されているかが理解されよう．

```
maple
> dN2:=[]; dFft:=[]; for i from 2 to 16 do N:=2^i; n2:=N*N;
Fft:=N/2*log[2](N);
> Fft/n2; printf("%10d %12d %12d
%10.5f\n",N,n2,Fft,evalf(Fft/n2));
> dN2:=[op(dN2),[N,n2]]; dFft:=[op(dFft),[N,Fft]]; end do:

maple
         4           16            4    0.25000
         8           64           12    0.18750
        16          256           32    0.12500
        32         1024           80    0.07812
        64         4096          192    0.04688
       128        16384          448    0.02734
       256        65536         1024    0.01562
       512       262144         2304    0.00879
      1024      1048576         5120    0.00488
      2048      4194304        11264    0.00269
      4096     16777216        24576    0.00146
      8192     67108864        53248    0.00079
     16384    268435456       114688    0.00043
     32768   1073741824       245760    0.00023
     65536   4294967296       524288    0.00012

maple
> with(plots):
```

```
> l1:=plot(dN2): l2:=plot(dFft):
> display(l1,l2);
```

FFT関数を用いた結果

```
maple
> KK:=4;
  N:=2^KK;i:='i';L:=1-0;
  x1:=array([evalf(seq(F(i/N),i=0..N-1))]);
  y1:=array([evalf(seq(0,i=0..N-1))]);
```

$$x1 := [-1.0 \quad -1.0 \quad -1.0 \quad -1.0 \quad -1.0 \quad -1.0 \quad -1.0 \quad -1.0 \quad 1.0 \quad 1.0 \quad 1.0 \quad 1.0$$

$$y1 := [0.0 \quad 0.0 \quad 0.0 \quad 0.0 \quad 0.0 \quad 0.0 \quad 0.0 \quad 0.0 \quad 0.0 \quad 0.0 \quad 0.0 \quad 0.0 \quad 0.0 \quad 0.0$$

```
maple
> FFT(KK,x1,y1);
```

$$16$$

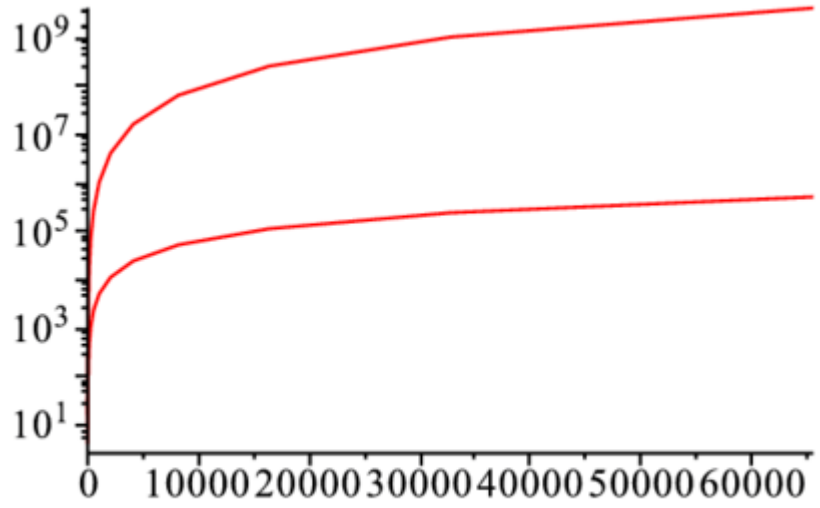
```
maple
> interface(displayprecision=2):
  print(x1);print(y1);
```

$$[0.0 \quad -2.0 \quad 0.0 \quad -2.0 \quad 0.0 \quad -2.0 \quad 0.0 \quad -2.0 \quad 0.0 \quad -2.0 \quad 0.0 \quad -2.0 \quad 0.0 \quad -2.$$

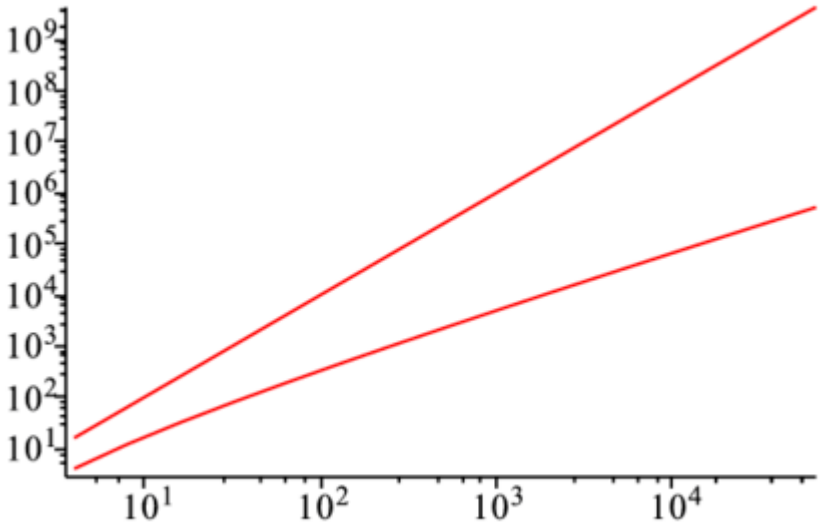
$$[0.0 \quad 10.05 \quad 0.0 \quad 2.99 \quad 0.0 \quad 1.34 \quad 0.0 \quad 0.40 \quad 0.0 \quad -0.40 \quad 0.0 \quad -1.34 \quad 0.0 \quad -2.9$$

```
maple
> F2:=unapply(sum(evalf((x1[i]+I*y1[i])*exp(I*2*Pi*(i-
1)/L*x)/N),i=1..N/2)+
         sum(evalf((x1[N-i+2]+I*y1[N-i+2])*exp(-I*2*Pi*(i-
1)/L*x)/N),i=2..N/2),x):
> plot({Re(F2(x)),F(x)},x=0..1);
```


```
maple
> l1:=logplot(dN2): l2:=logplot(dFft):
> display(l1,l2);
```



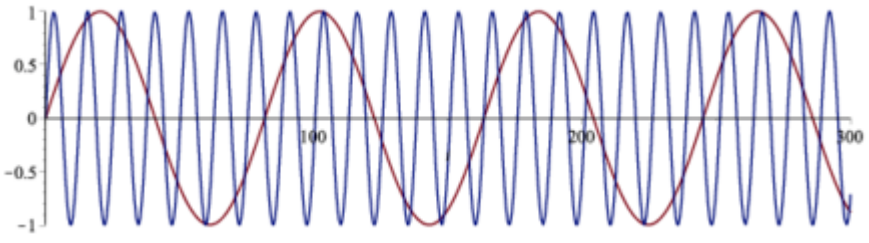

```
maple
> l1:=loglogplot(dN2): l2:=loglogplot(dFft):
> display(l1,l2);
```

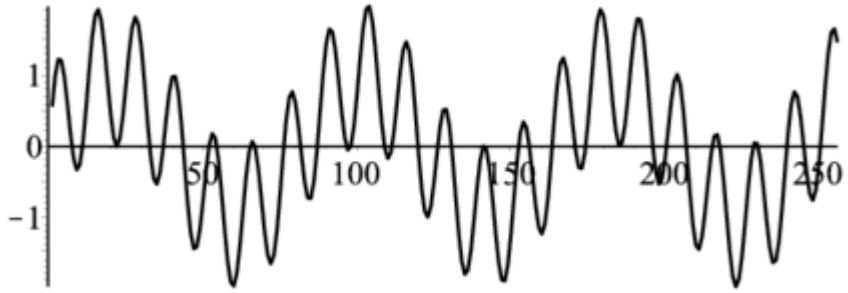


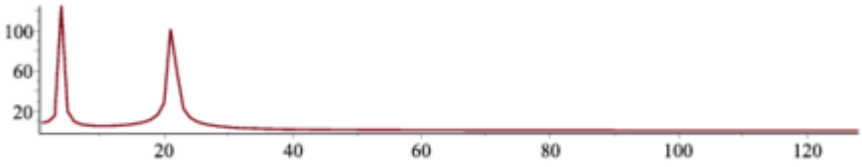
課題と解答例

合成波のFFT

下の例に従って，sin(i/13)とsin(i/2)の合成波を作成し，FFTをかけた後，周波数での強度を表示せよ．合成波(2*sin(i*2)+sin(i/2))との違いをのべよ．







In [ ]: