SASデータステップを使った高速フーリエ変換

杉本 忠則
大日本住友製薬 研究管理部

Fast Fourier transform using SAS data step

Tadanori Sugimoto
Research Administration, Dainippon Sumitomo Pharma Co., Ltd.

生態から経時的に取得されるデータの周波数解析

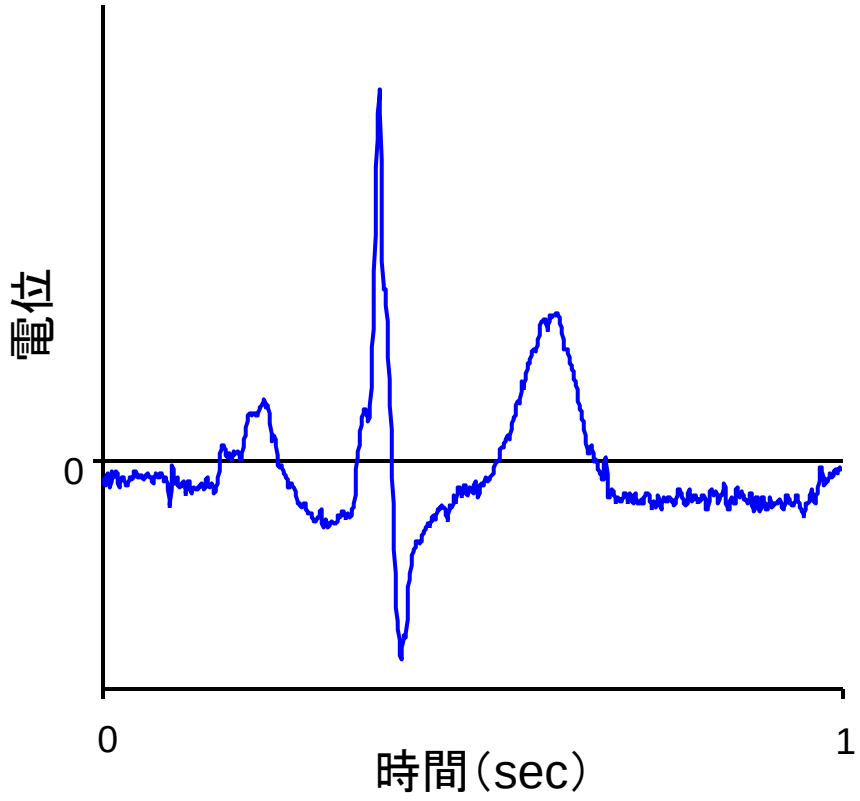
心電図データや血圧データ

脳波データ

筋電図データ

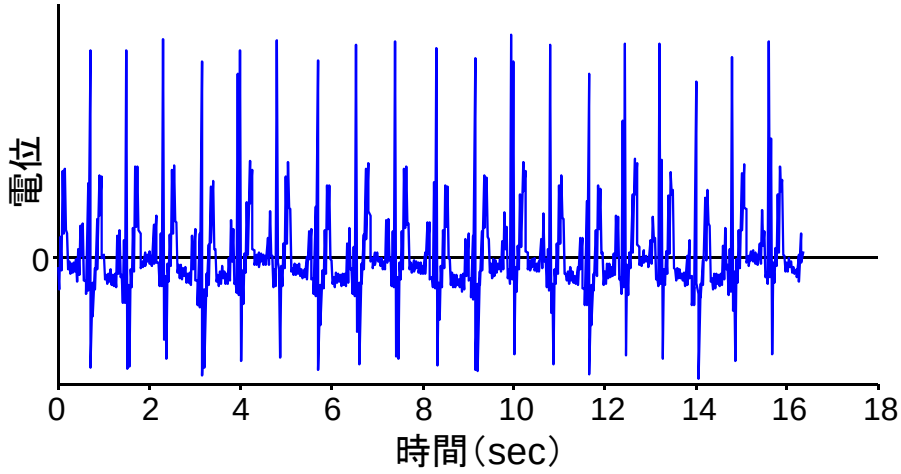
フーリエ変換による解析例（心電図の解析）

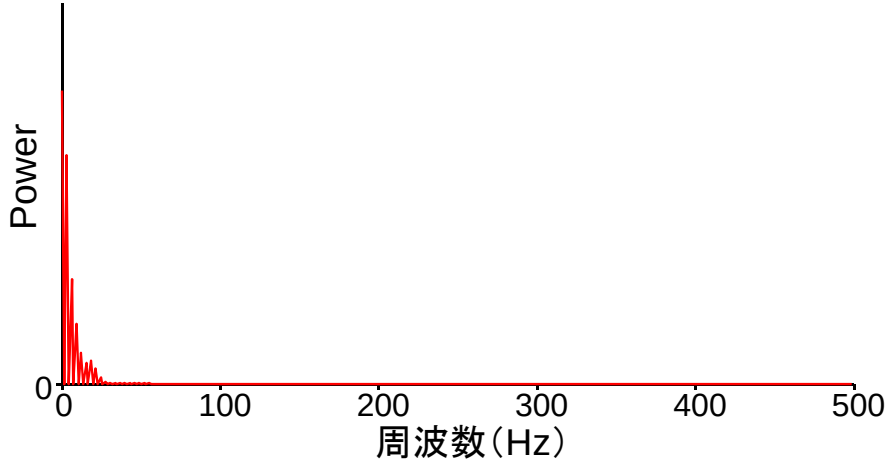
対象：健常者（人）
計測時間：16秒間（サンプリングタイム1msec）

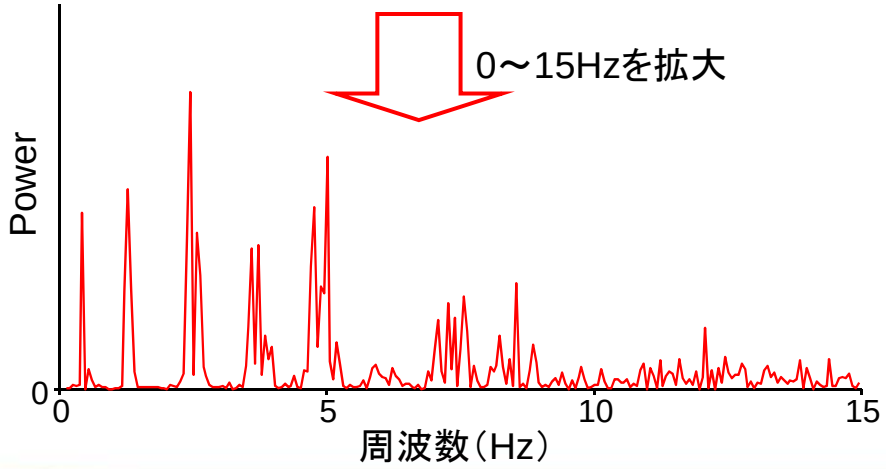


フーリエ変換による解析例（心電図の解析）

対象：健常者（人）
計測時間：16秒間（サンプリングタイム1msec）







2^3個のデータのフーリエ変換例

n	実数	虚数
0	2	0
1	4	-6
2	6	2
3	10	-4
4	8	2
5	2	0
6	6	2
7	8	6

フーリエ変換 →

← 逆フーリエ変換

n	実数	虚数
0	46	2
1	5.3	3.7
2	6	-14
3	5.3	-2
4	-2	10
5	-17.3	-7.7
6	-10	10
7	-17.3	-2

n	実数	虚数
0	46	2
1	-17.3	-2
2	-10	10
3	-17.3	-7.7
4	-2	10
5	5.3	-2
6	6	-14
7	5.3	3.7

注：定義により左右どちらかに変換される。

高速フーリエ変換（Fast Fourier Transform, FFT）アルゴリズムの流れ

本スライドでは2^3個のデータのFFTについて、次の4ステップに分けて説明します。

step1-1（データ変換過程1）
step1-2（データ変換過程2）
step1-3（データ変換過程3）
step2　（データ並替過程）

2^3個のデータのFFTアルゴリズムの流れ（step1-1）

変換前のデータ

n	実数	虚数
0	2	0
1	4	-6
2	6	2
3	10	-4
4	8	2
5	2	0
6	6	2
7	8	6

n=0,4組み合わせで以下の変換を行う。

n	実数		虚数	
	和	差(*a*)	和	差(*b*)
0	10	-6	2	-2

↓ 変換後の上位の実数へ　↓ 変換後の上位の虚数へ

n	$c=\cos\left(\frac{\pi n}{2^3}\times 2\right)$	$s=\sin\left(\frac{\pi n}{2^3}\times 2\right)$	$ac-bs$	$bc+as$
0	1	0	-6	-2

↓ 変換後の下位の実数へ　↓ 変換後の下位の虚数へ

同様に、*n*=1,5、*n*=2,6、*n*=3,7でも変換を行う。

変換後のデータ

n	実数	虚数
0	10	2
1	6	-6
2	12	4
3	18	2
4	-6	-2
5	5.7	-2.8
6	0	0
7	5.7	8.5

2^3個のデータのFFTアルゴリズムの流れ（step1-2）

変換前のデータ

n	実数	虚数
0	10	2
1	6	-6
2	12	4
3	18	2
4	-6	-2
5	5.7	-2.8
6	0	0
7	5.7	8.5

n=0,2組み合わせで以下の変換を行う。

n	実数		虚数	
	和	差(*a*)	和	差(*b*)
0	22	-2	6	-2

↓ 変換後の上位の実数へ　↓ 変換後の上位の虚数へ

n	$c=\cos\left(\frac{\pi n}{2^3}\times 4\right)$	$s=\sin\left(\frac{\pi n}{2^3}\times 4\right)$	$ac-bs$	$bc+as$
0	1	0	-2	-2

↓ 変換後の下位の実数へ　↓ 変換後の下位の虚数へ

同様に、*n*=1,3、*n*=4,6、*n*=5,7でも変換を行う。

変換後のデータ

n	実数	虚数
0	22	6
1	24	-4
2	-2	-2
3	8	-12
4	-6	-2
5	11.3	5.7
6	-6	-2
7	11.3	0

2^3個のデータのFFTアルゴリズムの流れ（step1-3）

変換前のデータ

n	実数	虚数
0	22	6
1	24	-4
2	-2	-2
3	8	-12
4	-6	-2
5	11.3	5.7
6	-6	-2
7	11.3	0

n=0,1組み合わせで以下の変換を行う。

n	実数		虚数	
	和	差(*a*)	和	差(*b*)
0	46	-2	2	10

↓ 変換後の上位の実数へ　↓ 変換後の上位の虚数へ

n	$c=\cos\left(\frac{\pi n}{2^3}\times 8\right)$	$s=\sin\left(\frac{\pi n}{2^3}\times 8\right)$	$ac-bs$	$bc+as$
0	1	0	-2	10

↓ 変換後の下位の実数へ　↓ 変換後の下位の虚数へ

同様に、*n*=2,3、*n*=4,5、*n*=6,7でも変換を行う。

変換後のデータ

n	実数	虚数
0	46	2
1	-2	10
2	6	-14
3	-10	10
4	5.3	3.7
5	-17.3	-7.7
6	5.3	-2
7	-17.3	-2

2^3個のデータのFFTアルゴリズムの流れ（step2）

step1終了時のデータ

n	実数	虚数	新*n*
0	46	2	0
1	-2	10	4
2	6	-14	2
3	-10	10	6
4	5.3	3.7	1
5	-17.3	-7.7	5
6	5.3	-2	3
7	-17.3	-2	7

新*n*の順番に並べ替える

なお、逆フーリエ変換では、下位実数・虚数を求める途中計算で下記式を用い、最終値を2^3で割る。

$$\text{下位実数} = ac + bs$$

$$\text{下位虚数} = bc - as$$

FFTの流れ（step1-1）

変換前のデータ

n	実数	虚数
0	2	0
1	4	-6
2	6	2
3	10	-4
4	8	2
5	2	0
6	6	2
7	8	6

n	上位		下位		実数		虚数		*c*	*s*	*a*c -*b*s	*b*c +*a*s
	実数	虚数	実数	虚数	和	差(*a*)	和	差(*b*)				
0	2	0	8	2	10	-6	2	-2	1	0	-6	-2
1	4	-6	2	0	6	2	-6	-6	0.7	0.7	5.7	-2.8
2	6	2	6	2	12	0	4	0	0	1	0	0
3	10	-4	8	6	18	2	2	-10	-0.7	0.7	5.7	8.5

変換後のデータ

n	実数	虚数
0	10	2
1	6	-6
2	12	4
3	18	2
4	-6	-2
5	5.7	-2.8
6	0	0
7	5.7	8.5

SASデータセットを利用すると
処理が楽にできるのではないだろうか！

SASを利用したFFT

SASデータセットの流れ1

```
data fft_data1;
    input n re im;
cards;
0   2   0
1   4  -6
2   6   2
3  10  -4
4   8   2
5   2   0
6   6   2
7   8   6
;
```

fft_data1

n	*re*	*im*
0	2	0
1	4	-6
2	6	2
3	10	-4
4	8	2
5	2	0
6	6	2
7	8	6

SASデータセットの流れ2

注意:
SASデータセットは2つに分割するので、上位、下位を意味するため変数に*u*、*l*を付けている。
変換前後を意味するため変数に1、2を付けている。

fft_data1

n	*re*	*im*	*F*
0	2	0	1
1	4	-6	1
2	6	2	1
3	10	-4	1
4	8	2	0
5	2	0	0
6	6	2	0
7	8	6	0

*F*の値で分割

n	*re*	*im*	*F*
0	2	0	1
1	4	-6	1
2	6	2	1
3	10	-4	1

n	*re*	*im*	*F*
4	8	2	0
5	2	0	0
6	6	2	0
7	8	6	0

SASデータセットのmerge

fft_data2

n_u	*re_u*1	*im_u*1	*F*	*n_l*	*re_l*1	*im_l*1
0	2	0	1	4	8	2
1	4	-6	1	5	2	0
2	6	2	1	6	6	2
3	10	-4	1	7	8	6

$$F = MOD\left(CEIL\left(\frac{n+1}{2^{3-1}}\right),2\right)$$

```
data fft_data1;
   set fft_data1;
   F= MOD(CEIL((n+1)/(2**(3-1))),2);

data fft_data2;
   merge fft_data1(where=(F=1) rename=(n=n_u re=re_u1 im=im_u1))
         fft_data1(where=(F=0) rename=(n=n_l  re=re_l1  im=im_l1));
```

SASデータセットの流れ3

$re\_u2 = re\_u1 + re\_l1$　　$im\_u2 = im\_u1 + im\_l1$

fft_data2

n_u	*re_u1*	*im_u1*	*F*	*n_l*	*re_l1*	*im_l1*	*re_u2*	*im_u2*	*re_l2*	*im_l2*
0	2	0	1	4	8	2	10	2	-6	-2
1	4	-6	1	5	2	0	6	-6	5.7	-2.8
2	6	2	1	6	6	2	12	4	0	0
3	10	-4	1	7	8	6	18	2	5.7	8.5

$$re\_l2 = (re\_u1 - re\_l1) \times \cos\left(\frac{\pi \times n\_u}{2^{3-1}}\right) - (im\_u1 - im\_l1) \times \sin\left(\frac{\pi \times n\_u}{2^{3-1}}\right)$$

$$im\_l2 = (im\_u1 - im\_l1) \times \cos\left(\frac{\pi \times n\_u}{2^{3-1}}\right) + (re\_u1 - re\_l1) \times \sin\left(\frac{\pi \times n\_u}{2^{3-1}}\right)$$

⇨

fft_data3

n	*re*	*im*
0	10	2
1	6	-6
2	12	4
3	18	2
4	-6	-2
5	5.7	-2.8
6	0	0
7	5.7	8.5

```
/*  データセットfft_data2において  */
   re_u2 =re_u1 +re_l1;
   im_u2=im_u1+im_l1;
   re_l2 =(re_u1 -re_l1 )*cos(3.14 * n_u / 2**(3-1))- (im_u1-im_l1)*sin(3.14 *n_u / 2**(3-1));
   im_l2=(im_u1-im_l1)*cos(3.14 * n_u / 2**(3-1))+(re_u1- re_l1 )*sin(3.14 *n_u / 2**(3-1));
```

```
data fft_data3;
   set fft_data2(keep=n_u re_u2 im_u2 rename=(n_u=n re_u2=re im_u2=im))
       fft_data2(keep=n_l  re_l2  im_l2  rename=(n_l=n  re_l2 =re im_l2 =im));
   by n;
```

注意：本スライド中では、円周率を3.14として表示しているが、実際のプログラムでは精度の高い値を使うことが望ましい。

2^3個のデータの
step1-1のプログラム

```
data fft_data1;
   set fft_data1;
   F= MOD(CEIL((n+1)/(2**(3-1))),2);
data fft_data2;
   merge fft_data1(where=(F=1) rename=(n=n_u re=re_u1 im=im_u1))
         fft_data1(where=(F=0) rename=(n=n_l  re=re_l1  im=im_l1));
   re_u2 =re_u1 +re_l1;
   im_u2=im_u1+im_l1;
   re_l2 =(re_u1 -re_l1 )*cos(3.14 * n_u / 2**(3-1)) -(im_u1-im_l1)*sin(3.14 * n_u / 2**(3-1));
   im_l2=(im_u1-im_l1)*cos(3.14 * n_u / 2**(3-1))+(re_u1 -re_l1 )*sin(3.14 * n_u / 2**(3-1));
data fft_data3;
   set fft_data2(keep=n_u re_u2 im_u2 rename=(n_u=n re_u2=re im_u2=im))
       fft_data2(keep=n_l   re_l2  im_l2 rename =(n_l =n re_l2 =re im_l2=im));
   by n;
```

2^3個のデータなので上記プログラムを3回繰り返すが、データ数が2^M個でも対応できるようにマクロ（下記）にする。

2^M個のデータの
step1-Xのプログラム
（マクロ）

```
%MACRO step1(DATASET,M,X);
   data &DATASET;
      set &DATASET;
      F= MOD(CEIL((n+1)/(2**(&M-&X))),2);
   data &DATASET;
      merge &DATASET(where=(F=1) rename=(n=n_u re=re_u1 im=im_u1))
            &DATASET(where=(F=0) rename=(n=n_l  re=re_l1  im=im_l1));
      re_u2 =re_u1 +re_l1;
      im_u2=im_u1+im_l1;
      re_l2 =(re_u1 -re_l1 )*cos(3.14*n_u/ 2**(&M -&X)) -(im_u1-im_l1)*sin(3.14*n_u/ 2**(&M -&X));
      im_l2=(im_u1-im_l1)*cos(3.14*n_u/ 2**(&M -&X))+(re_u1- re_l1 )*sin(3.14*n_u/ 2**(&M -&X));
   data &DATASET;
      set &DATASET(keep=n_u re_u2 im_u2 rename=(n_u=n re_u2=re im_u2=im))
          &DATASET(keep=n_l  re_l2  im_l2 rename =(n_l =n re_l2 =re im_l2=im));
      by n;
%MEND step1;
```

step1終了後のデータの並替（step2）

step1終了時のデータ

n	実数	虚数	新*n*
0	46	2	0
1	-2	10	4
2	6	-14	2
3	-10	10	6
4	5.3	3.7	1
5	-17.3	-7.7	5
6	5.3	-2	3
7	-17.3	-2	7

*n*から新*n*への変更は、
以下のように行える。

n	2進法表示		新*n*
0	000	000	0
1	001	100	4
2	010	010	2
3	011	110	6
4	100	001	1
5	101	101	5
6	110	011	3
7	111	111	7

```
%MACRO step2(DATASET,M);
   data &DATASET;
      set &DATASET;
      nn=0;
      do i=0 to &M-1;
         nn=nn+2**(&M-i-1)*MOD(INT(n/(2**i)),2);
      end;
      keep re im nn;
   proc sort data=&DATASET out=&DATASET; by nn;
%MEND step2;
```

注意：FFTの数値算出はSASデータステップのみで行っている。
ただし、最後の新*n*で並び替える操作（周波数順の並替）についてはproc SORTを利用している。

マクロstep1とマクロstep2を利用して、FFTを行うマクロ

```
%MACRO step1(DATASET,M,X);
   …
%MEND step1;

%MACRO step2(DATASET,M);
   …
%MEND step2;

%MACRO fft(DATASET,M);
   data &DATASET; set &DATASET(firstobs=1 obs=%eval(2**&M));
   %DO R=1 %TO %eval(&M);
      %step1(&DATASET,&M,&R);
   %END;
   %step2(&DATASET,&M);
%MEND fft;

%fft(fft_data1,3);
proc print data=fft_data1;
run;
```

上記マクロ（%MACRO fft）でデータセットDATASETの中の2^M個のデータがFFTされる。

SASによるFFT処理時間と他のプログラムとの比較

解析データ数	SAS	Excel（2003）		コンパイル済みプログラム	
		分析ツール	VBA	Visual Basic	Visual C++
2^{10}=1024	0.014sec（100回平均）	0.08sec（100回平均）	0.006sec（1000回平均）	0.00013sec（1000回平均）	0.00014sec（1000回平均）
2^{12}=4096	0.025sec（100回平均）	0.25sec（100回平均）	0.030sec（100回平均）	0.00045sec（1000回平均）	0.00042sec（1000回平均）
2^{15}=32768	0.087sec（100回平均）	不能	0.280sec（100回平均）	0.00402sec（1000回平均）	0.00401sec（1000回平均）
2^{20}=1048576	7.326sec（100回平均）	不能	12.070sec（100回平均）	0.44052sec（1000回平均）	0.44224sec（1000回平均）

SASによる解析条件
SAS:SAS BIServer
CPU:デュアルコアXeon 3.3GHz

他の解析条件
CPU:インテルCore2Quadプロセッサー 3.00GHz

Visual BasicやVisual C++でのFFTプログラム作成時の参考資料

奥村晴彦. C言語による最新アルゴリズム事典. 技術評論者（1991）
研究と教育と追憶と展望. http://tsuyu.cocolog-nifty.com/blog/2007/03/publi.html
高速フーリエ変換（アルゴリズム）. http://www.softist.com/programming/fft/fft.htm

まとめ

SASデータステップを用いて高速フーリエ変換（FFT）を行うプログラムを作成した。
ただし、変換後データの並替操作にはSASのproc SORTを使用した。

FFTを実行するソフトウェアとしては、Excelの分析ツールがある。また、FFT実行プログラムはExcel VBAや各種プログラム言語で作成することができるので、これらと性能を比較した。

その結果、処理時間は

コンパイル済みプログラム＜SAS ≃ Excel VBA＜Excel分析ツール

となった。
なお、Excel分析ツールでは2^{12}個のデータまで解析できるが、SASの場合はそれ以上のデータの解析が可能であった。

以上より、今回のプログラムは高速処理が求められる解析には不向きであるが、データ取得後の解析には十分利用できる。