腕の曲げ伸ばし運動を速く行う際の手首の動きの減少と上腕二頭筋、上腕三頭筋の筋活動量の増加

北田　和

2018 年 6 月 22 日

概要

ボールを投げたり、楽器を叩いたりする場合など腕を速く動かすことはよくある。だが、よりボールを速く投げたり、楽器を速く叩いたりしたいと考えた場合、どうすれば良いのだろうか。これらに共通することとして、腕を速く動かすことが挙げられる。そこで、腕の曲げ伸ばし運動に注目した。具体的には腕の曲げ伸ばし運動をする際に、何も指示されなかった場合となるべく速く動かすよう指示された場合の筋活動や関節の動きの変化を調べた。その結果、筋活動量の増加、腕を伸ばしきらない動きをすることで、腕の曲げ伸ばし運動を速くできることがわかった。

1　序論

ボールを投げたり、楽器を叩いたりする場合に腕を速く動かすことはよくある。そこで、「もっとボールを速く投げたい」、「もっと楽器を速く叩きたい」場合に重要となるのは、腕を速く曲げ伸ばしすることである。では、腕を速く曲げ伸ばしするにはどのようにすれば良いのだろうか。そこで、被験者に腕の曲げ伸ばし運動を何も指示をせずに行ってもらう場合 (slow-task) となるべく速く行うよう指示した場合 (fast-task) の 2 通りの計測実験を行うことにし、上腕二頭筋と上腕三頭筋の筋電と手首、肘、肩の座標を測定し、fast-task に対する筋活動量と関節の座標がどのように変化するかを調べた。

2　手法

被験者 (19 歳、男性、陸上部所属) に机に利き腕を置いて腕の曲げ伸ばし運動をするように指示した。図 1 にこの際のイメージを示す。測定計は、ワイヤレス筋電センサ 乾式・加速度 16G[LP-WS1222] を上腕二頭筋、上腕三頭筋の二カ所に、モーションキャプチャ用の反射マーカを図 2 のイメージのように手首、肘、肩に取り付けた。モーションキャプチャのカメラは被験者の頭上から撮影し、腕の曲げ伸ばし運動を二次元平面として捉えた。この平面の次元軸は図のとおりである。この際に撮影の関係上、被験者には膝立ちで机に向かってもらった。なお、筋電センサはサンプリング周波数 1000Hz で測定を行った。今回はモーションキャプチャによる関節の軌道に注目したため、付属している筋電センサは使用していない。

普通に腕の曲げ伸ばし運動を行う場合と速く曲げ伸ばし運動を行う場合を比較するため、被験者に対し何も指示を出さず腕の曲げ伸ばし運動を行った場合 (slow-task) となるべく速く腕の曲げ伸ばし運動を行うように指示した場合 (fast-task) を測定する二回の計測実験を行った。

実験一回の計測時間は 20 秒間に設定した。また、腕の曲げ伸ばしを開始して約 5 秒後から計測を開始することで、腕の曲げ伸ばし運動をしているデータのみを取得した。

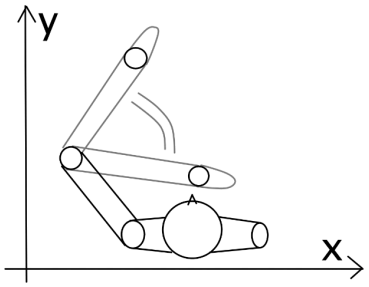

図 1　腕の曲げ伸ばし運動のイメージ

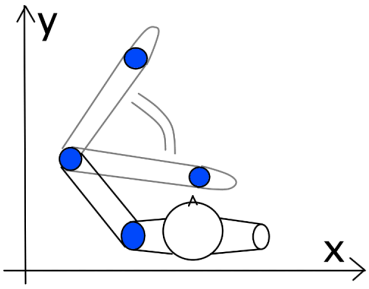

図 2　モーションキャプチャの反射マーカ貼り付け位置イメージ

2.1　筋電データ処理

筋活動の変化に注目するため、1∼4Hz の通過領域をもつバンドパスフィルタをかけることで、筋電計から取得したデータに混じる高周波のノイズを除去した。この際、全体の時間のずれを直すため filtfilt をかけた。次に、筋肉の活動度を時間変化として比較するため、(1) の式を用いて ΔT 間の筋活動量を算出した。以上の処理を行うプログラムを付録 A にまとめた。

$$a(t) = \frac{1}{\Delta T} \int_{t-\Delta T/2}^{t+\Delta T/2} |E(t)| dt \tag{1}$$

2.2　モーションキャプチャデータ処理

得られたデータには、ヘッダ部に計測点数、単位時間当たりのコマ数などの情報があり、テール部には最小値、最大値などの情報が含まれている。また、データ部にはシーン、時間、座標 X の各計測点データ、座標 Y の各計測点データが含まれている。しかし、この時間データは 10ms 刻みであるため、得られた座標データの時間よりも大まかである。そのため、ヘッダ部の単位時間当たりのコマ数から座標データに対応する細かな時間を求めて解析に使用した。以上の処理を行うプログラム付録 B にまとめた。

3　Results(結果)

図 3 に被験者の y 軸方向の手首の座標変化を計測開始時間から 1~4 秒の 3 秒間のデータを元に作成したグラフを掲示する (今後、説明がない場合は slow-task の結果を青、fast-task の結果を赤のグラフで示す。)。この図より、y 軸方向に slow-task が 3 回、fast-task が 7 回手首を前後させていることが示される。

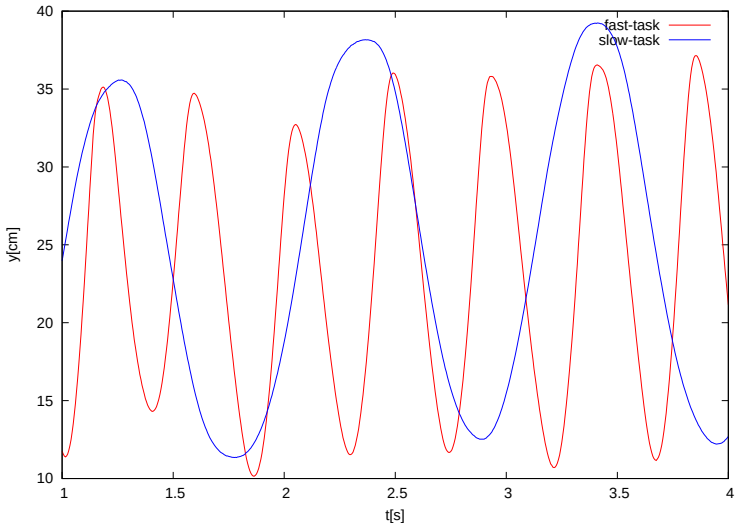

図 3　手首の時間における y 軸方向の動き

ここで、図 3 と同時間の上腕二頭筋、上腕三頭筋の EMG 活動量のデータをグラフにした図 4、5 に注目する。図 4 の fast-task では 1.6 秒に、slow-task では 2.3 秒に活動量のピークが存在する。一方、図 3 の fast-task では 1.6 秒に、slow-task では 2.4 秒に手首が y 軸の正の方向に出るタイミング存在する。これは、上腕二頭筋の活動量のピークと手首が y 軸の正の方向に出るタイミングがほぼ等しいことがこれ以外にも同様に存在するため、主に腕を伸ばすタイミングに上腕二頭筋を使用していることが示唆される。また、同様に、図 5 の fast-task では 1.8 秒に、slow-task では 2.7 秒に活動量のピークが存在する。一方、図 3 の fast-task では 1.8 秒に slow-task では 2.8 秒に手首が y 軸の負の方向に戻るタイミング存在する。このように、上腕三頭筋の活動量のピークと手首が y 軸の負の方向に戻るタイミングがほぼ等しいことがこれ以外にも同様に存在するため、主に腕を曲げるタイミングに上腕三頭筋を使用していることが示唆される。

次に、純粋な筋活動の特徴としては、上腕二頭筋の fast-task では 1.6 秒に、slow-task では 2.3 秒に筋活動量のピークが存在することは前述したが、その後 fast-task では 1.8 秒に、slow-task では 2.7 秒に筋活動量は下がっている。同様の傾向がその他にも存在する。一方、上腕三頭筋の fast-task では 1.8 秒に、slow-task では 2.7 秒に筋活動量のピークが存在し、その後 fast-task では 2.1 秒に、slow-task では 3.1 秒に小さなピークが存在し、同様の傾向がその他にも存在する。これは、腕の曲げ伸ばしに対して上腕二頭筋は腕を伸ばすタイミングで収縮させているが、上腕三頭筋は腕を曲げるタイミングと伸ばすタイミングどちらでも収縮させていることを現している。

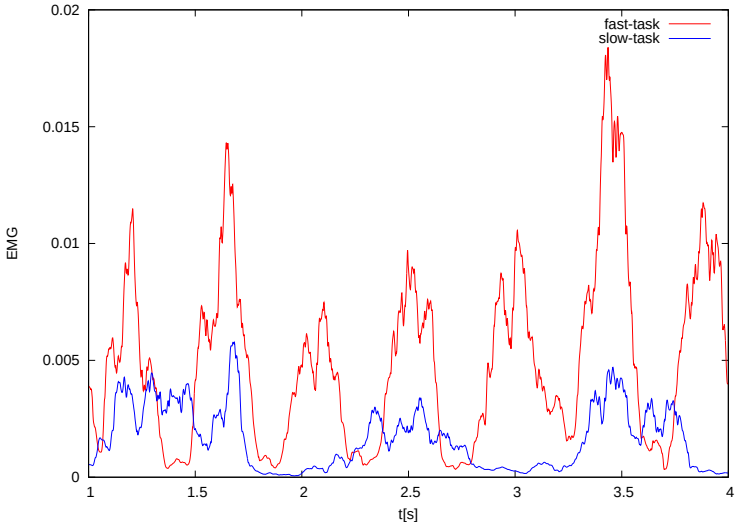

図 4　上腕二頭筋の EMG 活動量

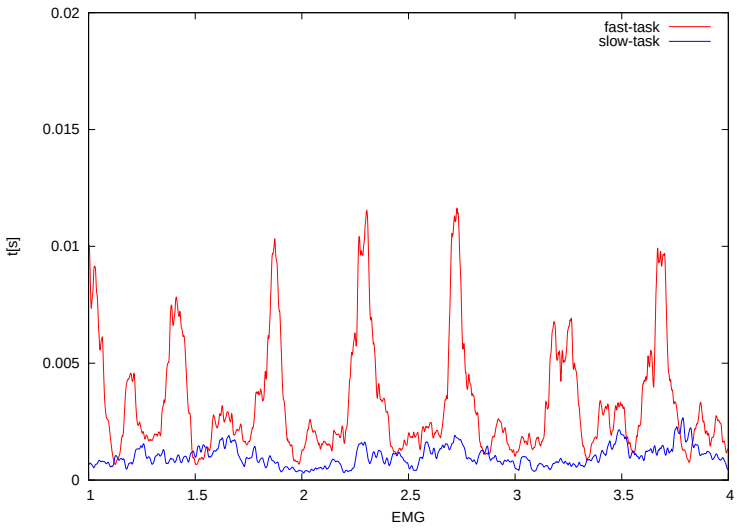

図 5　上腕三頭筋の EMG 活動量

そこで、図 6、図 7 に上腕二頭筋、上腕三頭筋のそれぞれの EMG 活動量のグラフと被験者の正面方向の手首の座標変化を重ねたグラフを示す。これより、上腕二頭筋では腕を伸ばしたタイミングより活動量増加のピークは若干遅く起こる傾向があった。また、上腕三頭筋では腕を曲げ終わるタイミングよりも前に活動量増加のピークが起こる傾向があった。

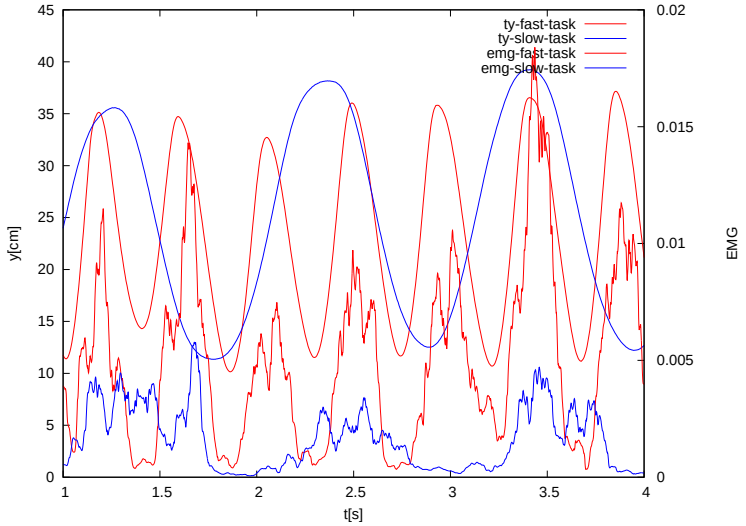

図 6　上腕二頭筋の EMG 活動量と手首の動きの比較

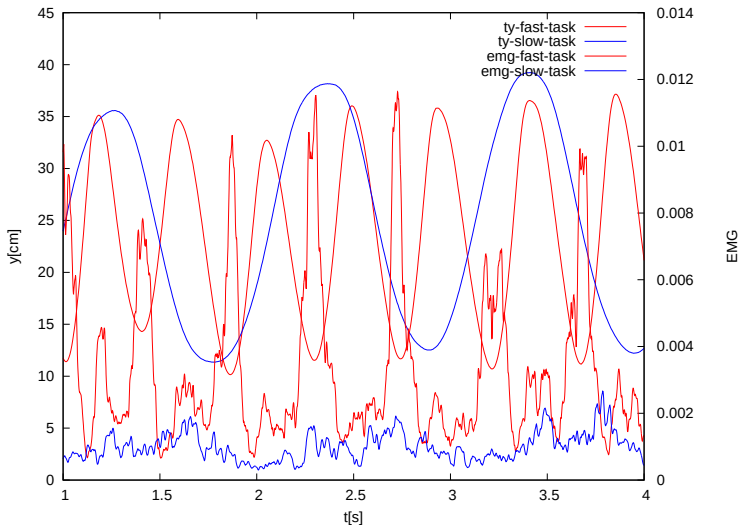

図 7　上腕三頭筋の EMG 活動量と手首の動きの比較

次に、slow-task と fast-task の比較を行う。筋活動に注目すると slow-task と fast-task では上腕二頭筋でも上腕三頭筋でも fast-task の筋活動量の方が多くなっている。これより、腕の曲げ伸ばし運動を速く行おうと意識した方が、筋活動量が多くなる傾向がある。では筋活動量の変動には大きな変化は見られるのだろうか、ここで、手首の y 軸方向の動きに対しての筋活動量の変化を見るために腕を曲げた状態から腕を 3 往復させた分のデータを取り出し比較する。図 8、図 9 に時間を slow-task が約 0.7∼4.0 秒、fast-task が約 1.4∼2.7 秒のデータを切り出し時間に対して正規化した結果と対応する上腕二頭筋、上腕三頭筋の筋活動量をそれぞれグラフ化したものを示す。

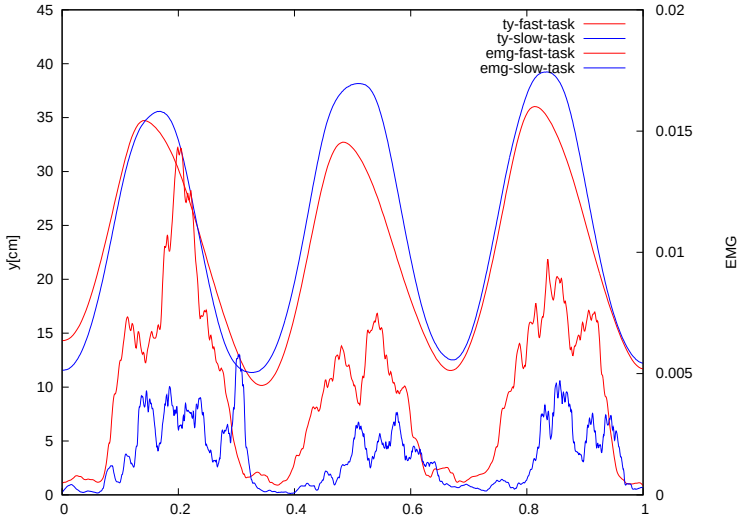

図 8　正規化後の上腕二頭筋の EMG 活動量と手首の動きの比較

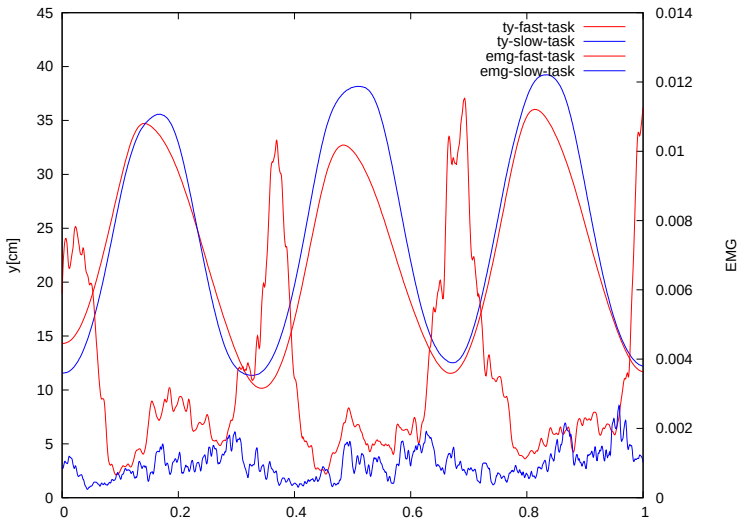

図 9　正規化後の上腕三頭筋の EMG 活動量と手首の動きの比較

まず、手首の y 軸方向の前後の軌道について注目すると slow-task に対して、fast-task の方が 3∼4cm ほど前後が小さくなっていることが示される。また、slow-task は腕の曲げる速度も伸ばす速度も大体等しいのに対して、fast-task は腕を伸ばすときの速度の方が曲げる速度よりも速くなっていることが示される。次に、筋活動に注目する。上腕二頭筋では大きな差は見られないのに対し、上腕三頭筋では slow-task は腕が曲がりきる前に筋活動量のピークが来ているのに対して、fast-task は腕が曲がりきった後に筋活動量のピークが来ていることが示される。

4 Discussion(考察)

fast-task では筋活動量が増加したことより、筋肉量をあげれば腕を速く曲げ伸ばしできることが考えられる。また、fast-task では y 軸方向の手首の動きが減少したことより、小さく曲げ伸ばし運動をすれば腕を速く曲げ伸ばしできることが考えられる。以上のことにより、楽器を叩く場合などのような伸ばしきる必要がない場合には腕をなるべく小さく曲げ伸ばしできるように練習することで、伸ばしきる必要がある場合にも共通しては、筋肉を鍛えることで速く動かすことができると考えられる。

5 Conclusions(結論)

腕の曲げ伸ばし運動を行う際には大まかに伸ばす時に上腕二頭筋を曲げる時に上腕三頭筋を使用している。しかし、それぞれの活動量のピークは上腕二頭筋は腕を伸ばし終わった後に、上腕三頭筋は slow-task の際は腕を曲げ終わる直前に、fast-task の際は腕を曲げ終わった後にある。

付録 A　筋電データの処理のプログラム

A.1　バンドパスフィルタのプログラム

```
import csv
import numpy as np
from scipy import signal
import matplotlib.pyplot as plt
%matplotlib inline
import pandas as pd
from pandas import Series, DataFrame

f = open('PID255_MID11_FileID002_000000_0010.csv',
'r',encoding="shift-jis")
s = open('data_12_2.txt','w',encoding="shift-jis")
i=0
datam=''
dataReader = csv.reader(f)

t, emg, x, y, z= [], [], [], [], []

for row in dataReader:
    if i<10:
        datam=datam+str(row)
    else:
        s.writelines('%s\n' %("   ".join(row)))
        t += [float(row[0])]
        emg += [float(row[1])]
        x += [float(row[2])]
        y += [float(row[3])]
        z += [float(row[4])]

    i=i+1

print(datam)
f.close()
s.close()

n = i-10
dt =0.001
f = 1000
fn = 1/(2*dt)
td = np.linspace(1, n, n)*dt-dt

fp = 1
fs = 40
kagen = 0.01
jougen = 1.0

Wp = fp/fn
Ws = fs/fn

N, Wn = signal.buttord(Wp, Ws, kagen, jougen)
b1, a1 = signal.butter(N,Wn, "low")
y1 = signal.filtfilt(b1, a1, emg)

plt.figure()
plt.plot(td, emg, "b")
plt.plot(td, y1, "r", linewidth=2 , label="butter")
plt.xlim(0,1)
plt.xlabel("Time [s]")
plt.ylabel("Amplitube");

x=0
ef = open('bandpass11_1.txt','w',encoding="shift-jis")
while x < n:
    ef.write('%f,' %td[x])
    ef.write('%f\n'  %y1[x])
    x+=1
```

A.2　整流化のプログラム

```
#include<stdio.h>
#include<math.h>
int main(void){
  FILE *rf, *wf;
  double data[2];

  rf = fopen("bandpass12_2.txt","r");
  wf = fopen("abs12_2.txt","w");

  while((fscanf(rf,"%lf,%lf",&data[0],&data[1]))!=EOF){
    fprintf(wf,"%f,%f\n",,data[0],fabs(data[1]));
  }

  return 0;
}
```

A.3　活動量の変化を求めるプログラム

```
#include<stdio.h>
#include<stdlib.h>
#include<math.h>
#define BUF_SIZE 256
#define DT 50
#define SAMPLE 1000
void integration(double e[],double a[], int n){
  int i;
  for(i=0; i<DT; i++){
    a[0]+=(e[i]/SAMPLE);
  }
  for(i=1; i<(n-DT); i++){
    a[i]=a[i-1]+(e[i+DT]-e[i-1])/SAMPLE;
  }
}

/*-----------------------------------------------*/
int main(void){
  FILE *rf, *wf;
  double data, *e, *a;
  int n=0, i;
  char buf[BUF_SIZE];

  if((rf = fopen("abs12_2.txt","r")) ==NULL)
```

```
25     return -1;
26   wf = fopen("katudou12_2.txt","w");
27
28   while((fgets(buf,BUF_SIZE,rf))!=NULL){
29     n++;
30   }
31
32
33   fclose(rf);
34   if((rf = fopen("abs12_2.txt","r")) ==NULL)
35     return -1;
36
37   e =(double*)malloc(sizeof(double)*n);
38   a =(double*)malloc(sizeof(double)*(n-DT));
39
40   if(e == NULL || a == NULL){
41     printf("メモリが確保できませんでした。\n");
42     exit(EXIT_FAILURE);
43   }
44
45   for(i=0; i<(n-DT); i++)
46     a[i]=0;
47
48   i=0;
49   while((fscanf(rf,"%lf ,",&data))!=EOF){
50     fscanf(rf,"%lf",&e[i]);
51     i++;
52   }
53
54   integration(e,a,n);
55
56
57   for(i=0; i<(n-DT); i++)
58     fprintf(wf,"%lf,%lf\n",((double)(i+(DT/2))/1000.0),a[i]);
59
60
61   fclose(rf);
62   free(a);
63   free(e);
64   fclose(wf);
65
66   return 0;
67 }
```

付録 B　モーションキャプチャデータの処理のプログラム

B.1　データをデータ部とそれ以外に分けるシェルスクリプト

```
 1 #!/bin/bash
 2 if test ! -e $1.csv && test ! -e $1; then
 3     echo "ファイルを指定してください"
 4 elif test -f $1.csv; then
 5 #    FNAME="$1"
 6     nkf -w $1.csv |sed -E "s_^,//g"| sed -E "s/^[\t ]+//g" | egrep "^[0-9]"  > pos-$1.dat
 7     nkf -w $1.csv | sed "s/,/ /g" | sed -E "s/^[\t ]+//g" | egrep -v "^[0-9]"  > info-$1.dat
 8 else
 9     nkf -w $1 |sed -E "s_^,//g"| sed -E "s/^[\t ]+//g" | egrep "^[0-9]"  > pos-${1/.csv/}.dat
10     nkf -w $1 | sed "s/,/ /g" | sed -E "s/^[\t ]+//g" | egrep -v "^[0-9]"  > info-${1/.csv/}.dat
11 fi
```

B.2　データ部をマーカごとにファイル分けするプログラム

```
 1 #include<stdio.h>
 2 #include<stdlib.h>
 3 #include<string.h>
 4 #include<math.h>
 5 #define N 20
 6 #define M 256
 7 int main(int argc,char *argv[]){
 8   FILE *rf,*wf[N];
 9   double sample, *data;
10   int makar, i=0,n=1, dmen;
11   char fn[N][M]={'\0'};
12
13
14   if( (sample = atof(argv[1]))==0 ||
(makar = atoi(argv[2])) ==0 || (dmen = atoi(argv[3]))==0 ){
15     printf("サンプリング周波数、マーカ数を数値で入力して
ください。\n\a");
16     i=-1;
17   }
18   else if((rf = fopen(argv[4],"r"))==NULL){
19     printf("ファイルが存在しません。\n\a");
20     i=-1;
21   }
22   if(i==-1){
23     printf("extract <サンプリング周波数> <マーカ数> <次
元> <ファイル名>\n で指定してください。\n");
24     return 0;
25   }
26
27
28
29   for(i=0; i<makar; i++){
30     sprintf(fn[i],"%d-%s",(i+1),argv[4]);
31     wf[i]=fopen(fn[i],"w");
32   }
33
34   data = malloc(sizeof(double)*(dmen*makar+2));
35
36   while(fscanf(rf,"%lf ,%lf ",&data[0],&data[1])!=EOF){
37
38     for(i=0; i<(makar*dmen); i++){
39       if( ( fscanf(rf,",%lf ",&data[i+2]))==EOF ){
40 printf("元ファイルに欠損があります。
出来たファイルは使えません。\n");
41 return 0;
42       }
43     }
44
45     for(i=0; i<makar; i++){
46       fprintf(wf[i],"%lf",data[0]/sample);
47     }
48
49     for(i=0; i<(dmen*makar); i++)
50       fprintf(wf[i%makar],",%lf",data[i+2]);
51     for(i=0; i<makar; i++)
52       fprintf(wf[i],"\n");
```

```
53   }
54
55
56   fclose(rf);
57   for(i=0; i<makar; i++)
58     fclose(wf[i]);
59
60   free(data);
61
62   return 0;
63 }
```

B.3　ヘッダ部から情報を取り出し B.2 に数値を与えるシェルスクリプト

```
1 #!/bin/bash
2 if test ! -e $1.csv && test ! -e $1; then
3     echo "ファイルを指定してください"
4 elif test -f $1.csv; then
5     ./extract $(grep "コマ数" info-$1.dat  |cut -f 1 -d "/"|sed "s/[^0-9|.]//g")
$(grep "計測点数" info-$1.dat  |cut -f 2 -d "," | grep -o [0-9]) 2 pos-$1.dat
6 else
7     ./extract $(grep "コマ数" info-${1/.csv/}.dat  |cut -f 1 -d "/"|sed "s/[^0-9|.]//g")
$(grep "計測点数" info-${1/.csv/}.dat  |cut -f 2 -d "," | grep -o [0-9]) 2 pos-${1/.csv/}.dat
8 fi
```

B.4　入力ファイルの座標データのみをファイル出力するシェルスクリプト

```
1 #!/bin/bash
2 if test ! -e $1.dat && test ! -e $1; then
3     echo "ファイルを指定してください"
4 elif test -f $1; then
5     cut -f 2- -d "," $1 |  sed -e 's/^[ ]*//g' > ${1/pos/xy}
6 else
7     cut -f 2- -d "," $1.dat | sed -e 's/^[ ]*//g' > ${1/pos/xy}.dat
8 fi
```