物理学情報処理演習

12. 数値計算

データ処理

最小二乗法

移動平均

Unix shell と パイプ処理・script 言語

参考文献

Numerical Recipes in C
W. H. Press他著　技術評論社

Advanced Engineering Mathematics
Erwin Keryszig　John Wiley & Sons, Inc.

実習UNIXシェル
エリー・クイグリー著　プレンティスホール出版

大久保晋

E-mail: buturi-johoshori@tiger.kobe-u.ac.jp

http://extreme.phys.sci.kobe-u.ac.jp/staffs/okubo/lectures/Programming/index.html

最小二乗法：一般解

より一般的な理論式にデータをフィットさせることを考えよう。

一般的な理論式として，n個の関数の組み

$$\phi_1(x),\phi_2(x),\cdots,\phi_n(x)$$

の線形結合

$$q_n\left(x;c_1,c_2,\cdots,c_n\right)=\sum_{i=1}^{n}c_i\phi_i(x)$$

とするとm個のデータ点(x_i, f_i)を最小二乗法でフィットさせるには関数

$$Q_n\left(c_1,c_2,\cdots,c_n\right)=\sum_{k=1}^{m}\left|q_n\left(x_k;c_1,c_2,\cdots,c_n\right)-f_k\right|^2$$

を最小 にする係数$c_1,c_2,\ldots,c_n$の組みを探せば良い。つまり全ての(x_k,f_k)で

$$\frac{\partial Q_n\left(c_1,c_2,\cdots,c_n\right)}{\partial c_1}=0$$

となればよい。

最小二乗法：一般解

$$Q_n(c_1,c_2,\cdots,c_n)=\sum_{k=1}^{m} {q_n}^2(x_k;c_1,c_2,\cdots,c_n)-2f_k q_n(x_k;c_1,c_2,\cdots,c_n)+f_k^2$$

であるので

$$\frac{\partial Q_n(c_1,c_2,\cdots,c_n)}{\partial c_1}=\sum_{k=1}^{m}\left(\frac{\partial {q_n}^2(x_k;c_1,c_2,\cdots,c_n)}{\partial c_i}-2f_k\frac{\partial q_n(x_k;c_1,c_2,\cdots,c_n)}{\partial c_i}\right)$$

$$=\sum_{k=1}^{m}\left(2q_n(x_k;c_1,c_2,\cdots,c_n)\frac{\partial q_n(x_k;c_1,c_2,\cdots,c_n)}{\partial c_i}-2f_k\frac{\partial q_n(x_k;c_1,c_2,\cdots,c_n)}{\partial c_i}\right)$$

となり，これがゼロであるから

$$\sum_{k=1}^{m}\left(q_n(x_k;c_1,c_2,\cdots,c_n)\frac{\partial q_n(x_k;c_1,c_2,\cdots,c_n)}{\partial c_i}\right)=\sum_{k=1}^{m}\left(f_k\frac{\partial q_n(x_k;c_1,c_2,\cdots,c_n)}{\partial c_i}\right)$$

$$\frac{\partial q_n(x_k;c_1,c_2,\cdots,c_n)}{\partial c_i}=\phi_i$$

となればよい。従って，以下の行列を解けばよい。

$$\begin{bmatrix}\sum_{k=1}^{m}\phi_1^2(x_k) & \sum_{k=1}^{m}\phi_1(x_k)\phi_2(x_k) & \cdots & \sum_{k=1}^{m}\phi_1(x_k)\phi_n(x_k)\\ \sum_{k=1}^{m}\phi_2(x_k)\phi_1(x_k) & \sum_{k=1}^{m}\phi_2^2(x_k) & \cdots & \sum_{k=1}^{m}\phi_2(x_k)\phi_n(x_k)\\ \vdots & \vdots & & \vdots\\ \sum_{k=1}^{m}\phi_n(x_k)\phi_1(x_k) & \sum_{k=1}^{m}\phi_n(x_k)\phi_2(x_k) & \cdots & \sum_{k=1}^{m}\phi_n^2(x_k)\end{bmatrix}\begin{bmatrix}c_1\\c_2\\\vdots\\c_n\end{bmatrix}=\begin{bmatrix}\sum_{k=1}^{m}f_k\phi_1(x_k)\\\sum_{k=1}^{m}f_k\phi_2(x_k)\\\vdots\\\sum_{k=1}^{m}f_k\phi_n(x_k)\end{bmatrix}$$

最小二乗法：一般解

特にxのn次式の場合

$$q_n\left(x;c_1,c_2,\cdots,c_n\right)=\sum_{i=1}^{n}c_i\phi_i(x)=x^{i-1}(i=1,2,\cdots,m)$$

では,

$$\begin{bmatrix} m & \sum_{k=1}^{m}x_k & \cdots & \sum_{k=1}^{m}x_k^{n-1} \\ \sum_{k=1}^{m}x_k & \sum_{k=1}^{m}x_k^{2} & \cdots & \sum_{k=1}^{m}x_k^{n} \\ \vdots & \vdots & & \vdots \\ \sum_{k=1}^{m}x_k^{n-1} & \sum_{k=1}^{m}x_k^{n} & \cdots & \sum_{k=1}^{m}x_k^{2(n-1)} \end{bmatrix}\begin{bmatrix} c_1 \\ c_2 \\ \vdots \\ c_n \end{bmatrix}=\begin{bmatrix} \sum_{k=1}^{m}f_k \\ \sum_{k=1}^{m}f_kx_k \\ \vdots \\ \sum_{k=1}^{m}f_kx_k^{n-1} \end{bmatrix}$$

となる。

これは，演習１１でやった連立一次方程式である。従って，Gauss-Jordan法で解けば理論式の係数$c_1,c_2,\ldots,c_n$が求まる。

演習１２－１：最小二乗法　2次式

$$\begin{bmatrix} m & \sum_{k=1}^{m} x_k & \cdots & \sum_{k=1}^{m} x_k^{n-1} \\ \sum_{k=1}^{m} x_k & \sum_{k=1}^{m} x_k^2 & \cdots & \sum_{k=1}^{m} x_k^n \\ \vdots & \vdots & & \vdots \\ \sum_{k=1}^{m} x_k^{n-1} & \sum_{k=1}^{m} x_k^n & \cdots & \sum_{k=1}^{m} x_k^{2(n-1)} \end{bmatrix} \begin{bmatrix} c_1 \\ c_2 \\ \vdots \\ c_n \end{bmatrix} = \begin{bmatrix} \sum_{k=1}^{m} f_k \\ \sum_{k=1}^{m} f_k x_k \\ \vdots \\ \sum_{k=1}^{m} f_k x_k^{n-1} \end{bmatrix}$$

３点(-2, -3), (-1, 2), (0, 1)を通る２次式の最小二乗の多項式（上の行列）は次の通り

$$\begin{bmatrix} 3 & \sum_{k=1}^{m} x_k & \sum_{k=1}^{m} x_k^{n-1} \\ \vdots & \cdots & \vdots \\ \sum_{k=1}^{m} x_k^{n-1} & \cdots & \sum_{k=1}^{m} x_k^{2(n-1)} \end{bmatrix} \begin{bmatrix} c_1 \\ c_2 \\ c_3 \end{bmatrix} = \begin{bmatrix} -3+2+1 \\ -3\cdot-2+2\cdot-1+1\cdot 0 \\ -3\cdot(-2)^2+2\cdot(-1)^2+1\cdot(0)^2 \end{bmatrix} = \begin{bmatrix} 0 \\ 4 \\ -10 \end{bmatrix}$$

となる。

演習１１－１で作ったGauss-Jordan法のプログラムを使ってこれを解け。
解は,

c_1=1, c_2=-4, c_3=-3

演習１２－２：最小二乗法　４次式

プログラム12-2.c の見本プログラム

このプログラムは、標準入力からスペース区切りのfloating型のXYデータを受け取り、4次式でfittingを行い、その結果の係数と誤差平均／誤差平均二乗を返す。

```
/* average error , standard deviation */
/* */
#include <stdio.h>
#include <math.h>

#define N 5   /* 5x5 matrix*/
#define MAX   10000   /* data max size */


int main(void){
  double a[N][N+1];
  double d[2][MAX];
  double p, dd, f=0.0, err=0.0, ave_err, std_err1, std_err2;
  int i=0, j, k, m;

  /* 入力エンドまで読み込みをする。キーボード入力時は ctrl + d でEOF */
  while ( (scanf("%lf %lf", &d[0][i], &d[1][i])) != EOF ){i++;}
```

標準入出力を利用するため stdio.h
Sqrtを使うため math.h
をincludeする。

Fittingの次数Nをここで定義。
5はxの4次式。10とすれば9次式まで計算する

MAXはデータの読み込み上限。MAXでデータを読み込む配列サイズを決めている

a[][]はN次式を解くための行列、d[2][]はXYデータを読み込む配列

i, j は行列を解くために, kはデータを掃引するため, mは読み込んだデータサイズ

関数 scanf の返り値は読み込んだ数、読み込み修了時にEOF (End Of File) が返り値となる
２つの浮動小数点をd[0][i], d[1][i]に読み込むが、EOFのときにはwhile loopから抜ける。

演習１２－２：最小二乗法　４次式

プログラム12-2.c の見本プログラム

```
/* make materix */
  m = i ;  /* number of data set, nubmering from 1 */
  /* sigma x */
  for (i=0; i<N; i++){
    for (j=0; j<N; j++){
      for (k=0; k<m; k++){
        /* calculate matrix element */
        a[i][j] += pow(d[0][k], ((i+j)*1.0));
      }
    }
  }
  a[0][0] = m*1.0; /* cast to double */
  for (i=0; i<N; i++){
    for (k=0; k<m; k++){
      /* calculate RHS element */
      a[i][N] += d[1][k] * pow(d[0][k], i*1.0);
    }
  }
```

データサイズmは、読み込み時にloopさせたiで分かる。Loopは0からスタートするのでm=iでよい。

関数pow(a, b)はa^bを浮動小数点で与える関数。(i+j)*1.0は、変数の型を整数からdoubleに変換するため

a[i][j]は Σx_k^{i+j}なので、kで足しあわせている。

m*1.0はint型の変数mをdouble型の変数a[][]に合わせるための変換

Castを使って、(double)m としてもよい

連立方程式AX=Bの行列Aの右列にBを加えてGauss-Jordan法を使える行列にする。

演習１２－２：最小二乗法　４次式

プログラム12-2.c の見本プログラム

連立方程式の解を求めるために、Gauss-Jordan法を使う。
プログラムは11-1と同じ。

```
/* ---- Gauss-Jordan method ---- */
  for (k=0; k<N; k++){
    p = a[k][k];
    for (j=k; j<N+1; j++){
      a[k][j]=a[k][j]/p;      /* substract pivot gyou by p */
    }
    for (i=0; i<N; i++){    /* pivot sweep out */
      if (i!=k){
        dd = a[i][k];
        for (j=k; j<N+1; j++){
          a[i][j]=a[i][j]-dd*a[k][j];
        }
      }
    }
  }
```

演習１２－２：最小二乗法　４次式

プログラム12-2.c の見本プログラム

誤差平均の計算、二乗誤差の計算

Fitting式を計算するのにデータのx_kを元に係数a[][]を使い計算する

Fitting式との差のsum、差の二乗のsum

誤差平均、誤差の二乗の平均

係数の出力、誤差平均、二乗誤差の出力

```
/* calculate average error and std error */
  for (k=0; k<m; k++){
    f = 0.0;
    /* evaluate fitting function */
    for (i=0; i<N; i++){
      f += a[i][N]* pow(d[0][k], i*1.0);
    }
    err += d[1][k] - f; /* sum error */
    std_err2 += (f - d[1][k])*(f - d[1][k]);  /* sum error square */
  }
  ave_err = err/m;    /* evaluate average error */
  std_err1 = sqrt( std_err2/m );    /* evaluate std error */

  /* ---- out put solution ---- */
  for (k=0; k<N; k++){
    printf("c%d= %lf¥n", k, a[k][N]);
  }
  printf("ave_error= %g¥t ¥nstd_error= %g¥n", ave_err, std_err1);

  return (0);
}
```

演習１２－２：最小二乗法　４次式

xy01.txtを読み込んでfittingをしてみよ。

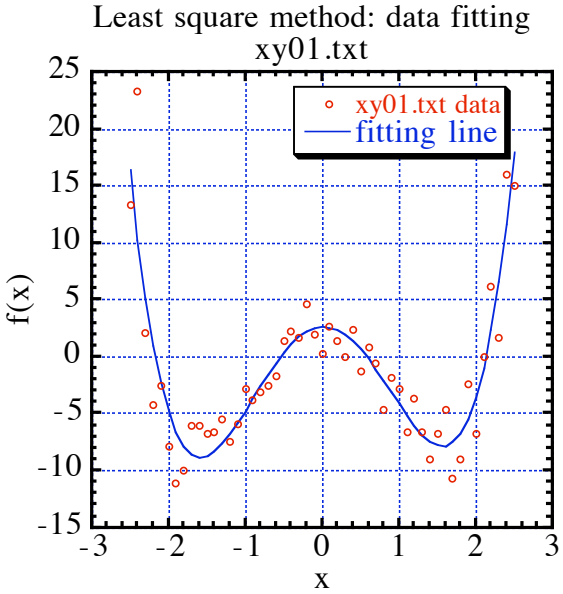
手順

1. Excelで xy01.txtを読み込んでプロットする
2. 次のコマンドを実行して、fittingしてその係数を記録する

   ./12-2 < xy01.txt
3. 先程のプロットにfittingしたプロットを合わせてプロットする

   Excel上で y = c0 + c1*x +c2*x*x + c3*x*x*x + c4*x*x*x*x としてプロットせよ。

右図のようになっているだろうか？



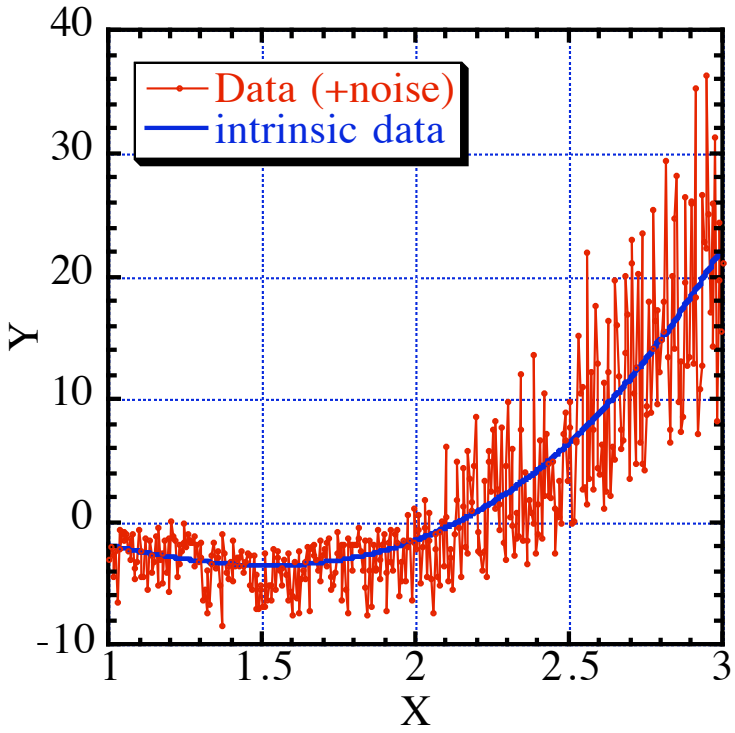
xy01.txtは
http://extreme.phys.sci.kobe-u.ac.jp/staffs/okubo/lectures/Programming/joho12/xy01.txt

演習１２－３：移動平均

実験データには、本質的な信号の他に電気回路や系全体から生じるノイズが含まれている。

通常、ノイズは高周波数成分が多く、データから見ると、本質的な信号を中心に左右に振れていると考えてよい。（右図参照）

本質的な信号がゆっくりと変化しているとすれば、データ点前後数点を平均化すれば高周波数のノイズは減るであろう。この考えをもとにノイズを減らしデータを滑らかにすることを移動平均（smoothing）と呼ぶ。



演習１２－３：移動平均

以下12-3.cは標準入力より入力されたデータ点を前後Nの平均として出力するプログラムである。

```
/* Idou Heikin */
#include <stdio.h>
#include <math.h>

#define N 5 /* heikin ryou */
#define MAX 10000 /* data max size
*/

int main(void){
  double d[2][MAX];
  double ave;
  int i, j, m;

  /* 入力エンドまで読み込みをする。キーボード入
力時は ctrl + d でEOF */
while ( (scanf("%lf %lf", &d[0][i],
&d[1][i])) != EOF ){i++;}

m=i;  /* number of data set,
numbering from 1 */
  for (i=(N-1); i<m; i++){
    ave = 0.0;
    for (j=0; j<N; j++){
      ave += d[1][j+i-N+1];
    }
    ave = ave/N;
    printf("%lf %lf¥n", d[0][i],
ave);
  }
  return (0);
}
```

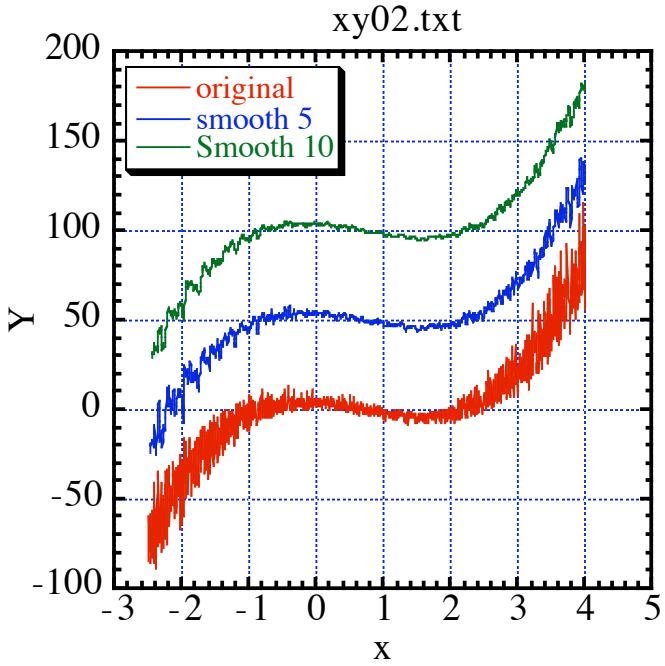
演習１２－３：移動平均

xy02.txtをsmoothing してプロットしてみる。

./12-3 < xy02.txt > xy02s.txt

としてみて、xy02s.txtと比較してみよう。

右図は、移動平均をとるときの移動平均量が５と１０を比べたものである。移動平均の量が大きければ大きい程滑らかになっていることが分かる



xy02.txtは
http://extreme.phys.sci.kobe-u.ac.jp/staffs/okubo/lectures/Programming/joho12/xy02.txt

Unix shell と パイプ処理・script 言語

データ処理をおこなう際に入力データ形式をC言語で作成したプログラムに整合するように書き換えたいことが多々ある。また、処理したデータを別のプログラムで処理をすることも多々ある。

これらを解決する方法はいくつもあるが、Unix上で処理をしているのであれば、極力標準入出力を使い、パイプ処理、scriptで自動処理させるのが最も単純で柔軟に処理できる。

演習１２－４：Unix shell と パイプ処理

次に12-2 のプログラムを使い最小二乗fittingを行うが、事前に12-3のプログラムでsmoothing を行いfitting を行うことを考える。
２つの方法でこれを実現しよう。

A.　12-3で処理し、ファイル(tmp.dat)を作成する。そのファイルを12-2で処理する。

```
./12-3 < xy02.txt > tmp.dat
./12-2 < tmp.dat
```

B.　中間ファイル tmp.dat を作らず、パイプ処理で一度に処理する。

```
./12-3 < xy02.txt | ./12-2 -
```

|　（縦棒）はパイプと呼ばれる。パイプの前の処理

```
./12-3 < xy02.txt
```

で標準出力に出力される結果を - に入力する仕事をする。
つまり、 `./12-3`で計算された結果を `./12-2`　に渡している。

UNIXでは、標準入出力を使ったプログラムならばパイプ処理が使えて便利である。このような単一処理のミニプログラムをつなげて高度な機能を持たせることができるので、新規処理にも柔軟に対応できる。

Unix shell：bash

UNIXのコマンドラインで、ユーザーインターフェースを担っているのはshellと呼ばれるプログラムである。MS-DOSなどではDOS-shellのみであるのに対して、UNIXには様々なshellがあるが、大別して以下の３つの代表的なshellがある。

Bourne shell, C shell, Korn shell

ほぼ同じ処理がどのshellでも可能であるが、文法と効率において違いがある。

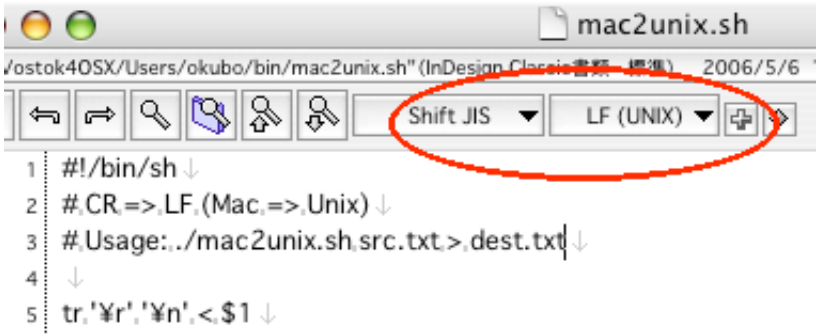
OSX tigerの標準は Bourne shell の一つである bash が使われている。この演習でもbashを使うことを想定して説明する。

shellでは、コマンドライン上で対話的に使用するのと、shell scriptと呼ばれる命令手順を記述したプログラムから利用する方法がある。

Unix shell：事前準備

演習の最初に述べたように、Macの標準改行コードはCR（改行）である。

一方UNIXの改行はLFである。UNIXで実行させるscriptは改行コードをLFで合わせる必要がある。（その方が無難である）



mac2unix.sh という名前で以下を入力する。
保存は LF(UNIX)で（右図参照）

```
#!/bin/sh
# CR => LF (Mac => Unix)
# Usage: ./mac2unix.sh src.txt > dest.txt

tr '¥r' '¥n' < $1
```

コマンドラインで

chmod u+x mac2unix.sh

として実行可能ファイルに変える。

unix2mac.sh という名前のファイルを作り、上と同様にLFで保存し、実行可能ファイルに変えておく

```
#!/bin/sh
# LF => CR (Unix => Mac)
# Usage: ./unix2mac src.txt > dest.txt

tr '¥n' '¥r' < $1
```

¥はバックスラッシュ \

Unix shell：事前準備

mac2unix.sh と unix2mac.sh の使い方

mac2unix.sh [入力ファイル] > [出力ファイル]

いう形で実行すれば、改行コードを Mac (CR) -> UNIX (LF) に変える。

unix2mac.sh も同様。

注意として、実行ファイルの在処（path）が通っていないので、実行するには、絶対パス指定するなどする必要がある。

例）$HOME/joho-shori/mac2unix.sh　など

演習１２−５：Unix shell（bash）

複数のファイルの拡張子の名前付けを一度に変更することをしてみよう。
ここにある.zipファイルをダブルクリックして展開するとnumberというフォルダーに　0.jpeg 〜 9.jpeg の１０個のファイルが用意される。

http://extreme.phys.sci.kobe-u.ac.jp/staffs/okubo/lectures/Programming/number.zip

A.　コマンドラインから対話式

このディレクトリnumberに移動してコマンドラインから以下のコマンドを入れてみよう。

```
ls
0.jpeg 3.jpeg 6.jpeg 9.jpeg
1.jpeg 4.jpeg 7.jpeg
2.jpeg 5.jpeg 8.jpeg                    ←拡張子はすべて.jpeg

for fname in *.jpeg
> do  ←拡張子がjpegのファイルが存在する限り、以下の命令を繰り返す
> mv $fname ${fname%.jpeg}.jpg
> done

ls
0.jpg 1.jpg 2.jpg 3.jpg 4.jpg 5.jpg 6.jpg 7.jpg 8.jpg 9.jpg
```

演習１２−５：Unix shell（bash）

先ほどの拡張子jpgをjpegに戻す。

B. shell scriptで解決する

miエディターでren.shという名前のファイルを同じディレクトリに作る。以下を入力する。改行コードをUNIXの改行コード LFで保存すること

```
#!/bin/bash
for fname in `ls *.jpeg`
do
mv $fname ${fname%.jpeg}.jpg
done
```

これを実行可能に変える。コマンドラインで以下を入力する

chmod u+x ren.sh

ls -l ren.sh

```
-rwxr--r--  1 okubo  okubo  69 19  6 12:26 ren.sh
```

これで実行可能になった。

このディレクトリnumberに移動してコマンドラインから実行してみよう。

```
./ren.sh
ls
0.jpg 1.jpg 2.jpg 3.jpg 4.jpg 5.jpg 6.jpg 7.jpg 8.jpg 9.jpg
```