空気力学課題レポート

03-170313
飯山敬大
2017/8/3

[課題 1]

例として、下図 1 のような、流れの中に置かれた鈍頭物体(円柱)後方の流れを考える。この時流れは物体後方で剥離し、剥離した流れからは剪断層(速度の異なる 2 つの流体が接する時にできる層)が形成される(剥離剪断層)。

剥離剪断層の内側には、渦により流体が循環するのみで、下流へと流れない死水域と呼ばれる領域が形成される。ここでは流れの逆流や y 方向変化も生じている。

また 2 つの剥離点を出発した 2 つの剥離剪断層(渦糸の列)は互いに干渉し、下図 2 のような孤立的な渦列が形成される。これをカルマンの渦列と呼ぶ。カルマンの渦列において、渦列が安定であれば、図 2 において

$$\begin{cases} \frac{h}{a} = 0.281 \\ x_0 = \frac{a}{2} \end{cases} \qquad \cdots\cdots\cdots(1)$$

が成り立つ。円柱後方に周期的に放出される渦の周波数は、(1)の安定条件によって決まると考えられる。

[課題 2]

1)

2 つの翼形状について検討した。

1 つは

$$a=0.6,\ V_\infty=1,\ \Gamma=4\pi a\sin(\alpha+\beta),\ \alpha=5[deg],\ \beta=20[deg], c=0.5$$

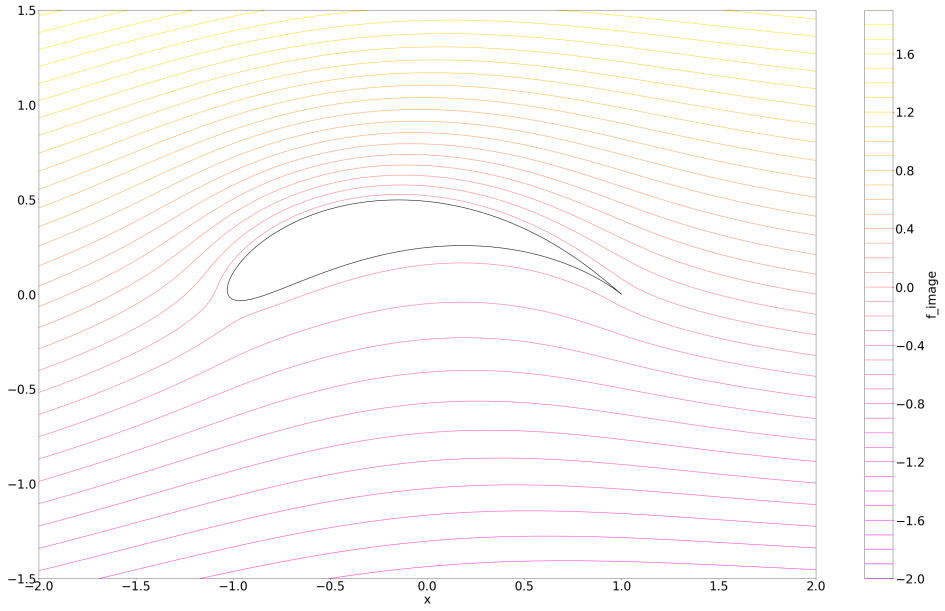
とパラメータを設定した翼（A とする）

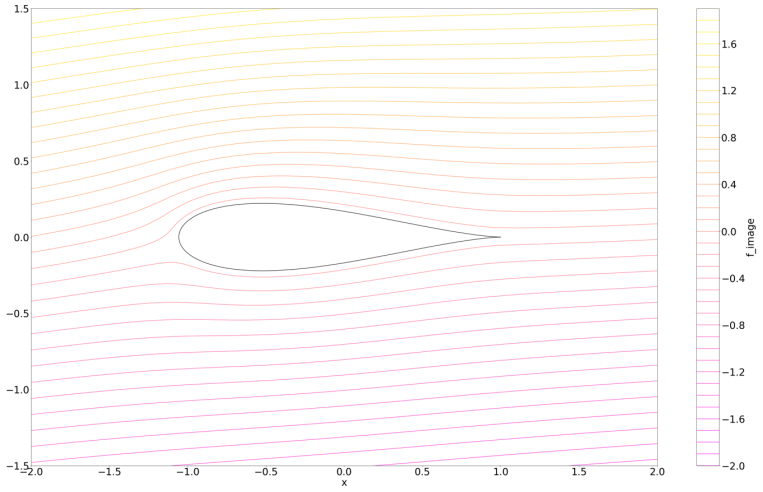
もう一つは他のパラメータは共通で、$\beta=0[deg]$

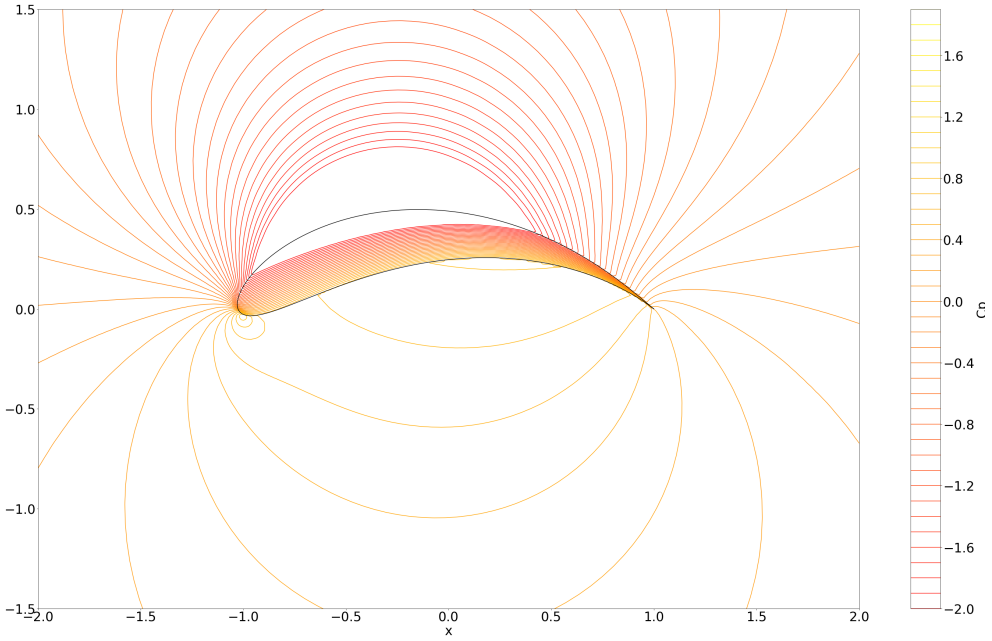
とした対称翼(B とする)である。翼形状は 2 以降の課題で記されている。

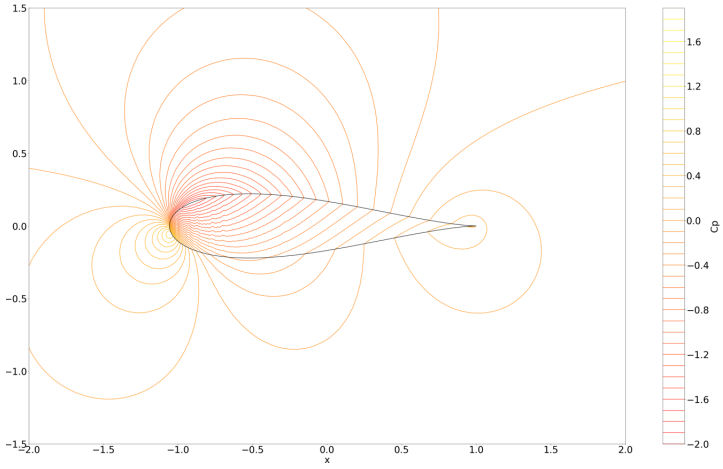
2)

流線と等圧線図は以下のようになった。(描画ソフトの等高線作成がうまく使えず等圧線図において翼内で線が繋がってしまっているが、本来はない線である。

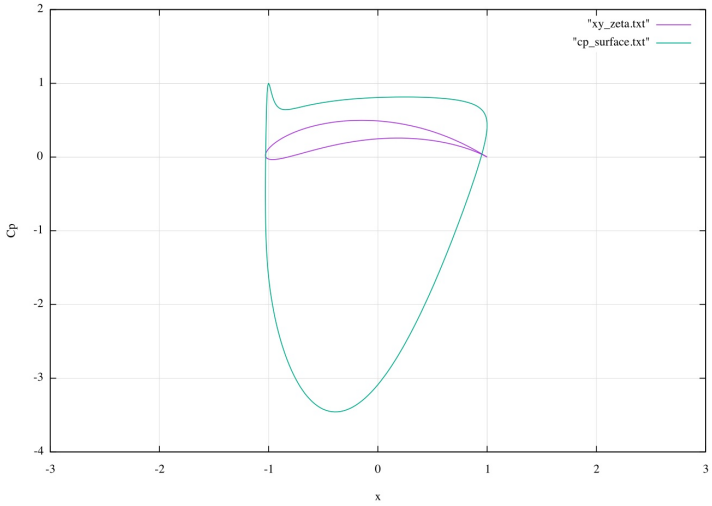


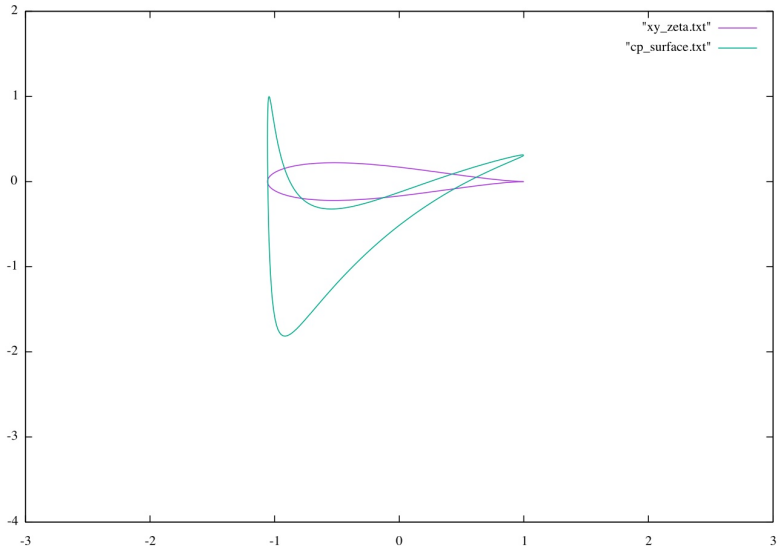






3)翼表面の圧力分布は下図のようになった。





4)計算の結果、$\beta = 20[deg]$の時、

$$\begin{cases} C_L = 3.186 \\ C_D = 0.00003 \\ x_{cp} = 0.4368 \end{cases}$$

となった。ただしx_{cp}は風圧中心で、単位は(前縁からの距離)／（コード長）である。
また、$\beta = 0[deg]$の対称翼の時は、

$$\begin{cases} C_L = 0.657 \\ C_D = -0.00002 \\ x_{cp} = 0.2492 \end{cases}$$

となった。この時風圧中心は 25%コード長にほぼ等しくなる。

5)次ページに記す(ある点周りの Cm を$\alpha = 0$~$10[deg]$で変化させながらプロットした)表から、迎角に寄らず Cm が一定である x の位置を読み取ると、$\beta = 20[deg]$の時、x_{ac}=-0.21[x 座標/コード長] となり、これが空力中心位置である。$\beta = 0[deg]$の時、x_{ac}=-0.24[x 座標/コード長]となる。それぞれ前縁からの長さに直せば、

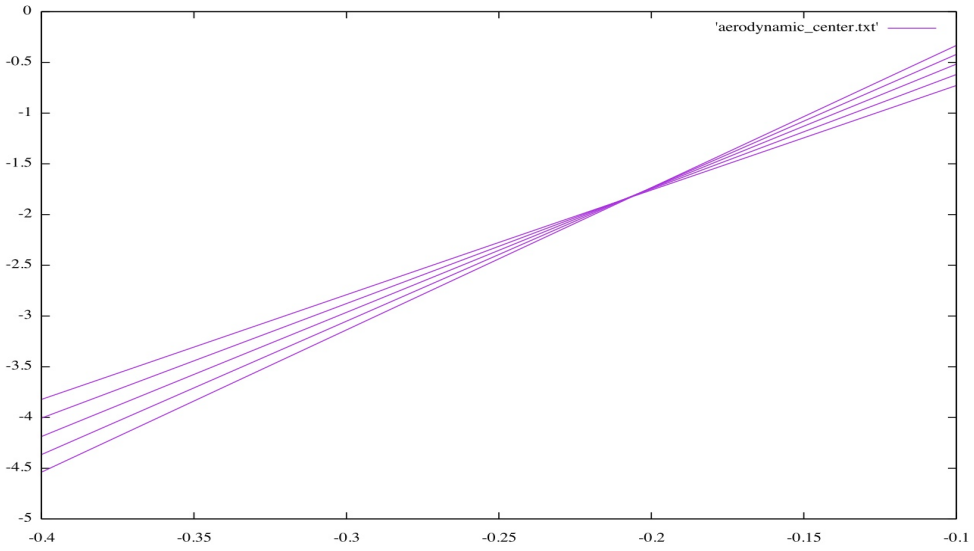
$$x_{ac} = \begin{cases} 0.29\ (\beta = 20[deg]) \\ 0.26\ (\beta = 0.0[deg]) \end{cases}$$

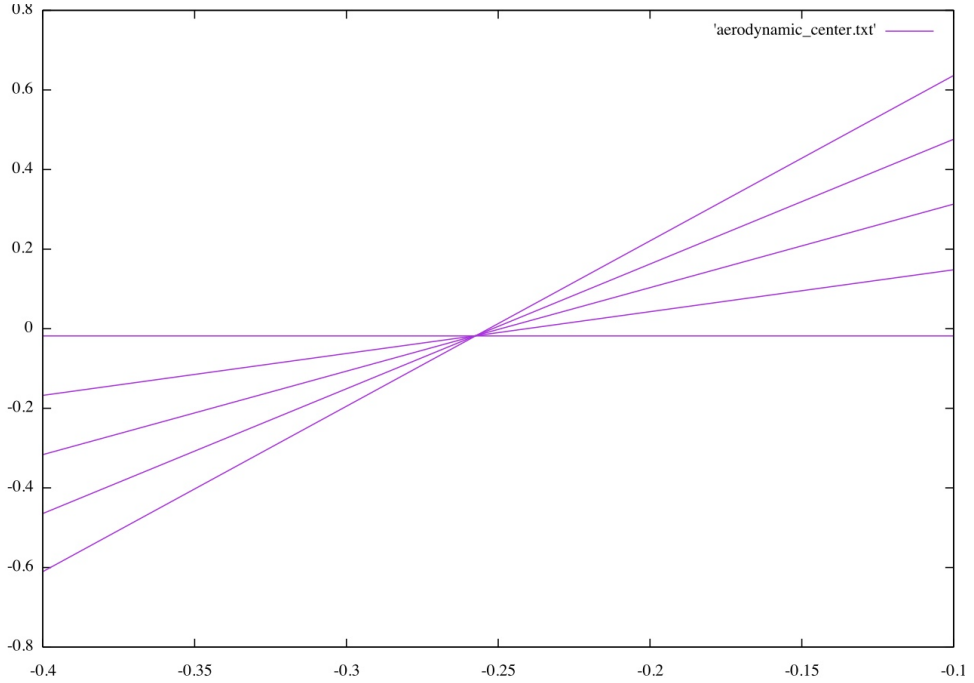
となる。

6)完全流体を仮定したので、非回転の流れが時間が経過しても維持されるが、実際には最初非回転の流れであっても、境界から必ず回転成分(渦)が誘起される。また、粘性や圧縮性の影響も考慮されていない。

巡行状態の流れ場であれば、流れが翼に沿っているため翼の表面に薄い境界層が形成されるのみであり、完全流体と仮定した流れでよく近似することができる(境界層の外側の流れ

では粘性が無視できるため)と考えられるが、迎角が大きくなり境界層が厚くなってきたり、さらに迎角が大きくなり流れが剥離して翼に揚力が生じなくなるような状態(stall)などの状態の流れ場は、完全流体では解析することができない。





[課題 3]

翼端とその先では揚力がないため、循環はゼロである。しかし、そこで渦糸が切れてしまうと考えると、循環の保存を示したヘルムホルツの渦定理に反してしまう。そこで、翼面状にスパン方向に循環の変化が生じた時に、そこから下流方向に随伴渦が延びていくという過程を置くと、それに伴い吹き下ろしが発生する。クッタ・ジュコフスキーの定理より、循環は渦糸に当たる風に直角方向に揚力を誘起するので、束縛渦(翼面上においた渦)に吹き下ろしが当たると、翼周りの循環は後方に抵抗を生成する。これが誘導抵抗である。
誘導抵抗をなくすためには、随伴渦が発生しないようにする工夫が必要である。そのためにはスパン方向の循環の変化をなくせば良い。

しかしながら、もし誘導抵抗が 0 にできるのであれば、翼端の循環が 0 になり、さらにスパン方向の循環の変化が 0 になるので、翼全体で循環が 0 になり、揚力が発生しないことになってしまう。よって揚力が 0 でない翼において誘導抵抗を完全に 0 にするのは不可能だと考えられる。

なるべく誘導抵抗を抑える方法を考える。証明は省略するが、ねじれなし、スパン方向に同一翼型の 3 次元翼であれば、循環はスパン方向に楕円分布しているので、楕円翼を設計すれば誘導抵抗を最小にできる。またアスペクト比を大きくすれば発生する揚力 (スパン長に比例) に対して翼端での循環の変化(断面形状に比例するため一定)の影響を少なくできるので、アスペクト比を大きく取れば(誘導抵抗)/(揚力)の割合は小さくできる。

誘導抵抗の利用方法としては、戦闘機などにおける空力ブレーキとして利用できる。

[参考文献]

1. 深野徹. (1994). 「わかりたい人の流体工学(II)」 裳華房

[ソースコード]

以下に計算に用いたソースコード(C++)を添付する。長くなるので、紙面の都合上データの描画に用いた Python プログラムは省略する。

```
//翼周りの流れのシミュレーション

#include<iostream>
#include<cstdio>
#include<cmath>
#include<complex>
#include<vector>
using namespace std;

double PI = 3.141592;

FILE *gid;
complex<double> cv,ca,cgama,ccent,small_z,large_z,cf_potential,cf_pressure;
double a = 0.6;
double v0 = 1.0;
double c025 = 0.5;
double gama = 5.0;

double x_z[180][100]; //Z 平面における x
double y_z[180][100]; //Z 平面における y
double x_zeta[180][100]; //zeta 平面における x
double y_zeta[180][100]; //zeta 平面における y
double f_stream[180][100]; //流線関数
double f_cp[180][100]; //圧力 Cp の分布

//計算用関数(空力中心以外)
void calculation(double alfa,double beta);
   void cal_z(int i,int j,double alfa,double beta);
   void grid_on_zetaplane(int i,int j);
   void cal_stream(int i,int j);
   void cal_pressure(int i,int j,double alfa);
void cal_force(double alfa,double beta);
//ファイルデータ書き込み関数
void write_data();
   void write_shape();
   void write_stream();
   void write_pressure();
   void write_surfacepressure();
//空力中心計算
void write_ac(double beta);
   double cal_cm(double x_sample);

//-----------------------------------------------main
int main(){
   int wing;
   double rad = (PI/180.0);
   double alfa,beta;
   printf("choose Jekofusuki(0) or symmetric(1):¥n");
```

```
    scanf("%d",&wing);
    if(wing == 0){
    //ジェコフスキー翼
        alfa = 5.0 * rad;
        beta = 20.0 * rad;
        calculation(alfa,beta);
        write_data();
        cal_force(alfa,beta); //Cl,CD,風圧中心を計算
        write_ac(beta); //空力中心を計算
    }
    else{
    //対称翼
        alfa = 5.0 * rad;
        beta = 0.0 * rad;
        calculation(alfa,beta);
        write_data();
        cal_force(alfa,beta); //Cl,CD,風圧中心を計算
        write_ac(beta); //空力中心を計算
    }
}

//-------------------------------------------------計算用関数(空力中心以外)
/*
考える領域をζではなくZで定義する
Z平面上で同心円上に計算点を配置。j=1がZ面、ζ面共に物体表面に対応している。
複素ポテンシャル(cf)とともに位置ζも求めている
*/
void calculation(double alfa,double beta){
    //Kutta condition
    gama = 4.0 * PI * v0 * a * sin(alfa+beta);

    for(int i=1;i<180;i++){
        for(int j=0;j<100;j++){
            //small_z,large_z
            cal_z(i,j,alfa,beta);
            //zeta-plain
            grid_on_zetaplane(i,j);
            //stream_function
            cal_stream(i,j);
            //pressure
            cal_pressure(i,j,alfa);
      }
    }
}

//large_z,small_zの計算
void cal_z(int i,int j,double alfa,double beta){
    cv = complex<double>(v0,0.0);
    ca = complex<double>(a,0.0);
    cgama = complex<double>(0.0,gama/(2.0*PI));
    //grid on Z-plane
    double dr = 0.05;
    double dtheta = 2.0*(PI/180.0); //rad
    double rrr = a + dr * ((double)j); //大きさ
```

```
    double theta0 = -beta;
    double theta = theta0 + dtheta * ((double)i); //角度
    //プリントの zexp(iα)に対応(Z 平面上の同心円)
    x_z[i][j] = rrr * cos(theta);
    y_z[i][j] = rrr * sin(theta);
    //プリントの Zc
    ccent = complex<double>(c025,0.0) + ca * exp(complex<double>(0.0,PI - beta));
    //プリントの Z に対応
    large_z = complex<double>(x_z[i][j],y_z[i][j])+ccent;
    //Step2 の z
    small_z = (large_z-ccent) * exp(complex<double>(0.0,-alfa));
}

//ξ 平面の x-y 関係の計算 j=0 が翼表面
void grid_on_zetaplane(int i,int j){
    x_zeta[i][j] = real(large_z + complex<double>(c025*c025,0.0)/large_z);
    y_zeta[i][j] = imag(large_z + complex<double>(c025*c025,0.0)/large_z);
}

//流線関数の計算
void cal_stream(int i,int j){
    //プリント step2 の f
    cf_potential = cv * (small_z + ca*ca/small_z) + cgama * log(small_z);
    f_stream[i][j] = imag(cf_potential);
}

//圧力の計算
void cal_pressure(int i,int j,double alfa){
    cf_pressure = exp(complex<double>(0.0,-alfa))
          *(cv * (complex<double>(1.0,0.0)-ca*ca/(small_z*small_z)) + cgama/small_z)
          /(complex<double>(1.0,0.0)
          - complex<double>(c025,0.0) * complex<double>(c025,0.0) / (large_z * large_z));
    double cp = 1.0
          - (real(cf_pressure) * real(cf_pressure) + imag(cf_pressure)*imag(cf_pressure))/(v0 * v0);
    f_cp[i][j] = cp;
    if(i==0 && j==0){
    printf("real_cp:%f,img_cp:%f¥n",real(cf_pressure),real(cf_pressure));
    printf("f_cp[0][0]:%f¥n",cp);
    }
}

//Cl,Cd,風圧中心の計算
void cal_force(double alfa,double beta){
    calculation(alfa,beta);
    double cxp = 0.0;
    double cyp = 0.0;
    double sum = 0.0;
    for(int i=1;i<179;i++){
        double dxw = x_zeta[i+1][0] - x_zeta[i][0];
        double dyw = y_zeta[i+1][0] - y_zeta[i][0];
        double dnx = dyw;
        double dny = -dxw;
        double cpm = (f_cp[i+1][0]+f_cp[i][0])/2.0;
        cxp = cxp - cpm * dnx;
```

```
        cyp = cyp - cpm * dny;
        double fx = -cpm * dnx;
        double fy = -cpm * dny;
        sum = sum + (fy * x_zeta[i][0] - fx * y_zeta[i][0]);
    }
    double center_of_pressure = sum/cyp; //x 座標
    cxp = cxp / (4.0 * c025);
    cyp = cyp / (4.0 * c025);
    double cdp = cxp*cos(alfa) + cyp*sin(alfa);
    double clp = cyp*cos(alfa) - cxp*sin(alfa);

    printf("CL = %f¥n",clp);
    printf("CD = %f¥n",cdp);
    printf("center_of_pressure=%f(前縁からの距離)¥n",center_of_pressure/(4.0 * c025) + 0.5);
}

//---------------------------------------------データのファイル書き込み

void write_data(){
        write_shape();
        write_stream();
        write_pressure();
        write_surfacepressure();
}

//翼型の表示
void write_shape(){
    FILE *fid;
    const char *data={"xy_zeta.txt"};
    fid = fopen(data,"w");
        for(int p=1;p<180;p++){
            fprintf(fid,"%f¥t%f¥n",x_zeta[p][0],y_zeta[p][0]);
        }
}

//流線関数の表示
void write_stream(){
    FILE *fid2;
    const char *data2={"streamline.txt"};
    fid2 = fopen(data2,"w");
    for(int i=1;i<180;i++){
        for(int j=0;j<100;j++){
            fprintf(fid2,"%f¥t%f¥t%f¥n",x_zeta[i][j],y_zeta[i][j],f_stream[i][j]);
      }
      fprintf(fid2,"¥n");
  }
}

//圧力の表示
void write_pressure(){
    FILE *fid3;
    const char *data3={"cp.txt"};
    fid3 = fopen(data3,"w");
    for(int i=1;i<180;i++){
```

```
      for(int j=0;j<100;j++){
        fprintf(fid3,"%f¥t%f¥t%f¥n",x_zeta[i][j],y_zeta[i][j],f_cp[i][j]);
     }
     fprintf(fid3,"¥n");
  }
}

//翼面上の圧力を表示
void write_surfacepressure(){
   FILE *fid4;
   const char *data4={"cp_surface.txt"};
   fid4 = fopen(data4,"w");
      for(int i=1;i<180;i++){
        fprintf(fid4,"%f¥t%f¥n",x_zeta[i][0],f_cp[i][0]);
      }
   fprintf(fid4,"¥n");
}

//-----------------------------------------空力中心計算

//空力中心計算用のデータの書き込み
void write_ac(double beta){
   FILE *fid5;
   const char *data5={"aerodynamic_center.txt"};
   fid5 = fopen(data5,"w");
   for(int i=0;i<5;i++){
      double alfa = i*2*(PI/180.0);
      calculation(alfa,beta);
      double x_sample = -0.9; //x=-0.9-0.9
      for(int j=0;j<18;j++){
        x_sample = x_sample + 0.1;
        double cm = cal_cm(x_sample);
        fprintf(fid5,"%f¥t%f¥n",x_sample/(4.0*c025),cm);
      }
      fprintf(fid5,"¥n");
   }
}

//参照点(x_sample,0)周りピッチングモーメントの計算
double cal_cm(double x_sample){
   double cm = 0.0;
   for(int i=1;i<179;i++){
      double dxw = x_zeta[i+1][0] - x_zeta[i][0];
      double dyw = y_zeta[i+1][0] - y_zeta[i][0];
      double dnx = dyw;
      double dny = -dxw;
      double cpm = (f_cp[i+1][0]+f_cp[i][0])/2.0;
      double fx = -cpm * dnx;
      double fy = -cpm * dny;
      cm = cm + (fy * (x_sample - x_zeta[i][0]) + fx * y_zeta[i][0]);
   }
   return cm;
}
```