Desarrollo y aplicación de técnicas de vorticidad a problemas de separación de flujo en perfiles

Apéndice B: Código computacional

Grado en Ingeniería en Vehículos Aeroespaciales

Trabajo Final de Grado

Alumno: Somoza Pérez, Gonzalo

Director: Ortega, Enrique
Tutor: Garcia Almiñana, Daniel

20-01-2016

Tabla de Contenidos

1 Código computacional **1**

1.1 Modelo Potencial 1

1.2 Modelo Separación de Flujo 7

2 Funciones **19**

2.1 Velocidad inducidad por un panel en un punto 19

2.2 Velocidad inducidad por un vórtice en un punto 21

2.3 Velocidad inducidad por un vórtice en un panel 23

2.4 Corrección de circulación de una matriz 25

1 Código computacional

1.1 Modelo Potencial

```
1
2  clear all
3  close all
4  clc
5
6  %INPUTS----------------------------------------------------------------
7
8
9  designation='2412' ; %iaf.designation = NACA 4 digit airfol designation (eg.
10 N=100; %no of panels
11 v_inf=1;%en m/s
12 alfa=3*pi/180. ;%en radianes
13 rho=1.;
14 c=1.;
15 %--------------------------------------------------------------------2
16
17 %1.GEOMETRY DISCRETIZATION---------------------------------------------
18 %[Key parameters-------------------------------------------------------]
19
20
21 m=str2double(designation(1))/100;
22 p=str2double(designation(2))/10;
23 t=str2double(designation(3:4))/100;
24
25 %[Panelation-----------------------------------------------------------
26 % Panel points generation
27
28
29 x=zeros(1,N+1);
30 z=zeros(1,N+1);
31 zt=zeros(1,N+1);
32 zc=zeros(1,N+1);
```

```
dzc_dxc=zeros(1,N+1);
theta=zeros(1,N+1);

beta=linspace(-pi,pi,N+1);
a0= 0.2969;a1=-0.1260;a2=-0.3516;a3= 0.2843;a4=-0.1036;
x1=0.5*(1-cos(beta(:)))';

for i=1:1:N+1
    zt(i)=5*t*((a0*sqrt(x1(i)))+(a1*x1(i))+(a2*x1(i)^2)+(a3*x1(i)^3)+(a4*x1(i
    if (x1(i)>=0 && x1(i)<p)
        zc(i)=(m/p^2)*((2*p*x1(i))-x1(i)^2);
        dzc_dxc(i)=((2*m)/((p)^2))*(p-x1(i));
    elseif (x1(i)>=p && x1(i)<1)
        zc(i)=(m/(1-p)^2)*(1-(2*p)+(2*p*x1(i))-(x1(i)^2));
        dzc_dxc(i)=((2*m)/((1-p)^2))*(p-x1(i));
    end
    theta(i)=atan(dzc_dxc(i));
    if i<=(N/2)
        z(i)=zc(i)-zt(i)*cos(theta(i));
        x(i)=x1(i)+zt(i)*sin(theta(i));
    elseif i>(N/2)
        z(i)=zc(i)+zt(i)*cos(theta(i));
        x(i)=x1(i)-zt(i)*sin(theta(i));
    end

end

%]

%[Panel geometrical data
xcp = zeros(2,N);
lp = zeros(1,N);
n_cp = zeros(2,N);
t_cp = zeros(2,N);

for i=1:N

```

```
70      dx = x(i+1)-x(i);
71      dz = z(i+1)-z(i);
72
73      lp(i) = sqrt(dx^2+dz^2) ;
74
75      n_cp(1,i) = -dz / lp(i);
76      n_cp(2,i) =  dx / lp(i);
77
78      t_cp(1,i) = dx / lp(i);
79      t_cp(2,i) = dz / lp(i);
80
81      xcp(1,i)=0.5*(x(i+1)+x(i));
82      xcp(2,i)=0.5*(z(i+1)+z(i));
83
84  end
85  % naca=str2double(designation);
86  % f1=figure(naca);
87  % plot(x,z)
88  % hold on
89  % plot(xcp(1,:),xcp(2,:),'or')
90  % quiver(xcp(1,:),xcp(2,:),n_cp(1,:),n_cp(2,:),'b')
91  % quiver(xcp(1,:),xcp(2,:),t_cp(1,:),t_cp(2,:),'k')
92  % axis equal
93
94  %-------------------------------------------------------------------------]
95  %-------------------------------------------------------------------------
96
97  %2.INDUCED VELOCITY-----------------------------------------------------
98
99  A=zeros(N);
100 RHS=zeros(N,1);
101
102 vcosa = v_inf*cos(alfa) ;   vsina = v_inf*sin(alfa) ;
103
104 inv_twopi = 1.0/(2.*pi) ;
105
106 for i=1:N
```

```
107
108     xi = xcp(1,i) ; zi = xcp(2,i) ;
109     txi = t_cp(1,i) ; tzi = t_cp(2,i);
110
111     for j=1:N
112
113         txj = t_cp(1,j) ; tzj = t_cp(2,j);
114         nxj = n_cp(1,j) ; nzj = n_cp(2,j);
115
116         x2 = (x(j+1)-x(j))*txj + (z(j+1)-z(j))*tzj ;
117         z2 = (x(j+1)-x(j))*nxj + (z(j+1)-z(j))*nzj ;
118
119         xj = (xi-x(j))*txj + (zi-z(j))*tzj ;
120         zj = (xi-x(j))*nxj + (zi-z(j))*nzj ;
121
122         delta = xj-x2 ;
123
124         u_p = inv_twopi*(atan2(zj,delta)-atan2(zj,xj));
125
126         w_p = 0.5*inv_twopi*log((delta^2+zj^2)/(xj^2+zj^2));
127
128         u = txj*u_p + nxj*w_p ;
129         w = tzj*u_p + nzj*w_p ;
130
131         A(i,j) = u*txi + w*tzi;
132
133     end
134
135     RHS(i)= -(vcosa*txi+vsina*tzi);
136
137 end
138
139  for i=1:N
140      A(i,i) = -0.5 ;  %enforce diagonal values to avoid numerical errors...
141  end
142
143 %[kutta condition
```

```
144
145 kp = floor(N/4)+1 ;  % select a kutta panel according to Katz
146
147 A(kp,:)=0. ; A(kp,1) = 1.0 ; A(kp,N) = 1.0 ;
148 RHS(kp)=0. ;
149 %-----------------------------------------------------------------------]
150 vort_int = A\RHS;
151 %-----------------------------------------------------------------------
152
153 vort_int(kp) = (vort_int(kp-1)+vort_int(kp+1))*0.5 ;  % correct the vorticity
154
155 %3.LOADS CALCULATION-----------------------------------------------------
156
157 Cp=1-((vort_int/v_inf).^2);
158
159 figure
160 plot(xcp(1,:),Cp)
161 axis ij;
162
163
164 Cz_p=zeros(N,1);
165 Cx_p=zeros(N,1);
166 Cmref=zeros(N,1);
167
168 for j=1:N
169
170     Cz_p(j)=Cp(j)*lp(j)*n_cp(2,j);
171     Cx_p(j)=Cp(j)*lp(j)*n_cp(1,j);
172     dx = x(j+1)-x(j);
173     dz = z(j+1)-z(j);
174     xref = 0.25*c;
175     zref =(t/0.2)*((a0*sqrt(xref))+(a1*xref)+(a2*xref^2)+(a3*xref^3)+(a4*xref
176     cm0z=n_cp(1,j)*(xcp(2,j)-zref);
177     cm0x=n_cp(2,j)*(xcp(1,j)-xref);
178     Cmref(j)=Cp(j)*lp(j)*(cm0z-cm0x);
179 end
180 Cz=(-1/c)*sum(Cz_p);
```

```
181 Cx=(-1/c)*sum(Cx_p);
182 Cm0=(-1/c^2)*sum(Cmref);
183
184 Cl=Cz*cos(alfa)-Cx*sin(alfa);
185 Cd=Cz*sin(alfa)+Cx*cos(alfa);
186 [num] = max(A(:));
187 [xmax , ymax] = ind2sub(size(A),find(A==num));
188
189 %-------------------------------------------------------------------------
```

1.2 Modelo Separación de Flujo

```
clear all
close all
clc


%[1]INPUT DATA & DATA PREPARATION-----------------------------------------------
designation='2412' ; %iaf.designation = NACA 4 digit airfol designation (eg.
N=100; %no of panels
v_inf=1.; %en m/s
alfa=3*pi/180. ;%en radianes
rho=1.0;
p_inf=0.;
c=1.;
Re=10^5;
visc_kin=(c*v_inf)/Re;

%[Geometry discretization-------------------------------------------------------
%Key parameters-----------------------------------------------------------------
m=str2double(designation(1))/100;
p=str2double(designation(2))/10;
t=str2double(designation(3:4))/100;
%Panel points generation--------------------------------------------------------
x=zeros(1,N+1);
z=zeros(1,N+1);
zt=zeros(1,N+1);
zc=zeros(1,N+1);
dzc_dxc=zeros(1,N+1);
theta=zeros(1,N+1);

beta=linspace(-pi,pi,N+1);
a0= 0.2969;a1=-0.1260;a2=-0.3516;a3= 0.2843;a4=-0.1036;
x1=0.5*(1-cos(beta(:)))';

for i=1:1:N+1
    zt(i)=5*t*((a0*sqrt(x1(i)))+(a1*x1(i))+(a2*x1(i)^2)+(a3*x1(i)^3)+(a4*x1(i
```

```
36      if (x1(i)>=0 && x1(i)<p)
37          zc(i)=(m/p^2)*((2*p*x1(i))-x1(i)^2);
38          dzc_dxc(i)=((2*m)/((p)^2))*(p-x1(i));
39      elseif (x1(i)>=p && x1(i)<1)
40          zc(i)=(m/(1-p)^2)*(1-(2*p)+(2*p*x1(i))-(x1(i)^2));
41          dzc_dxc(i)=((2*m)/((1-p)^2))*(p-x1(i));
42      end
43      theta(i)=atan(dzc_dxc(i));
44      if i<=(N/2)
45          z(i)=zc(i)-zt(i)*cos(theta(i));
46          x(i)=x1(i)+zt(i)*sin(theta(i));
47      elseif i>(N/2)
48          z(i)=zc(i)+zt(i)*cos(theta(i));
49          x(i)=x1(i)-zt(i)*sin(theta(i));
50      end
51  end
52  %-----------------------------------------------------------------------]
53  %[Panel geometrical data-----------------------------------------------
54  xcp = zeros(2,N);
55  lp = zeros(1,N);
56  n_cp = zeros(2,N);
57  t_cp = zeros(2,N);
58
59  for i=1:N
60      dx = x(i+1)-x(i);
61      dz = z(i+1)-z(i);
62      %panel length
63      lp(i) = sqrt(dx^2+dz^2) ;
64      %normal vector components
65      n_cp(1,i) = -dz / lp(i);
66      n_cp(2,i) =  dx / lp(i);
67      %tangential vector components
68      t_cp(1,i) = dx / lp(i);
69      t_cp(2,i) = dz / lp(i);
70      %Control points position
71      xcp(1,i)=0.5*(x(i+1)+x(i));
72      xcp(2,i)=0.5*(z(i+1)+z(i));
```

```
end
naca=str2double(designation);

%-------------------------------------------------------------------------

nstep_fin = 100;  % number of time steps

dt = 0.05;%0.075; %mean(lp)/v_inf;    % time increment

eps = mean(lp)/4;

sfact = min(lp)*0.25; %eps*0.5; %min(lp) ; % safety factor to compute induced

Nvmax = 15*N   ;  % maximum free vortices to be stored (this parameter should

%[2]Coupling coefficients-----------------------------------------------------
A=zeros(N); RHS_freeflow=zeros(N,1);
vcosa = v_inf*cos(alfa) ;  vsina = v_inf*sin(alfa) ;
for i=1:N
    xi=xcp(1,i);zi=xcp(2,i);
    txi=t_cp(1,i);tzi=t_cp(2,i);
    for j=1:N
        xj_ini=x(j);xj_fin=x(j+1);zj_ini=z(j);zj_fin=z(j+1);
        txj=t_cp(1,j);tzj=t_cp(2,j);nxj=n_cp(1,j);nzj=n_cp(2,j);
        [uij,wij]=vind_pan2point_1(xi,zi,xj_ini,zj_ini,xj_fin,zj_fin,txj,tzj,
        A(i,j)=uij*txi + wij*tzi;
        if i==j
            A(i,j)=-0.5;
        end
    end
    RHS_freeflow(i)= -(vcosa*txi+vsina*tzi);
end

AK = A;

%-------------------------------------------------------------------------
%THIS IS ONLY FOR THE STATIONARY POTENTIAL SOLUTION
```

```
%kutta condition
kp =floor(N/4)+1 ;  % select a kutta panel according to Katz
AK(kp,:)=0.; AK(kp,1) = 1.0 ; AK(kp,N) = 1.0 ;
RHS=RHS_freeflow; RHS(kp)=0. ;
%system solution
vort_int = AK\RHS;
vort_int(kp) = (vort_int(kp-1)+vort_int(kp+1))*0.5 ;  % correct the vorticity
%pressure and forces computation
cfx = 0.0 ; cfz = 0.0 ;
for i=1:N
    Cp(i) = 1.0 - vort_int(i)^2 ;   % assumes unit freestream velocity
    cfx = cfx - Cp(i)*lp(i)*n_cp(1,i) ;  % force contributions, assumes unit
    cfz = cfz - Cp(i)*lp(i)*n_cp(2,i) ;
end
Cl = cfz*cos(alfa)-cfx*sin(alfa) ;
Cd = cfz*sin(alfa)+cfx*cos(alfa) ;   % this must tend to zero
%plot(xcp(1,:),Cp(:))

Cl_steady = Cl

%-------------------------------------------------------------------------

% A matrix correction to enforce zero circulation(panels)

A1 = A ;

[A1] = matrix_columncorrection_1(A1,lp);

for i=1:N
    for j=1:N
        A_lp(i,j)=A1(i,j)+lp(j);
    end
end

%-------------------------------------------------------------------------
% Solve the system for gamma

```

```
147 vort_int = A_lp\RHS_freeflow ;
148
149 check = dot(vort_int,lp)   % vorticity check
150
151 %-------------------------------------------------------------------------
152 %-------------------------------------------------------------------------
153
154 xv = zeros(2,Nvmax) ;  % free_vortex positions
155 d_gamma = zeros(Nvmax,1) ; % circulations
156 aero_forces = zeros(2,nstep_fin); % to store aerodynamic forces
157
158 time = linspace(dt,dt*nstep_fin,nstep_fin);
159
160 % figure(1)  % geometry
161 % hold on
162 % axis equal
163 figure(2)  % lift coefficient
164 hold on
165
166 Nv = 0; current_time = 0.; vort_rem=0.;
167
168 for nstep = 1:nstep_fin
169
170     current_time = current_time + dt ;
171
172     if ( nstep > 1 )  % move previous vortices
173
174         aux = zeros(3,Nv) ; % auxiliar storage
175         aux(1,1:Nv) = xv(1,1:Nv) ;
176         aux(2,1:Nv) = xv(2,1:Nv) ;
177         aux(3,1:Nv) = d_gamma(1:Nv) ;
178         j = 0;
179         for k = N+1:N+Nv   % copy value in new position
180             j = j + 1;
181             xv(1,k)=aux(1,j);
182             xv(2,k)=aux(2,j);
183             d_gamma(k)=aux(3,j);
```

```
184         end
185     end
186
187     for k = 1:N   % generate new vortices
188         xv(1,k)=xcp(1,k)+ eps*n_cp(1,k);
189         xv(2,k)=xcp(2,k)+ eps*n_cp(2,k);
190         d_gamma(k)=vort_int(k)*lp(k);
191     end
192
193     Nv = min([Nv+N,Nvmax]) ;  % update # of free vortices
194
195     % Convection ------------------------------------------------------------
196
197     v_ind = zeros(2,Nv) ;  % induced velocities
198     xv_o = xv;
199
200     for conv_it = 1:2
201
202         for i=1:Nv
203
204             xi = xv(1,i) ; zi = xv(2,i) ;
205
206             u_i = 0.0 ; w_i=0.0 ;
207
208             for j=1:N % loop over the airfoil panels
209                 xj_ini=x(j);xj_fin=x(j+1);zj_ini=z(j);zj_fin=z(j+1);
210                 txj=t_cp(1,j);tzj=t_cp(2,j);nxj=n_cp(1,j);nzj=n_cp(2,j);
211                 pan_int=vort_int(j);pan_long=lp(j);
212                 [uij,wij]=vind_pan2point_1(xi,zi,xj_ini,zj_ini,xj_fin,zj_fin,
213                 u_i = u_i + pan_int*uij ;
214                 w_i = w_i + pan_int*wij ;
215             end
216
217             for k=N+1:Nv %1:Nv % loop over the free vortices
218                 xk = xv(1,k) ; zk = xv(2,k) ; dgk = d_gamma(k);
219                 [uik,wik]=vind_vort2point_1(xi,zi,xk,zk,sfact);
220                 u_i = u_i + dgk*uik;
```

```
221                 w_i = w_i + dgk*wik;
222             end
223
224             v_ind(1,i) = u_i + vcosa ;
225             v_ind(2,i) = w_i + vsina ;
226
227         end
228
229          for i = 1:N % test!! forces tangent velocities near the surface
230              vn = dot(v_ind(:,i),n_cp(:,i));
231              v_ind(:,i)=v_ind(:,i)-n_cp(:,i)*vn;
232          end
233
234         for i = 1:Nv
235             xv(:,i) = xv(:,i) + v_ind(:,i)*dt*0.5 ; % update position
236         end
237
238     end % convection iterations
239
240 %     figure;
241 %     plot(x(:),z(:),'-',xv_o(1,1:Nv),xv_o(2,1:Nv),'ok')
242 %     hold on
243 %     axis equal
244 %     quiver(xv_o(1,1:Nv),xv_o(2,1:Nv),v_ind(1,1:Nv),v_ind(2,1:Nv))
245 %     plot(xv(1,1:Nv),xv(2,1:Nv),'or')
246 %     aa=0;
247
248     % Diffusion ----------------------------------------------------------
249
250     P=zeros(1,Nv); Q=zeros(1,Nv);
251     Ad=1.01316;
252     P_r=rand; Q_r=rand;
253     P1=(Ad+P_r)^5;
254     P(1)=P1-floor(P1);
255     Q1=(Ad+Q_r)^5;
256     Q(1)=Q1-floor(Q1);
257     for i=2:Nv
```

```
258         P_c=(Ad+P(i-1))^5;
259         P(i)=P_c-floor(P_c);
260         Q_c=(Ad+Q(i-1))^5;
261         Q(i)=Q_c-floor(Q_c);
262     end
263     xv_d = zeros(2,Nv);
264     for i=1:Nv
265         xi=xv(1,i); zi=xv(2,i);
266         dO_i=2*pi*Q(i);
267         dr_i=(4*visc_kin*dt*log(1/P(i)))^.5;
268         xv_d(1,i)=xi+(dr_i*cos(dO_i));
269         xv_d(2,i)=zi+(dr_i*sin(dO_i));
270     end
271
272     for i = 1:Nv
273         xv(:,i) = xv_d(:,i);
274     end
275
276 %     close all
277 %     plot(x(:),z(:),'-',xv(1,1:Nv),xv(2,1:Nv),'ok')
278 %     axis equal
279
280     % Eliminate inner vortices ----------------------------------------
281
282     vort_in = zeros(1,Nv);
283
284     for j=1:Nv
285         xj=xv(1,j); zj=xv(2,j);
286         if (xj>=0 && xj<=1.)
287             zt_xj=5*t*(a0*xj^.5 + a1*xj + a2*xj^2 + a3*xj^3 + a4*xj^4);
288             if xj<p
289                 zc_xj=(m*(2*p*xj-xj^2))/p^2;
290                 dzc_dxc=(2*m*(p-xj))/p^2;
291             elseif xj>=p
292                 zc_xj=(m*(1-2*p+2*p*xj-xj^2))/(1-p)^2;
293                 dzc_dxc=(2*m*(p-xj))/(1-p)^2;
294             end
```

```
295             theta_xj=atan(dzc_dxc);
296             zext_xj=zc_xj+zt_xj*cos(theta_xj);
297             zint_xj=zc_xj-zt_xj*cos(theta_xj);
298             if (zj>=zint_xj && zj<=zext_xj)    % mark inner vortices
299                 vort_in(j)=1;
300                 vort_rem = vort_rem + d_gamma(j);
301             end
302         end
303     end
304
305     jout = 0;
306     for k = 1:Nv
307         in = vort_in(k);
308         if in < 1  % skips deleted vortices
309             jout = jout + 1;
310             xv(:,jout) = xv(:,k);
311             d_gamma(jout) = d_gamma(k);
312         end
313     end
314
315     Nv = jout;  % updates counter
316
317 %    figure(1);
318 %    plot(x(:),z(:),'-k',xv(1,1:Nv),xv(2,1:Nv),'o','Markersize',1.5)
319
320     % Contribution of the free vortices to the RHS ----------------------
321
322     RHS = RHS_freeflow ;
323
324     for k = 1:Nv
325
326         xk = xv(1,k) ; zk = xv(2,k) ; dgk = d_gamma(k);
327
328         for i=1:N % loop over the panels
329             xi=xcp(1,i);zi=xcp(2,i);
330             [uik,wik]=vind_vort2point_1(xi,zi,xk,zk,sfact);
331             auxvec(i)= uik*t_cp(1,i)+wik*t_cp(2,i);
```

```
332         end
333
334         maxval = -1.e6 ;
335         for i = 1:N
336             if ( abs(auxvec(i))>maxval & i~=j )
337                 maxval = abs(auxvec(i)) ;
338                 maxpos = i ;
339             end
340         end
341         value = auxvec(maxpos);
342         auxvec(maxpos) = 0.0 ; circsum = 0.0 ;
343         for i = 1:N
344             circsum = circsum + auxvec(i)*lp(i) ;
345         end
346         auxvec(maxpos) = -circsum / lp(maxpos) ;
347
348         RHS(1:N) = RHS(1:N) - auxvec(1:N).'*dgk ;  % add contributions of the
349
350     end
351
352     old_vort = vort_int;
353
354     vort_int = A_lp\RHS;  % solve for the vortex strengths
355
356     check = dot(vort_int,lp)  % vorticity conservation
357
358     % Pressure and forces computation using unsteady method
359
360     ps_1(1) = -rho*vort_int(1)*lp(1)/dt;
361
362     for j=2:N
363         ps_1(j)= ps_1(j-1) - rho*vort_int(j)*lp(j)/dt;
364     end
365
366     ps_max = -1.e6;imax=1;
367     for j = 1:N
368         if (xcp(1,j)<0.2)  % searches near the leading edge
```

```
369                 if ( ps_1(j)>ps_max )
370                     ps_max=ps_1(j);
371                     imax=j;
372                 end
373             end
374         end
375
376         ps_max = ps_1(imax);
377         qinf = 0.5*rho*v_inf^2; p0 = p_inf + qinf;
378         dp = p0 - ps_max; % correction to match the total pressure at the stagnat
379
380         ps_1(:) = ps_1(:) + dp ;
381
382         ps_1(1) = ps_1(N) ; % equalizes pressures at the trailing edge  ?
383
384         for j = 1:N
385             Cp(j) = (ps_1(j)-p_inf)/qinf ;
386         end
387
388         cfx = 0.0 ; cfz = 0.0 ;
389         for i=1:N
390             cfx = cfx - Cp(i)*lp(i)*n_cp(1,i) ;  % force contributions, assumes u
391             cfz = cfz - Cp(i)*lp(i)*n_cp(2,i) ;
392         end
393
394         aeroforces(1,nstep) = cfz*cos(alfa)-cfx*sin(alfa) ;      % lift coefficien
395         aeroforces(2,nstep) = cfz*sin(alfa)+cfx*cos(alfa) ;    % drag
396
397         figure(2)
398         plot(time(1:nstep),aeroforces(1,1:nstep),'-o')
399
400 end % time steps
401
402
403
404 figure (4)
405 plot(x(:),z(:),'-k')
```

```
406 hold on
407 plot(xv(1,1:Nv),xv(2,1:Nv),'o','Markersize',1.5)
408 axis equal
```

2 Funciones

2.1 Velocidad inducidad por un panel en un punto

```
function [u,w]=vind_pan2point_1(x,z,xpan_ini,zpan_ini,xpan_fin,zpan_fin,txpan
inv_twopi = 0.159154943091895;  %1/(2*pi);
range = eps; %long_pan*0.1;
x2=(xpan_fin-xpan_ini)*txpan+(zpan_fin-zpan_ini)*tzpan;
z2=(xpan_fin-xpan_ini)*nxpan+(zpan_fin-zpan_ini)*nzpan;
xp_pan=(x-xpan_ini)*txpan+(z-zpan_ini)*tzpan;
zp_pan=(x-xpan_ini)*nxpan+(z-zpan_ini)*nzpan;
rp2 = xp_pan^2+zp_pan^2;
if ( rp2>range*range )  % compares squared lengths to avoid sqrt
    delta=xp_pan-x2;
    u_pan=inv_twopi*(atan2(zp_pan,delta)-atan2(zp_pan,xp_pan));
    w_pan=0.5*inv_twopi*log((delta^2+zp_pan^2)/(xp_pan^2+zp_pan^2));
    u = u_pan*txpan+w_pan*nxpan;
    w = u_pan*tzpan+w_pan*nzpan;
else
    u=0.; w=0.;
end
end
```

2.2 Velocidad inducidad por un vórtice en un punto

```
function [u,w] = vind_vort2point_1(x,z,xvort,zvort,dr_lim)
inv_twopi = 0.159154943091895;  %1/(2*pi);
r1 = (x-xvort);r2=(z-zvort);
rk2 = r1*r1+r2*r2;
if rk2 > dr_lim*dr_lim    % compares squared lengths to avoid sqrt
    rk2 = 1.0/rk2 ;
    u = inv_twopi*r2*rk2;
    w = -inv_twopi*r1*rk2;
else
    u=0.; w=0.;
end
end
```

2.3 Velocidad inducidad por un vórtice en un panel

```
function [u,w]=vind_vort2pan_1(x,z,xvort,zvort,long_pan,x_ini,z_ini,tx_pan,tz
inv_twopi=1/(2*pi);
r1=x-xvort; r2=z-zvort;
r1_l= r1*tx_pan + r2*tz_pan; r2_l=r1*nx_pan + r2*nz_pan;
if(abs(r1_l)>(0.5*long_pan))
    param=2;
else
    param=abs(r2_l/long_pan);
end;

if ((param>=(0.4))&&(param<(1.0)))
    rk=sqrt((r1)^2+(r2)^2);
    ns= 1 +round(2*long_pan/rk);
    u_ikcont=0.;w_ikcont=0.;
    for n=1:ns
        dl=(2*n -1)*long_pan/(2*ns);
        xn=x_ini +dl*tx_pan;
        zn=z_ini +dl*tz_pan;
        r_1=xn-xvort;r_2=zn-zvort;
        rkn2=(r_1)^2+(r_2)^2;
        u_ikcont=u_ikcont+r_2/rkn2;
        w_ikcont=w_ikcont-r_1/rkn2;
     end
     u=inv_twopi*u_ikcont/ns;
     w=inv_twopi*w_ikcont/ns;
elseif (param<(0.4))
    u_aux=0.5*inv_twopi/r2;
    u=u_aux*tx_pan;
    w=u_aux*tz_pan;
else
    rk2=(r1)^2+(r2)^2;
    u=inv_twopi*r2/rk2;
    w=-inv_twopi*r1/rk2;
end
end
```

2.4 Corrección de circulación de una matriz

```
function [matrix_c]=matrix_columncorrection_1(matrix,pan_lenght)
matrix_c=matrix;
[N1,N2]=size(matrix);
matrix_max=max(matrix,[],1);
for m=1:N2
    max_value=matrix_max(m);
    p_max=find(matrix(:,m)==max_value);
    sum=0.;
    for p=1:N1
        mat_pm=matrix(p,m);
        lp_p=pan_lenght(p);
        if p==p_max
            continue,
        end
        sum=sum+mat_pm*lp_p;
    end
    matrix_c(p_max,m)=-sum/pan_lenght(p_max);
end


end
```