APPENDICE B

Procedura di calcolo delle traiettorie di inseguimento

Di seguito riporto alcune parti del codice relativo alla procedura di calcolo delle traiettorie di inseguimento data una traiettoria di riferimento. Le considerazioni di principio sono già state esposte nel § 4.5.3.

```
function calcolatraj(handles,valw,eventdata)


%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Aggiorna lo stato di avanzamento della wait bar e ridisegna la terna    %
% sulla traiettoria da inseguire                                          %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%


ph = waitbar(valw,'Please wait...','Name','Drawing follower trajectory...','Color',[0.92 0.91 0.84]);
ph1=get(ph,'Children');
ph2=get(ph1,'Children');
set(ph2(2),'FaceColor',[0 0 1]);
set(ph2(2),'EdgeColor',[0 0 1]);

newpoint=getappdata(handles.Main,'newpoint');
if ishandle(newpoint)
    delete(newpoint);
end

axesobj=getappdata(handles.Main,'axesobj');
if ishandle(axesobj)
    delete(axesobj);
end

followingpoint=getappdata(handles.Main,'Selectedpoints');
[n,m]=size(followingpoint);

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Acquisisce le caratteristiche dei punti della traiettoria da inseguire  %
% X,Y,Z -> posizione                                                      %
% Roll,Pitch, Yaw, Ra -> orientazione                                     %
% V -> velocità                                                           %
% T -> tipo di elemento base di appartenenza (Line=1 , Circle=0)          %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
```

```
X=followingpoint(1,:);
Y=followingpoint(2,:);
Z=followingpoint(3,:);
V=followingpoint(4,:);
T=followingpoint(5,:);
Roll=followingpoint(6,:);
Pitch=followingpoint(7,:);
Yaw=followingpoint(8,:);
Ra=followingpoint(9,:);
matriceterne=[];
t=0;
dimvel=size(Z);
for i=1:dimvel(2)-1

    deltad=sqrt((X(i+1)-X(i))^2+(Y(i+1)-Y(i))^2+(Z(i+1)-Z(i))^2);
    deltav=(V(i+1)+V(i))*10^3/7200;
    t2=(deltad/deltav)+t(i);
    t=[t,t2];

end
tempoprec=t2;
XYZ1=zeros(3,m-2);
XYZ=zeros(3,m-2);
XYZ=[X(2:m-2);Y(2:m-2);Z(2:m-2)];
AirShow=getappdata(handles.Main,'allairshow');
manovra=AirShow.Manovre(getappdata(handles.Main,'numvel'),getappdata(handles.Main,'numfig')).Fig;
if manovra.Numeroblocchi>0
    point=[];
    prop=manovra.Listaproprieta;
    i=manovra.Numeroblocchi;
    point=prop(i).Puntixyz;
    p=point(:,end);
    set(handles.vista3d,'Nextplot','add');
    plot3(p(1,1),p(2,1),p(3,1),'r.','Markersize',33,'Parent',handles.vista3d);
    xterna=[X(2),Y(2),Z(2)]-[X(1),Y(1),Z(1)];



    xternasuc=[X(3),Y(3),Z(3)]-[X(2),Y(2),Z(2)];
    yterna=cross(xterna,xternasuc);
    zterna=cross(xterna,yterna);
    yterna=yterna/sqrt(yterna(1)^2  + yterna(2)^2 + yterna(3)^2);
    xterna=xterna/sqrt(xterna(1)^2  + xterna(2)^2 + xterna(3)^2);
    zterna=zterna/sqrt(zterna(1)^2  + zterna(2)^2 + zterna(3)^2);
    ptraslato=p-transpose([X(1),Y(1),Z(1)]);
    sistemariferimento=[transpose(xterna),transpose(yterna),transpose(zterna)];
    coordp=inv(sistemariferimento)*ptraslato;
    set(handles.deltax0,'String',coordp(1,1));
    set(handles.deltay0,'String',coordp(2,1));
    set(handles.deltaz0,'String',coordp(3,1));
else
    xterna=[X(2),Y(2),Z(2)]-[X(1),Y(1),Z(1)];
    az=atan2(xterna(2),xterna(1));
    el=-atan2(xterna(3),sqrt(xterna(2)^2  + xterna(1)^2));
    matricerotz=[[cos(az),-sin(az),0];...
            [sin(az),cos(az),0];...
            [0,0,1]];

    matriceroty=[[cos(el) , 0 , sin(el)];...
            [    0    , 1 ,     0   ];...
            [-sin(el), 0 , cos(el)]];

    mat=matricerotz*matriceroty*[[1;0;0],[0;1;0],[0;0;1]];
    xterna=[mat(1,1),mat(2,1),mat(3,1)];
    yterna=[mat(1,2),mat(2,2),mat(3,2)];
    zterna=cross(xterna,yterna);

    set(handles.deltax0,'Enable','off');
    set(handles.deltay0,'Enable','off');
    set(handles.deltaz0,'Enable','off');
```

-
-
-

.

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Ad ogni punto della traiettoria  da inseguire viene associato una terna di riferimento
% la cui orientazione è influenzata dai vincoli di continuità
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%


for i=2:m-1
    ph=waitbar(valw + i*(1-valw)/m);
    p=transpose(matriceterne(i-1).y);
    yterna=matriceterne(i-1).y;
    yterna=yterna/sqrt(yterna(1)^2  + yterna(2)^2 + yterna(3)^2);
    xterna=[X(i+1),Y(i+1),Z(i+1)]-[X(i),Y(i),Z(i)];
    xterna=xterna/sqrt(xterna(1)^2  + xterna(2)^2 + xterna(3)^2);
    zterna=cross(xterna,yterna);
    yterna=cross(zterna,xterna);
    x=xterna;
    y=yterna;
    z=zterna;
    terna=struct ('x',x,...
        'y',y,...
        'z',z);
    matriceterne=[matriceterne,terna];
end
matriceterne=[matriceterne,matriceterne(end)];
setappdata(handles.Main,'matriceterne',matriceterne);


%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Vengono valutati i parametri di inseguimento impostati
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%


if isempty(str2num(get(handles.deltax0,'String')))
    deltax0=0;
else
```

.

.

.

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% calcolo dei punti X Y Z della traiettoria di inseguimento
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
    for i=1:m

        xa=transpose(deltax(i)*matriceterne(i).x);
        ya=transpose(deltay(i)*matriceterne(i).y);
        za=transpose(deltaz(i)*matriceterne(i).z);
        O=[X(i);Y(i);Z(i)];
        XYZ1(:,i)=O+xa+ya+za;

    end

catch
    errordlg('Error setting following parameters','Parameters error')
    return
    setappdata(handles.Main,'OK',0);
end

dimvel=size(V);
X=XYZ1(1,:);
Y=XYZ1(2,:);
Z=XYZ1(3,:);
t2=floor(t2*10)/10;
ts=0:0.1:t2;
set(handles.vista3d,'Nextplot','add');
newpoint=[];
for i=2:m-1
    h=plot3(XYZ1(1,i),XYZ1(2,i),XYZ1(3,i),'b.','LineWidth',2,'Parent',handles.vista3d);
    set(h,'ButtonDownFcn','Followvelivol(''ButtonDownpoint_callback'',gcbo,[],guidata(gcbo))');
    newpoint=[newpoint,h];

end
setappdata(handles.Main,'newpoint',newpoint);
proprieta=getappdata(handles.Main,'Proprieta');
```

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% La velocità della traiettoria di inseguimento viene adattata in modo da
% sincronizzare i leader e wingman
% Per traiettorie paricolari l'accelerazione potrebbe essere elevata per
% cui è necessario limitare la derivata della velocità e adeguarla ai
% parametri dell'inviluppo di volo
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%


vettorevelocita=vettorevelocita*3600/1000;
tp=0;
for i=1:dimvel(2)-1


    deltad=sqrt((X(i+1)-X(i))^2+(Y(i+1)-Y(i))^2+(Z(i+1)-Z(i))^2);
    deltav=(vettorevelocita(i+1)+vettorevelocita(i))*10^3/7200;
    t2=(deltad/deltav)+tp(i);
    tp=[tp,t2];

end

dimv=size(vettorevelocita);
dim=size(X);
dimv=size(vettorevelocita);
dim=size(Y);
diml=size(vettorevelocita);
setappdata(handles.Main,'X',X);
setappdata(handles.Main,'Y',Y);
setappdata(handles.Main,'Z',Z);
dim=size(X);

for i=1:dim(2)-1
    if int16(atan2( (Y(i+1)-Y(i)),(X(i+1)-X(i)))*180/pi)<=-179 || int16(atan2( (Y(i+1)-Y(i)),(X(i+1)-X(i)))*180/pi)>=179
        psi(i)=180;
    else


          psi(i)=atan2( (Y(i+1)-Y(i)),(X(i+1)-X(i)))*180/pi;
      end
      int16(atan2( (Y(i+1)-Y(i)),(X(i+1)-X(i)))*180/pi);

      psin(i)=i;
  end

  for i=1:dim(2)-1
      theta(i)=-atan2( (Z(i+1)-Z(i)),sqrt((Y(i+1)-Y(i)).^2+(X(i+1)-X(i)).^2))*180/pi;
      thetan(i)=i;
  end

  phi=Roll;
  for i=1:dim(2)-2
      if round(phi(i))~=psi(i)

          if theta(i)<-45

              if round(psi(i+1))==180
                  phi(i+1)=phi(i);
              end

          elseif theta(i)>45

              if round(psi(i+1))==0
                  phi(i+1)=phi(i);
              end
          end
      end
  end

  psi=[psi,psi(dim(2)-1)];
  theta=[theta,theta(dim(2)-1)];
  setappdata(handles.Main,'psiangle',psi);
  setappdata(handles.Main,'thetangle',theta);
```

```
% ************************************************************************
% Nel caso l'accellerazione del velivolo supera ii valori consentiti viene
% ricalcolata la velocità attraverso la procedura limitdvelocity(....)
% ************************************************************************


if boolean==1
    vettorevelocita=limitdvelocity(handles,vettorevelocita,phi,theta,Ra,7)
end
tp=0;
dist=0;
for i=1:dimvel(2)-1


    deltad=sqrt((X(i+1)-X(i))^2+(Y(i+1)-Y(i))^2+(Z(i+1)-Z(i))^2);
    dist=dist+deltad;
    deltav=(vettorevelocita(i+1)+vettorevelocita(i))*10^3/7200;
    t2=(deltad/deltav)+tp(i);
    tp=[tp,t2];

end

Vm=dist/t2;
Vm1=dist/tempoprec;

set(handles.endtime,'String',tempoprec);
dvm=(Vm1-Vm)/1000*3600;
vettorevelocita=vettorevelocita+dvm;
tp=0;
dist=0;
```

```
% ************************************************************************************************
% Vengono aggiornate le proprietà dell'elemento follow block
% ************************************************************************************************
proprieta.Puntixyz=[X;Y;Z];
proprieta.Vettorevelocita=vettorevelocita;
proprieta.AngoliRPY=real([psi;theta;phi]);
proprieta.AngoloRoll=real(Ra);
proprieta.inpos.y=Y(1);
proprieta.inpos.z=Z(1);
proprieta.outpos.x=X(dim(2));
proprieta.outpos.y=Y(dim(2));
proprieta.outpos.z=Z(dim(2));
proprieta.outvel=vettorevelocita(end);
proprieta.invel=vettorevelocita(1);
continuita=getappdata(handles.Main,'Continuitaout');
continuita.psi=0;
continuita.theta=theta(dim(2));
continuita.phi=atan2( (Y(end)-Y(end-1)),(X(end)-X(end-1)));
continuita.x1=X(dim(2)-1);
continuita.y1=Y(dim(2)-1);
continuita.z1=Z(dim(2)-1);
continuita.x2=X(dim(2));
continuita.y2=Y(dim(2));
continuita.z2=Z(dim(2));
setappdata(handles.Main,'Proprieta',proprieta);
setappdata(handles.Main,'Continuitaout',continuita);
set(handles.vista3d,'Nextplot','replacechildren');
waitbar(1);
close(ph);
setappdata(handles.Main,'pointselected',dim(2));
set(handles.phislider,'Max',dim(2));
setappdata(handles.Main,'Vectorphi',[1,dim(2)]);
setappdata(handles.Main,'vectorvel',[1,dim(2)]);
```

La funzione che adegua la velocità ai parametri dell'inviluppo di volo

```
function out=limitdvelocity(handles,velocity,phi,gamma,roll,G)

phi=phi/180*pi;
gamma=gamma/180*pi;
roll=roll/180*pi;
dimphi=size(phi);
dimgamma=size(gamma);
dimroll=size(roll);
X=getappdata(handles.Main,'X');
Y=getappdata(handles.Main,'Y');
Z=getappdata(handles.Main,'Z');
dimx=size(X);
dimy=size(Y);
dimz=size(Z);
dimvel=size(velocity);
t=0;

for i=1:dimvel(2)-1
    deltad=sqrt((X(i+1)-X(i))^2+(Y(i+1)-Y(i))^2+(Z(i+1)-Z(i))^2);
    deltav=(velocity(i+1)+velocity(i))*10^3/7200;
    t2=(deltad/deltav)+t(i);
    t=[t,t2];
end
dgamma=diff(gamma)./diff(t);
dgamma=[dgamma,dgamma(end)];
dimdgamma=size(dgamma)
n=cos(gamma).*cos(phi+roll) + (dgamma.*velocity*1000/3600).*cos(phi)/9.81;

%%%%%%%%%%%%%%%%%%%% Limita la velocità in caso non rispetti l'inviluppo di volo%%%%%

parametri=getappdata(handles.Main,'parametrivelivolo');
vsv=getappdata(handles.Main,'vsv');
nsn=getappdata(handles.Main,'nsn');


 nsn=getappdata(handles.Main,'nsn');
 ngn=getappdata(handles.Main,'ngn');
 vgv=getappdata(handles.Main,'vgv');

 dimvel=size(velocity);


    vstallo=ppval(vsv,1);%%%%%%%%%%%%%%
    velout=[];
    for i=1:dimvel(2)
        if velocity(i)>vstallo
            if velocity(i)<parametri(2)
                maxlf=ppval(nsn,velocity(i));%%%%%%%%%%%%
                minlf=ppval(ngn,velocity(i));%%%%%%%%%%%%%%
                if n(i)<maxlf && n(i)>minlf
                    velout(i)=velocity(i);
                else
                    if n(i)>0
                        velout(i)=ppval(vsv,maxlf);%%%%%%%%%%%%
                    else
                        velout(i)=ppval(vgv,minlf);%%%%%%%%%%%%%%%%%%%%

                    end
                end
            else
                maxlf=parametri(4);
                minlf=parametri(5);
                velout(i)=velocity(i);
            end
        else
            velout(i)=vstallo;
        end
        if velout(i)>parametri(3)
            velout(i)=parametri(3);
```

```
        end
    end
velocity=velout;
T=parametri(6)*4;
D=parametri(6);
dvdtmax=((ones(1,dimgamma(2))*(T-D)/parametri(6))-(sin(gamma)))*9.81;
dvdtmin=((ones(1,dimgamma(2))*(-D)/parametri(6))-(sin(gamma)))*9.81;
dvdtmax=((ones(1,dimgamma(2))))*G*9.81;
dvdtmin=-((ones(1,dimgamma(2))))*G*9.81;
velocity=velocity*1000/3600;
 dvdt=diff(velocity)./diff(t);
dvdt=[dvdt,dvdt(end)];
    for i=1:dimvel(2)-1

        if dvdt(i)>0
            if dvdt(i)>dvdtmax(i)

                                disp('positive')
                                velocity(i+1)=velocity(i)+ dvdtmax(i)*(t(i+1)-t(i));
             end

         else
             if dvdt(i)<dvdtmin(i)
                                disp('negative')
                                velocity(i+1)=velocity(i)+  dvdtmin(i)*(t(i+1)-t(i));
             end
         end
         dvdt=diff(velocity)./diff(t);
    end
 dvdt=diff(velocity)./diff(t)

velocity=velocity*3600/1000;
out=velocity;
```