Capitolo 5

Rendering air show e verifica della sincronizzazione

5.1 Introduzione

Il Rendering 3D dell'evoluzioni acrobatiche costituisce uno degli aspetti cruciali dell'applicazione realizzata. L'ambiente di sviluppo MATLAB non offre molte possibilità per la visualizzazione 3D arricchita con oggetti e scenari realistici. È stato necessario prevedere, come output dell'applicazione, un file che alimentasse i canali di rendering di un'applicazione, in grado di assicurare una visualizzazione 3D di modelli realistici dei velivoli nonché degli scenari in cui ha luogo l'air show.

Il modulo di rendering messo a disposizione dell'applicazione costituisce, pur nella sua semplicità, un valido supporto alla verifica della sincronizzazione dei velivoli, lasciando a Dyna World il compito di rendere più realistica la visualizzazione 3D.

Il problema che si è dovuto superare per permettere una corretta visualizzazione 3D, è stato quello di ridefinire le traiettorie con il vincolo di essere valutate ad istanti temporali multipli. Le traiettorie definite attraverso la sequenza dei blocchi base, sono costituite da punti [*x,y,z,roll,pitch,yaw,R,G,B,on/off*] riferiti ad istanti temporali non multipli.

5.2 Ridefinizione delle traiettorie

Ridefinire le traiettorie in modo da specificarle ad istanti temporali multipli, permette di :

- Verificare la sincronizzazione
- Produrre un rendering sincronizzato degli oggetti in movimento

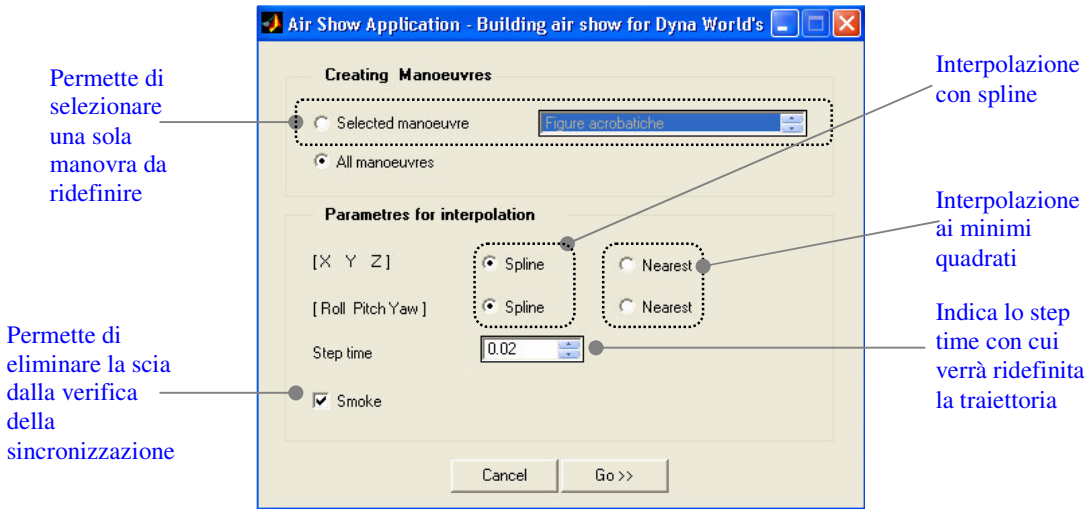
In generale, specificare la traiettoria di un oggetto, significa definirne posizione ed orientazione nel tempo. Nel caso il rendering sia interessato da più oggetti in movimento, definire le traiettorie ad istanti non multipli e non con lo stesso fattore moltiplicativo per tutti gli oggetti, vuol dire produrre un rendering non sincronizzato.

Nel caso in cui la durata delle traiettorie non coincida, la procedura di ridefinizione tiene conto della minima tra queste e fallisce nel caso la traiettoria di uno o più velivoli non sia definita *(anche parzialmente)*.

La procedura di verifica della sincronizzazione permette di verificare anche una sola figura acrobatica, nel caso le altre non siano state ancora definite.

L'utente immette alcuni parametri per la procedura di ridefinizione come si vede nella figura 5.1

Figura 5.1

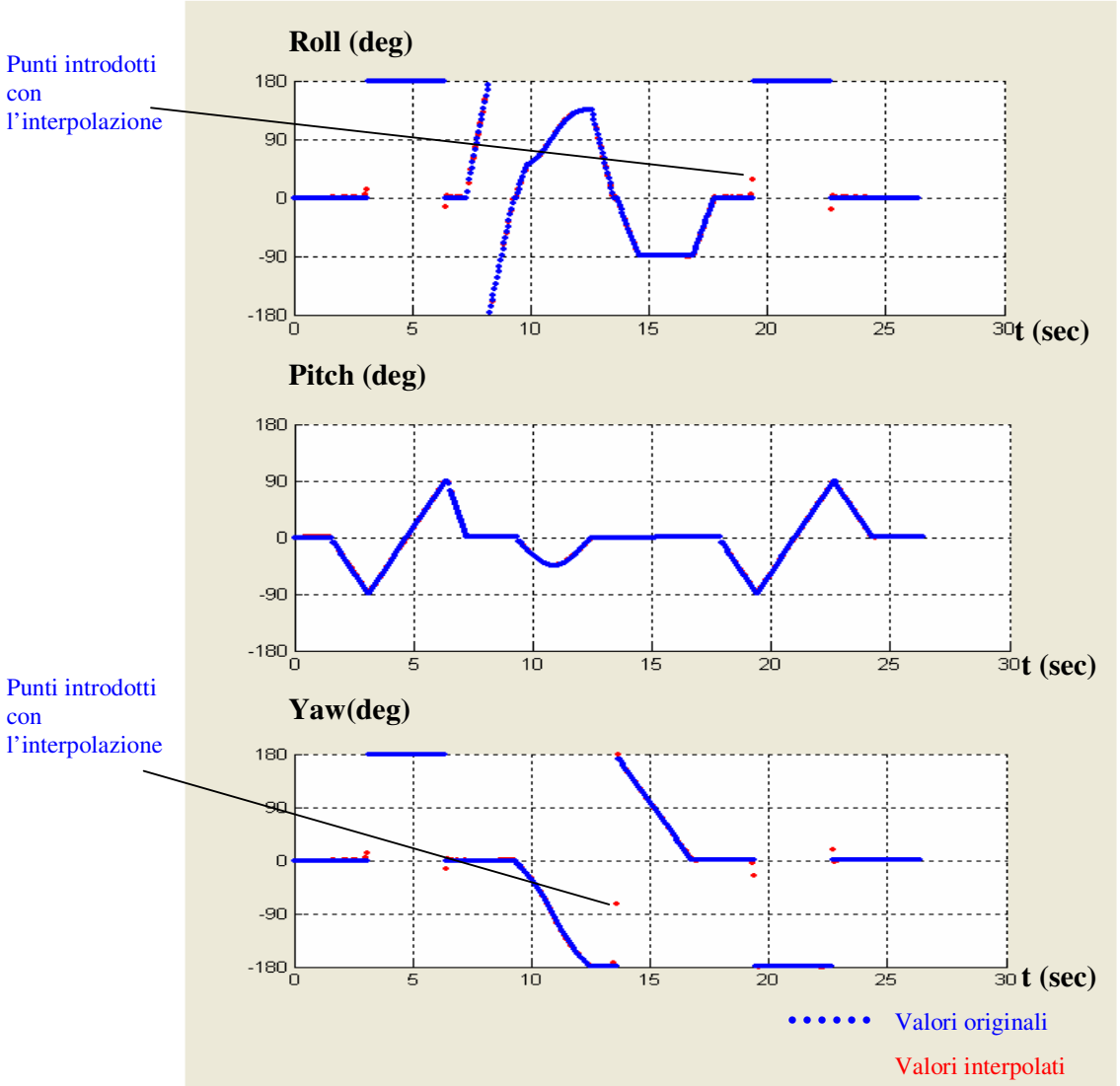


Dai test effettuati la procedura di ridefinizione, sia per errori numerici, sia per le inevitabili discontinuità dei valori degli angoli di orientazione della terna assi corpo del velivolo produce, nel caso si utilizzi l'interpolazione tramite spline, risultati non in linea con le attese.

I Grafici riportati di seguito mostrano come nei punti di discontinuità dei valori degli angoli, l'interpolazione produca dei valori intermedi non significativi, con conseguente incongruenza della traiettoria ridefinita con quella originale.

Il problema esposto si è presentato solo per la ridefinizione dei valori degli angoli di orientazione della terna assi corpo dei velivoli, in quanto i valori atti a definire la posizione del velivolo sono per definizione continui *(nei limiti della definizione di traiettoria per punti)*. Nella figura 5.2 si può notare l'incongrenza tra i valori degli angoli di orientazione originali e quelli calcolati ad istanti temporali multipli.

Figura 5.2

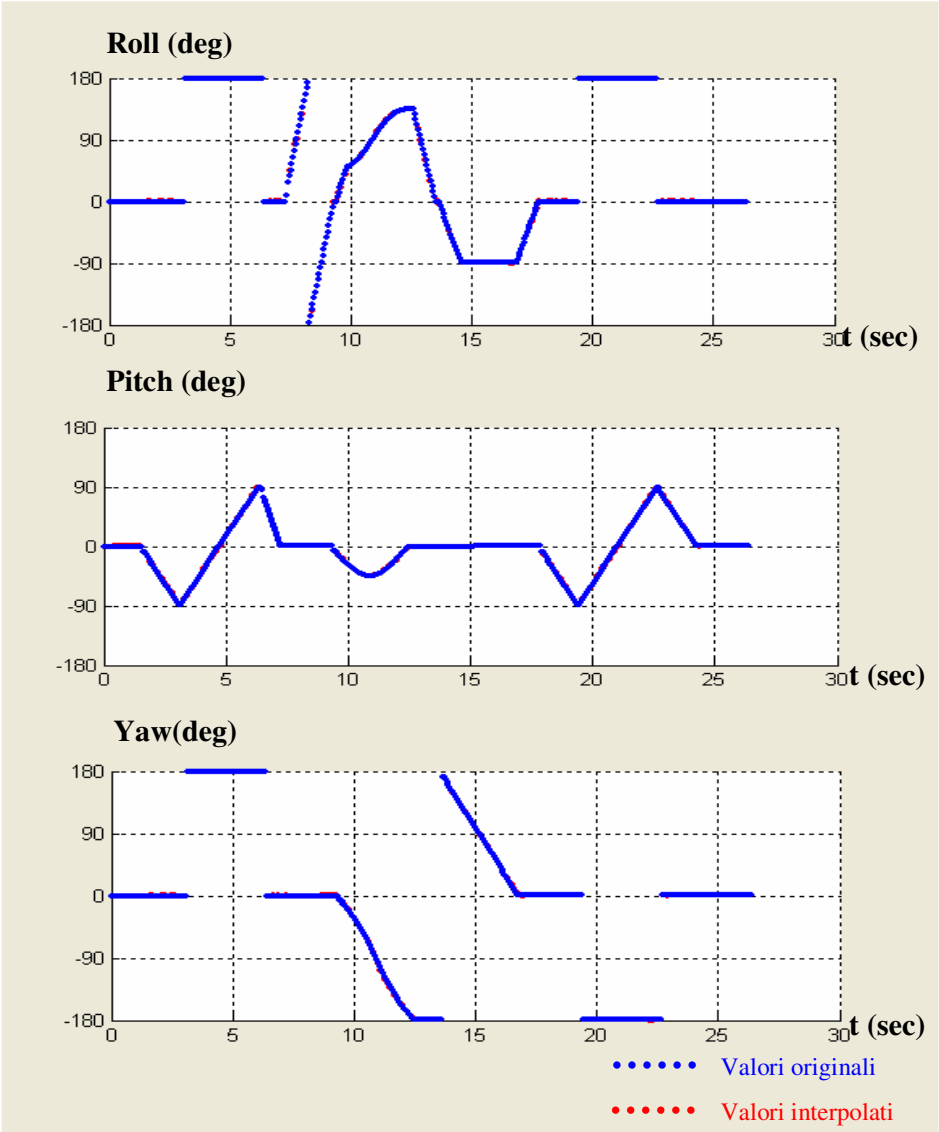


Il problema è stato risolto sostituendo all'interpolazione con spline, l'interpolazione mista *(spline/metodo minimi quadrati).* L'interpolazione mista è implementata valutando lo scarto tra i valori originali e quelli calcolati con l'interpolazione spline, e sostituendo questi ultimi con quelli calcolati con il metodo a minimi quadrati quando lo scarto risulta maggiore di un valore soglia.

```
if (get(handles.sa,'Value')==1)

        Yawtsn=real(interp1(t,Yaw,ts,'nearest'));
        Pitchtsn=real(interp1(t,Pitch,ts,'nearest'));
        Rolltsn=real(interp1(t,Roll,ts,'nearest'));

        Yawts=spline(t,Yaw,ts);
        Pitchts=spline(t,Pitch,ts);
        Rollts=spline(t,Roll,ts);

        deltapointy=Yawts-Yawtsn;
        deltapointp=Pitchts-Pitchtsn;
        deltapointr=Rollts-Rolltsn;
        dimensiondelta=size(deltapointy);

        for indexdelta=1:dimensiondelta(2)
          if abs(deltapointy(indexdelta))>1

            Yawts(indexdelta)=Yawtsn(indexdelta);

          end

          if abs(deltapointp(indexdelta))>1

            Pitchts(indexdelta)=Pitchtsn(indexdelta);

          end

          if abs(deltapointr(indexdelta))>1

            Rollts(indexdelta)=Rolltsn(indexdelta);

          end
        end
end
```

In figura 5.3 è possibile notare come siano stati eliminati i valori non significativi prodotti dall'interpolazione spline e l'accuratezza dei valori interpolati.

Figura 5.3



Durante la fase di test della procedura di ridefinizione delle traiettorie è stato osservato che, per alcuni valori dello *step time* si possono verificare delle piccole discordanze tra valori originali e valori interpolati, probabilmente per il valore di default di soglia utilizzato nell'interpolazione mista.

Un problema simile a quello rilevato per la ridefinizione dei valori degli angoli di orientazione è stato notato nella interpolazione dei parametri di

scia per i quali, nei punti di attacco o di fine scia, i colori assumono valori intermedi.

```
Sciaonofft=pchip(t,Scia);
Sciaonoffts=ppval(Sciaonofft,ts);
Sciaonoffts=floor(Sciaonoffts);

Rt=pchip(t,R);
Rts=ppval(Rt,ts);
Gt=pchip(t,G);
Gts=ppval(Gt,ts);
Bt=pchip(t,B);
Bts=ppval(Bt,ts);
```

L'array di booleani che indica la presenza/assenza della scia in caso assuma valori intermedi nell'intervallo [0,1] si considerano 0

L'interpolazione del colore è eseguita con il metodo pchip

5.3 Verifica della sincronizzazione

La durata delle manovre dei singoli velivoli è imposta dall'utente durante la costruzione della sequenza, definendo i parametri degli elementi base nel caso del *line block* e del *circle block* e selezionando la traiettoria da inseguire nel caso del *follow block (sincronizzazione implicita).*

La verifica della sincronizzazione permette di :

- Valutare la posizione della formazione ad istanti multipli dello step time
- Valutare l'efficienza della procedura di ridefinizione delle traiettorie

Il modulo che si occupa della verifica della sincronizzazione si presenta molto simile a quello per la verifica del comportamento dinamico dei velivoli con le seguenti differenze:

- Viene indicato il tempo di simulazione
- Si può seguire uno dei velivoli della formazione
- Si può verificare la sincronizzazione dei velivoli selezionati

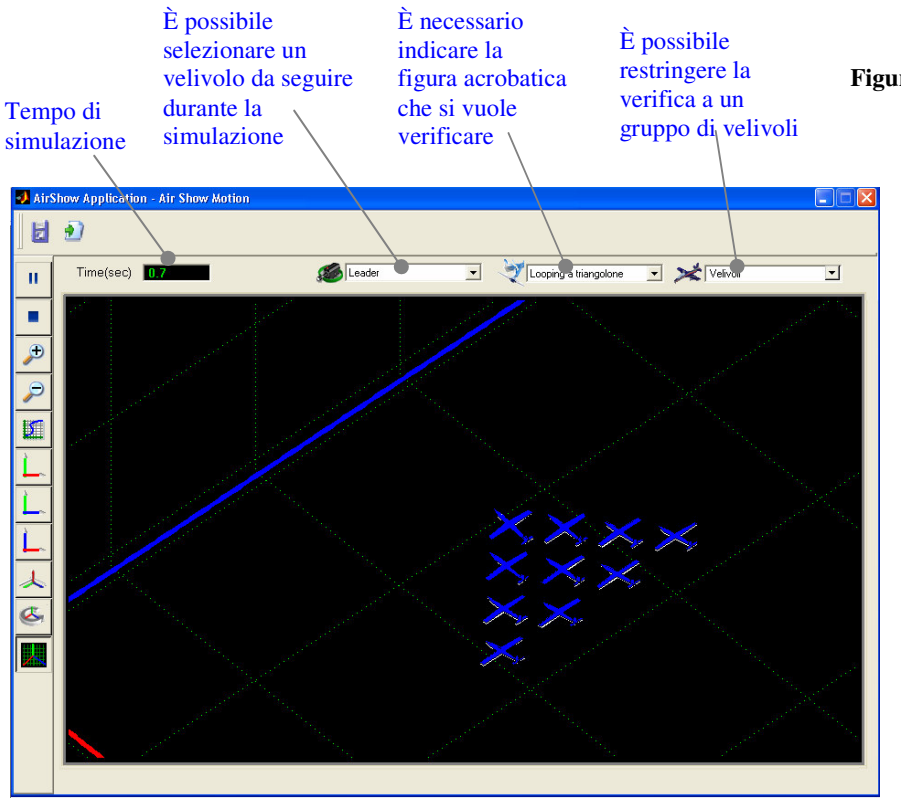
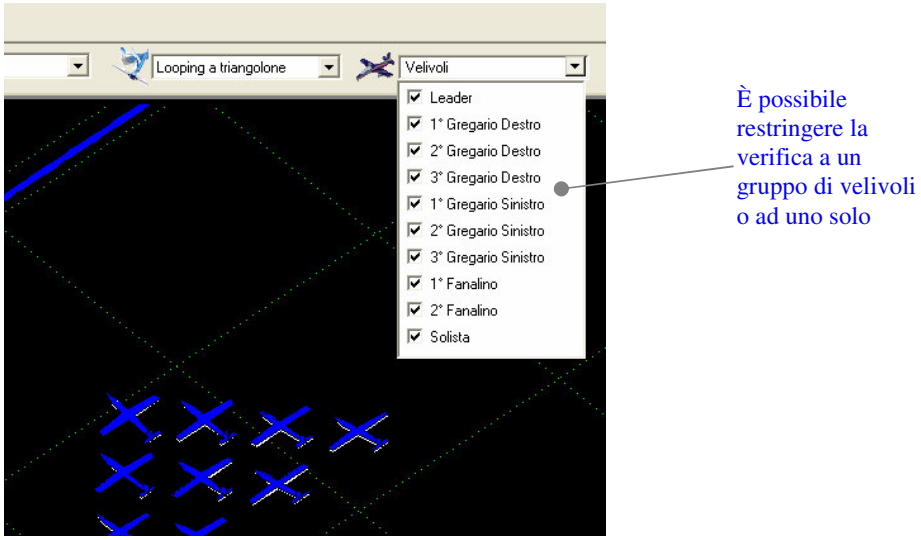


Figura 5.4

Figura 5.5



Al modulo di verifica della sincronizzazione viene passato come parametro la matrice MATRICEDWAF *(fig 5.6).*

Figura 5.6

	Velivolo 1	Velivolo 2	……	Velivolo n-1	Velivolo n
Figura 1					
…	⋮	⋮	⋮	⋮	⋮
Figura n					

Per ogni velivolo e per ogni figura acrobatica sono definiti [*t,x,y,z,roll,pitch,yaw,R,G,B,on/off*]. Il numero di righe della matrice MATRICEDWAF è dato dalla lunghezza del vettore dei tempi per cui è influenzata dallo *step time*. Il numero di colonne della matrice invece è imposto dal numero di velivoli.

5.4 Rendering 3D con Dyna Worlds Professional

Dyna Worlds Professional è un' applicazione di visualizzazione 3D alla quale viene passato come input un file contenente le informazioni per il rendering. Il file di input per Dyna Worlds è costruito da una funzione matlab *creaxdf2.m,* i cui parametri di ingresso sono, il nome del file *.xdf* e la matrice MATRICEDWF,che rispetto alla matrice MATRICEDWAF ha un solo vettore dei tempi, di riferimento per tutti i velivoli.

Dyna Worlds Professional permette di associare ad ogni singola traiettoria un oggetto 3D, la cui posizione e orientazione nel tempo sono imposte dalla traiettoria definita.

Nella figura 5.8 è possibile notare i parametri che è necessario definire per collegare correttamente il canale di input per il rendering.

Figura 5.7

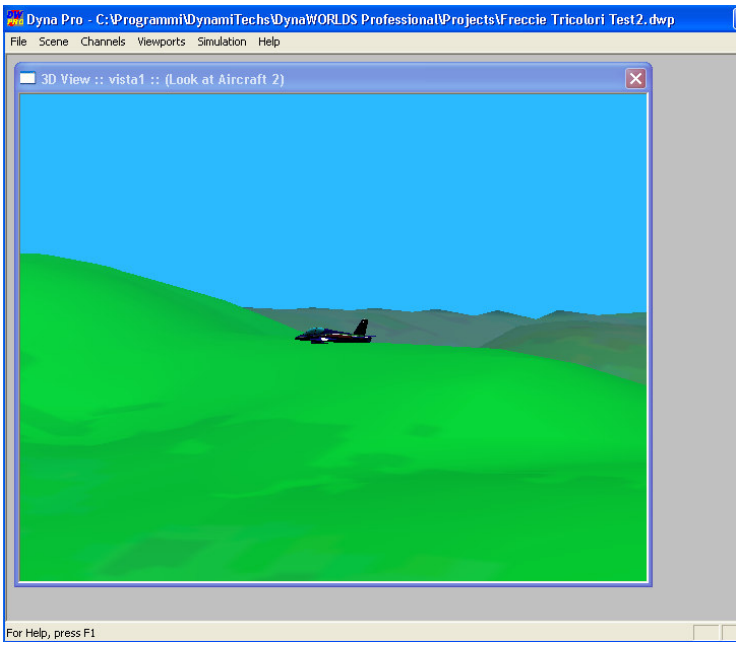
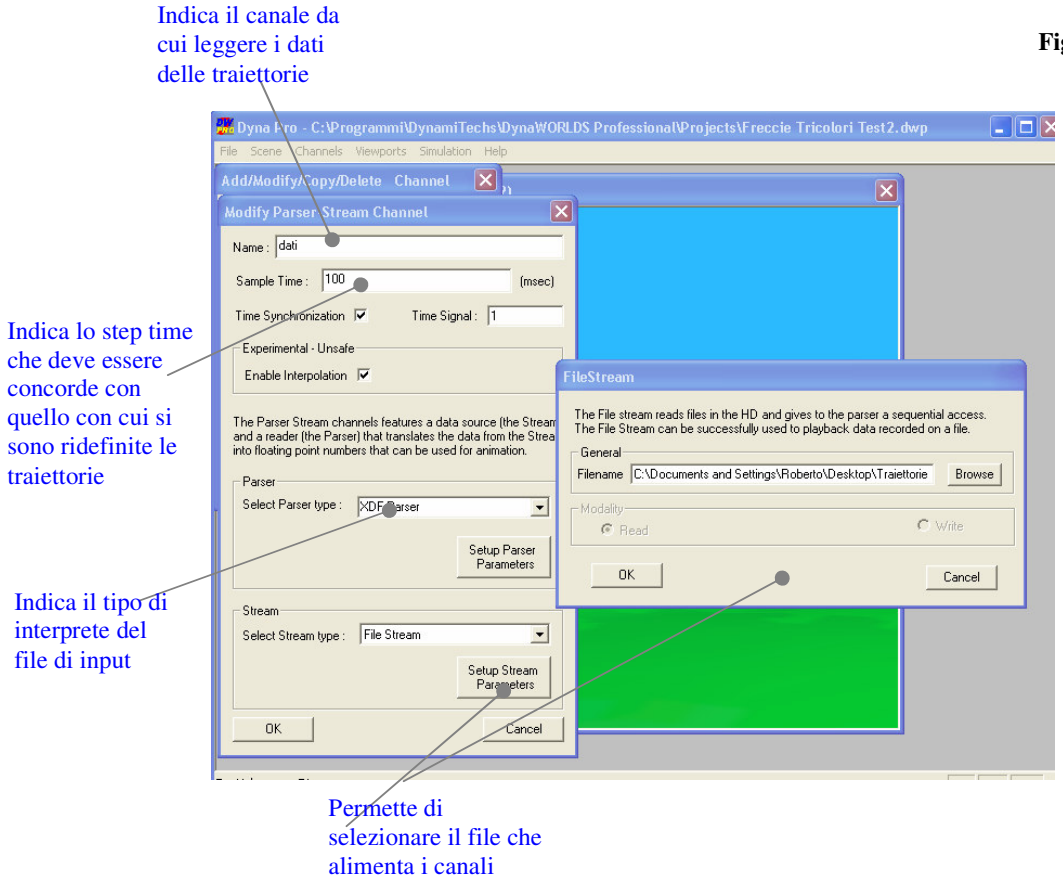


Figura 5.8



Di seguito riporto il codice della funzione matlab che costruisce il file .xdf che alimenta il canale di Dyna World Professional

```
function creaxdf2(nomefile,MATRICEDWF)

% nomefile='freccie_test4.xdf';

MATRICEDWF;

MATDATA=10; %[x y z YAW PITCH ROLL r g b On/Off_scia]
nAircraft=(size(MATRICEDWF,2)-1)/MATDATA;

%get number of samples from time vector of first target
nSamples=size(MATRICEDWF,1);
timeVector=MATRICEDWF(:,1)
pause
if length(nomefile)>3
    if (strcmp(nomefile(length(nomefile)-3:length(nomefile)),'.xdf')==0)
        nomefile=[nomefile,'.xdf'];
    end;
else
    nomefile=[nomefile,'.xdf'];
end;

%SCRITTURA
[fid]=fopen(nomefile,'w');

fwrite(fid,1,'int16'); %num. tipo mess.
fwrite(fid,'messdati  ','char'); %tipo mess. 1

NTIPI=1;
fwrite(fid,2+14*NTIPI,'int16'); %lung. in byte della descr.

%descrizione mess. 1
fwrite(fid,NTIPI,'int16'); %num. tipi di ogg.
fwrite(fid,NTIPI,'int16'); %num. tipi di ogg.

fwrite(fid,to10ch('AIRCRAFT'),'char'); %tipo ogg. 1: AIRCAFT
fwrite(fid,10,'int16'); %num. dati ogg. 1: [x y z ROLL PITCH YAW r g b On/Off_scia]
fwrite(fid,nAircraft,'int16'); %num. max ogg. tipo 1: 1
dimMessAircraft=30+2+10*4; % 3 * 10char strings + validity + 10 single precision numbers

for i=1:nSamples
    fwrite(fid,timeVector(i),'float32'); %tempo

    fwrite(fid,1,'int16');%numero mess
    fwrite(fid,'messdati  ','char');%tipo mess (mess di tipo dati)

    MESSDIM=dimMessAircraft*nAircraft;
    NUMOGG=nAircraft;

    fwrite(fid,MESSDIM,'int16');%lungh mess
    fwrite(fid,NUMOGG,'int16');%num ogg

    for na=nAircraft:-1:1
        fwrite(fid,to10ch('AIRCRAFT'),'char'); %tipo ogg. 1
        fwrite(fid,to10ch('Aereo'),'char'); %nome
        fwrite(fid,to10ch(['A' num2str(nAircraft-na)]),'char');     %ID
        fwrite(fid,1,'int16');%flag di presenza
        fwrite(fid,MATRICEDWF(i,1+(na-1)*MATDATA+1),'float32');%x
        fwrite(fid,MATRICEDWF(i,1+(na-1)*MATDATA+2),'float32');%y
        fwrite(fid,MATRICEDWF(i,1+(na-1)*MATDATA+3),'float32');%z
        fwrite(fid,MATRICEDWF(i,1+(na-1)*MATDATA+6),'float32');%ROLL
        fwrite(fid,MATRICEDWF(i,1+(na-1)*MATDATA+5),'float32');%PITCH
        fwrite(fid,MATRICEDWF(i,1+(na-1)*MATDATA+4),'float32');%YAW
        fwrite(fid,MATRICEDWF(i,1+(na-1)*MATDATA+7)*255.0,'float32');%R
        fwrite(fid,MATRICEDWF(i,1+(na-1)*MATDATA+8)*255.0,'float32');%G
        fwrite(fid,MATRICEDWF(i,1+(na-1)*MATDATA+9)*255.0,'float32');%B
        fwrite(fid,MATRICEDWF(i,1+(na-1)*MATDATA+10),'float32');%B
    end
end;
fclose(fid);
```