Sommario

La presente tesi si inserisce all'interno di un' attività di ricerca svolta presso il Dipartimento di Ingegneria Aerospaziale (DIA) e relativa allo studio del sistema Dati Aria di un moderno velivolo con *Flight Control System* (FCS) di tipo *Fly-by-Wire*. Tale sistema, mediante opportuni algoritmi di calcolo, permette la determinazione dei parametri di volo (quota, velocità, angoli di incidenza e derapata) a partire dalla misurazione di grandezze locali esterne affidata ad apposite sonde di pressione installate sulla fusoliera.

Presso il DIA sono stati sviluppati opportuni algoritmi per la stima dei parametri di volo (procedura di elaborazione*)* e modelli capaci di generare i segnali forniti dalle sonde (procedura di simulazione)

La presente tesi ha ripreso ed aggiornato il modello relativo alla procedura di simulazione, realizzato in ambiente *Matlab-Simulink®*, del sistema Dati Aria, concepito e sviluppato presso il DIA. La procedura di simulazione si basa su un *database* proveniente da prove in galleria del vento condotte su un modello in scala di un velivolo di elevate prestazioni. Dato che tale *database* non ricopre tutto l'inviluppo di volo, in una prima fase del lavoro si è proceduto ad un ampliamento di quest'ultimo. Tale ampliamento è stato realizzato sviluppando modelli basati sulla tecnica di approssimazione ai Minimi Quadrati.

La successiva fase di lavoro è stata dedicata ad individuare le possibili *failure* che possono provocare un malfunzionamento delle sonde di pressione e allo sviluppo di modelli di simulazione di tali *failure*.

Per la gestione delle *failure* e la visualizzazione delle uscite di simulazione è stato realizzato un pannello di controllo in ambiente *LabVIEW®*. Infine il modello di simulazione completo di pannello di gestione delle *failure* è stato sottoposto a prove di simulazione in tempo reale in ambiente *Matlab-Simulink-xPC Target ®*.

Indice

Introduzione

Il presente lavoro si inserisce all'interno delle attività del Dipartimento di Ingegneria Aerospaziale dell'Università degli Studi di Pisa (DIA), nell'ambito di un contratto stipulato con due note due aziende aeronautiche italiane e riguardante il progetto di ricerca denominato *"Sviluppo di un sistema di controllo FBW (Fly-By-Wire) dei comandi primari di volo con attuazione idraulica",* per un velivolo di elevate prestazioni. Tale progetto prevede lo sviluppo di uno studio che contribuisca alla definizione, allo sviluppo ed alla validazione di un sistema di controllo dei comandi primari di volo con segnale di trasmissione elettrico ed attuazione idraulica.

L'oggetto di studio della presente tesi è il sistema Dati Aria di un moderno velivolo di tipo F*ly-by-Wire.* Il ruolo svolto dal sistema Dati Aria nei *Flight Control System* (FCS) di tipo F*ly-by-Wire* (FBW) è quello di determinare con opportuna ridondanza sia i parametri di volo (quota e velocità) sia gli angoli di incidenza (α) e derapata (β) a partire dalla conoscenza di grandezze di flusso locale misurate da apposite sonde installate generalmente sulle fusoliere dei velivoli. Gli angoli di incidenza (α) e derapata (β) vengono determinati sulla base di angoli di di flusso locale misurati dalle sonde, mentre quota è velocità di volo vengono determinati grazie alla misura di pressioni locali statiche e dinamiche effettuate dalle sonde. La derivazione dei parametri di volo, a partire dalle misurazioni dalle sonde, viene di solito affidata ad un algoritmo eseguito dai *Flight Control Computers* (FCCs) del FCS o da unità di calcolo dedicate. A tal proposito, presso il DIA sono stati sviluppati opportuni algoritmi per la stima dei parametri di volo (procedura di elaborazione*)* [1], [2] e modelli capaci di generare i segnali forniti dalle sonde (procedura di simulazione) [3]. La procedura di simulazione si è resa necessaria per la progettazione e la convalidazione degli algoritmi di elaborazione. Inoltre essa sarà utilizzata nella fase di simulazione su

Iron-Bird durante la quale verrà verificata la funzionalità dell'intero FCS del velivolo. Lo sviluppo di entrambe le procedure è stato supportato da un *database* proveniente da prove in galleria del vento condotte su un modello in scala del velivolo preso in esame.

Occorre precisare che è previsto che vadano in volo più versioni della procedura di elaborazione e che le prime non includeranno la parte relativa alla ricostruzione degli angoli di incidenza e di derapata del velivolo a partire dagli angoli del flusso locale misurati dalle sonde. Nei primi voli infatti, il velivolo preso in esame utilizzerà un *Noseboom* per la valutazione degli angoli di incidenza e derapata. La procedura di simulazione, dunque, deve simulare oltre che le uscite generate dalle sonde multi-fuzione anche l'angolo di incidenza e di derapata misurato dal *Noseboom*.

Obbiettivo della presente tesi è lo sviluppo di modelli ed algoritmi per la simulazione del funzionamento del sistema Dati Aria e del *Noseboom*, nonché lo sviluppo di metodologie per la simulazione delle possibili avarie relative al sistema Dati Aria ed al *Noseboom.*

La prima parte del lavoro ha riguardato l'ampliamento a tutto l'inviluppo di volo α-β di interesse delle *look-up-table* di interpolazione presenti nella precedente versione del modello di simulazione sviluppato presso il DIA [4]. Il *database* aerodinamico, proveniente da prove in galleria del vento realizzate da una delle due azienze con cui il DIA collabora, fornisce le grandezze locali del flusso agenti attorno al velivolo in funzione del numero di Mach al variare degli angololi di incidenza e di derapata, ma non ricopre alcune zone del dominio di interesse. Per realizzare la simulazione dei Dati Aria in modo soddisfacente e su tutto l'inviluppo di volo α-β è stato quindi necessario interpolare i dati disponibili. Sono state utilizzate delle metologie basate sulla tecnica di Minimi Quadrati utilizzando funzioni approssimanti di tipo polinomiale. I polinomi ricavati hanno permesso di predire le grandezze di simulazione nelle zone dell'inviluppo in cui non sono disponibili dati di galleria del vento. Per poter ben

vincolare i polinomi approssimanti in modo da evitare la loro divergenza in particolari zone del dominio di approssimazione sono stati introdotti ulteriori dati. Per alcuni valori del numero di Mach, infatti, i dati disponibili non sono sufficienti e sono stati ampliati utilizzando le informazioni note ad altri numeri di Mach opportunamente scalate utilizzando leggi di tipo *Prandtl-Meyer*.

La successiva fase di lavoro è stata rivolta allo studio delle possibili *failure* che possono provocare un malfunzionamento delle sonde di pressione e allo sviluppo di modelli di simulazione di tali *failure* in modo da consentire la valutazione delle possibili perdite di prestazione del sistema.

Inoltre è stato realizzato un pannello di controllo in ambiente *LabVIEW®* per attivare le *failure* nel modello e visualizzare le uscite di quest'ultimo. Il modello completo di pannello di attivazione delle *failure* è stato sottoposto a prove di simulazione in tempo reale in ambiente *Matlab-Simulink-xPC Target ®*.

La procedura di simulazione è stata sottoposta anche ad altri tipi di test, effettuati non in tempo reale, al fine di esaminare la correttezza dei risultati ottenuti, attraverso un confronto con i valori sperimentali a disposizione, e di avere una documentazione e una verifica dei risultati nelle più svariate condizioni di impiego.

1. Descrizione del Sistema Dati Aria

Il sistema Dati Aria (*Air Data System*, ADS) di un velivolo FBW ha come scopo la determinazione con opportuna ridondanza degli angoli di incidenza (α) e derapata (β), vedi fig. 1.1, della pressione statica ambiente (Psa) e della pressione totale (Pt) a partire dalla misurazione di grandezze relative al flusso attorno alla fusoliera (grandezze di flusso locale). La rilevazione fisica di tali grandezze è affidata ad apposite sonde di pressione, tipicamente situate nella parte prodiera della fusoliera che forniscono misure di pressione e direzione del flusso aerodinamico nei punti di installazione delle sonde stesse. La successiva derivazione dei parametri di volo, a partire dalle misurazioni di tali sonde, viene di solito affidata ad un algoritmo eseguito dai *Flight Control Computers* del *Flight Control System* o da unità di calcolo dedicate.

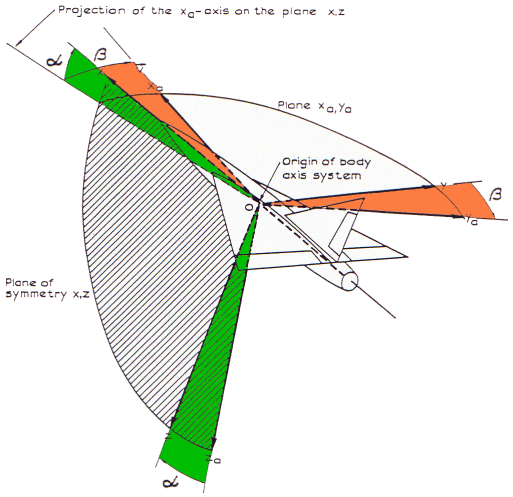

Fig. 1.1

Il sistema Dati Aria preso in esame, è riferito ad un tipico velivolo di elevate prestazioni. In genere questi sistemi, dotati di quattro sonde multi-funzione, devono essere in grado di sopperire alla presenza di una *failure* senza sensibili perdite di prestazioni e di operare comunque in sicurezza quando il numero di avarie è maggiore di uno (*fail operative – fail safe*).

Le fig. 1.2 e 1.3 mostrano una classica installazione di tali sonde in fusoliera. Le sonde identificate come *Lower Right* (LR) e *Lower Left* (LL) sono installate nella sezione A-A mentre nella sezione B-B sono posizionate le sonde *Upper Right* (UR) e *Upper Left* (UL).

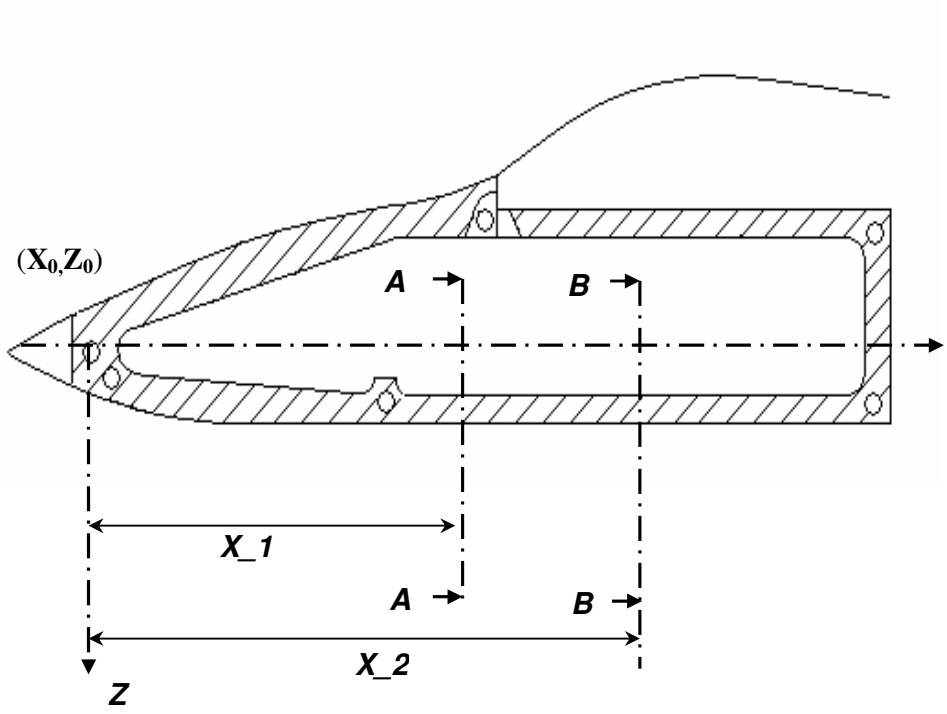

Fig. 1.2: Sezione longitudinale della fusoliera

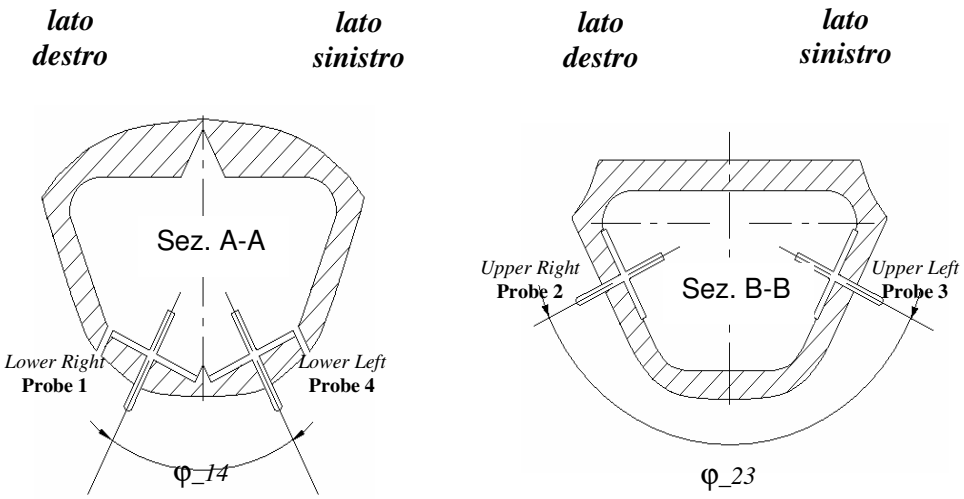

Fig. 1.3: Posizione angolare delle quattro sonde

Le sonde prese in esame in questo lavoro sono di forma conica e dotate di cinque prese (*slots)* di pressione, come mostrato in fig. 1.4.

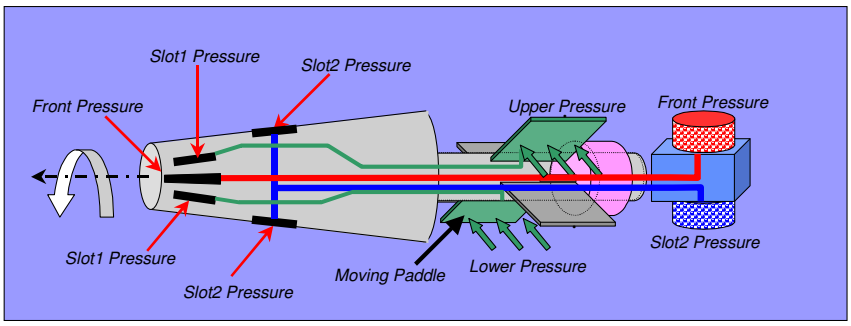

Fig. 1.4: Vista interna della sonda

Queste sonde si allineano alla direzione locale del flusso dal quale sono investite ruotando attorno al loro asse di simmetria conico (*Probe Axis*).

La dinamica di allineamento viene indotta da una differenza di pressioni rilevata dai due *slot1* ed evolve in modo tale da ripristinare una condizione di equilibrio tra le due pressioni (condizione di allineamento). La fig. 1.5 mostra una vista in pianta della sonda conica allineata alla direzione locale del flusso.

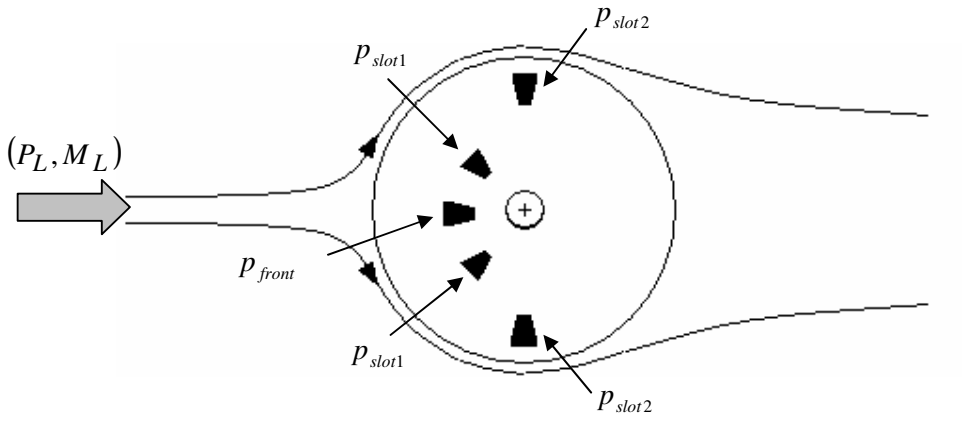

Fig. 1.5: Vista in pianta della sonda multi-funzione

La presa di pressione che risulta allineata alla direzione locale del flusso viene chiamata *frontal slot* e misura una pressione locale chiamata *frontal pressure* (P_{front}) che è paragonabile alla pressione totale rilevata da una sonda anemometrica classica (*like total pressure*). Le pressioni misurate dalle prese *slot2* vengono fisicamente mediate tramite un condotto di comunicazione in quella che viene chiamata *slot2 pressure* (P_{slot2}), che è paragonabile alla pressione statica rilevata da una sonda classica (*like static pressure*).

In realtà, dato il profilo delle pressioni attorno alla sonda conica, sia la *frontal pressure* sia la *slot2 pressure* sono sensibilmente diverse dalla *total pressure* e dalla *static pressure* che verrebbero misurate da una sonda di tipo tradizionale installata al posto della sonda multi-funzione.

Nella maggior parte dei casi le quattro sonde sono montate nella parte anteriore della fusoliera, che perturba il campo aerodinamico, per cui ogni sonda è immersa

in un flusso locale le cui caratteristiche P_{Li} ed M_{Li} (fig. 1.6) possono essere determinate in una campagna di prove in galleria del vento su un modello in scala del velivolo.

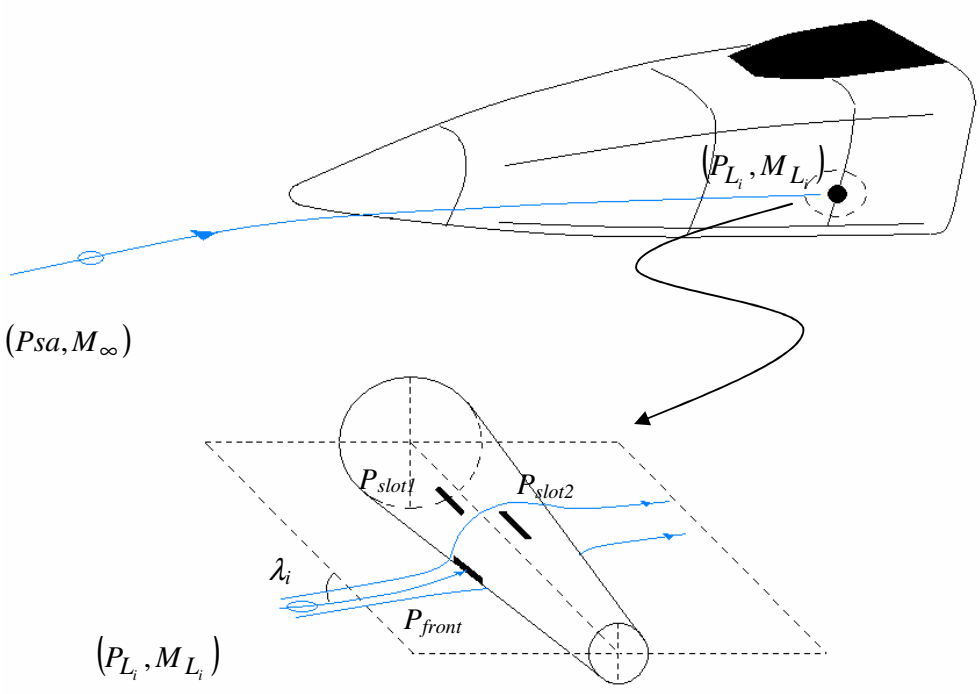

Fig. 1.6: Condizioni di flusso asintotiche e locali

Oltre alle misure di pressione locale, la sonda multi-funzione restituisce una misura dell'angolo locale del flusso dal quale è investita (di seguito indicato con la lettera λ), rispetto ad una direzione di zero opportunamente assegnata: tale misura viene effettuata da un trasduttore di posizione angolare noto in letteratura come *Rotary Variable Differential Transducer* (R.V.D.T.).

Le sonde forniscono anche per ogni misura effettuata un segnale di validity. I segnali di validity vengono generati da un algoritmo *Built-in-Test* (BIT) delle sonde che effettua un monitor di tipo elettrico per stabilire se le misure associate possono essere usate.

I moderni FBW *Full Authority* richiedono la quadruplice ridondanza degli elementi essenziali del FCS e quindi sono dotati di quattro *Flight Control Computers* (FCCs). Le misure delle sonde possono essere fornite ai quattro FCCs in più modi.

La fig. 1.7 illustra una possibile architettura dell'ADS in cui ogni sonda fornisce due pressioni locali ed un angolo di flusso ad un singolo *Flight Control Computer* (FCC); i quattro FCCs comunicano tra loro mediante un sistema di comunicazione e scambio di dati (*Cross Channel Data Link*) in modo tale che ogni calcolatore acquisisca i segnali relativi alle pressioni ed agli angoli locali di tutte e quattro le sonde.

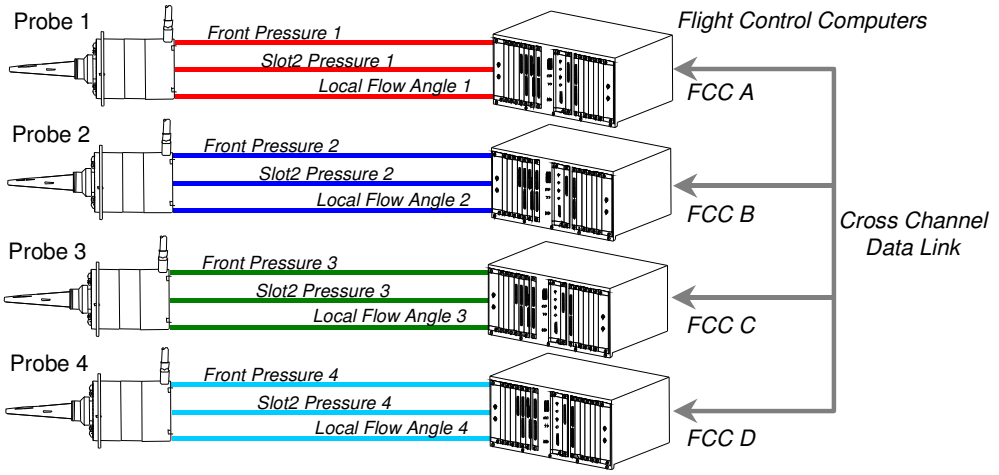

Fig. 1.7: Possibile architettura dell'Air Data System

2. Procedura di Simulazione

Come visto nel capitolo precedente relativamente alla descrizione del sistema Dati Aria, ogni sonda fornisce in uscita tre dati, ovvero l'angolo di flusso locale e due misure di pressione. Sulla base di questi dati, mediante opportuni algoritmi di elaborazione dei Dati Aria (procedura di elaborazione), vengono stimati i dati di volo richiesti. Parallelamente alla procedura di elaborazione è necessario sviluppare dei modelli di simulazione dei Dati Aria (procedura di simulazione) capaci di generare i segnali forniti dalle sonde e costituenti gli *input* dell'algoritmo di elaborazione (fig. 2.1) nelle fasi di sviluppo e convalidazione del sistema.

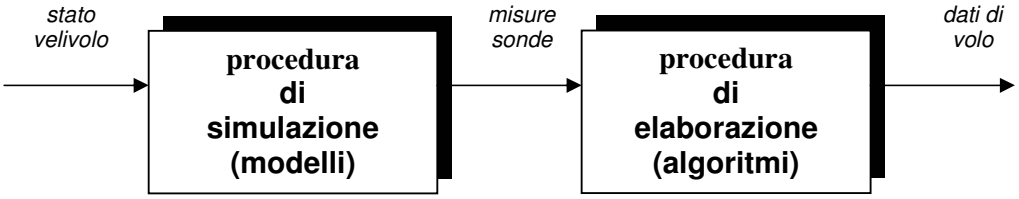

Fig. 2.1: Interazione tra la procedura di simulazione e quella di elaborazione

Si ricorda, come già affermato, che la versione della procedura di elaborazione in corso di sviluppo non include la parte relativa alla ricostruzione degli angoli di incidenza e di derapata del velivolo a partire dagli angoli del flusso locale misurati dalle sonde. E' infatti previsto che in una prima fase, la procedura di elaborazione utilizzi angoli di incidenza e di derapata forniti da un *Noseboom.*

La procedura di simulazione, dunque, simula oltre che le uscite generate dalle sonde multi-fuzione anche l'angolo di incidenza e di derapata misurato dal *Noseboom*.

La procedura di simulazione è necessaria sia in fase di progetto degli algoritmi di elaborazione sia nella fase di simulazione su *Iron-Bird* durante la quale viene verificata la funzionalità dell'intero *Flight Control System*.

Viene di seguito descritta la procedura di simulazione sviluppata, che riguarda quei processi che permettono di trasformare gli *input*, quali quota, numero di Mach, angolo di incidenza e derapata, configurazione del velivolo, velocità lineare e angolare, negli *output*, ovvero i segnali generati dalle sonde e dal *Noseboom*.

2.1 Descrizione della procedura di Simulazione dei sensori

L'algoritmo di simulazione è costituito da un insieme di processi e modelli che permettono di ottenere le grandezze misurate dalle sonde del sistema Dati Aria $\lambda_i, P_{front_i}, P_{slot2_i}$ (dove il pedice i (i=1,…,4) rappresenta l'indice delle sonde), noti il moto del velivolo e le caratteristiche dell'ambiente operativo.

Nell'aviazione moderna vengono usati diversi tipi di sonde per la misura dei Dati Aria, ma in generale tutte le diverse tipologie esistenti effettuano misure dipendenti dal campo aerodinamico instauratosi attorno alle sonde stesse. La forma geometrica delle sonde e le condizioni di flusso in prossimità dei loro punti di installazione sono i principali elementi che determinano tale campo aerodinamico. Le condizioni di flusso locale nei punti di installazione delle sonde sono esprimibili attraverso i parametri λ_i, P_{L_i} e M_{L_i} che sono funzioni sia delle perturbazioni del flusso introdotte dalla fusoliera sia degli effetti di manovra. D'altra parte le perturbazioni del flusso sono strettamente legate alla geometria della fusoliera ed alle condizioni asintotiche a monte del velivolo, esprimibili in termini del numero di Mach di volo M_∞, della pressione statica ambiente Psa e degli angoli d'incidenza α e derapata β.

Alla luce di queste considerazioni, la simulazione dei segnali generati dal sistema Dati Aria deve basarsi sulla conoscenza delle seguenti funzioni, nella loro più generale accezione:

$$
\begin{aligned}
a)\quad & \begin{cases} Cp_{Li}\left(\alpha,\beta,M_\infty,\vec{\Omega}_B,t,Config\right) \\ \lambda_i\left(\alpha,\beta,M_\infty,\vec{\Omega}_B,t,Config\right) \end{cases} \\
b)\quad & \begin{cases} P_{Li} = f_P\left(Cp_{Li}\right) \\ M_{Li}\left(\alpha,\beta,M_\infty,\vec{\Omega}_B,t,Config\right) \end{cases} \\
c)\quad & \begin{cases} P_{front_i} = f_{front}\left(P_{Li},M_{Li}\right) \\ P_{slot2_i} = f_{slot}\left(P_{Li},M_{Li}\right) \end{cases}
\end{aligned}
\qquad (2.1)
$$

Le prime due funzioni caratterizzano il "*flusso locale*", cioè il flusso in prossimità dei punti di installazione delle sonde, in funzione delle condizioni asintotiche (α, β e M_∞), della velocità angolare del velivolo ($\vec{\Omega}_B$), della configurazione del velivolo (*Config.)* e degli effetti non stazionari (t). Il Cp_{Li} è il coefficiente di pressione locale e da esso, come si vedrà nel seguito, può essere ricavata la pressione nel punto d'installazione della sonda i-esima (2.1b), mentre M_{Li} è il numero di Mach locale. Le ultime due funzioni rappresentano il modello stazionario del funzionamento della sonda isolata e mettono in relazione le pressioni misurate con le condizioni di flusso subito a monte della sonda stessa. Per loro definizione, quindi, queste funzioni possono essere desunte da uno studio "*stand alone*" della sonda, cioè un'analisi della sonda isolata senza prendere in considerazione gli effetti di perturbazione introdotti dalla fusoliera.

La determinazione delle funzioni Cp_{Li}, λ_i e M_{Li} può essere ottenuta mediante prove in galleria aerodinamica, ma al tal proposito va precisato che è difficile ottenere la dipendenza di tali funzioni da $\vec{\Omega}_B$ e dagli effetti non stazionari. Più ragionevole è pensare di realizzare delle prove assimilabili a condizioni di moto rettilineo e stazionario con velocità angolare nulla e tenere conto degli effetti di manovra mediante lo sviluppo di modelli teorici basati su correzioni di tipo cinematico da tarare, successivamente, in fase di prove di volo. La presenza della velocità angolare $\vec{\Omega}_B$ comporta una rotazione del velivolo intorno al baricentro e, quindi, modifica il campo di velocità intorno al velivolo stesso. In particolare, le componenti di velocità angolare Q_B e R_B comportano una differenza tra il numero di Mach e gli angoli d'incidenza e di derapata relativi al baricentro e quelli riferiti ad una generica sezione della fusoliera. Questa differenza esistente tra i valori baricentrici e quelli di sezione, porta a concludere che una possibile via per considerare gli effetti di Q_B e R_B è quella di calcolare, in fase di simulazione, le funzioni Cp_{Li} e λ_{Li}, ottenute in condizioni stazionarie

con velocità angolare nulla, non utilizzando i valori baricentrici α, β, e M_∞, ma quelli relativi alle sezioni di installazione, ovvero α_{si}, β_{si}, e M_{si}. Più difficile è includere l'effetto della velocità di rotazione P_B; quest'ultima comporta una rotazione del velivolo intorno all'asse X_B e, quindi, a differenza di Q_B e R_B, non introduce delle differenze tra i valori baricentrici e quelli di sezione. Nonostante ciò, tale rotazione influenza comunque il flusso in prossimità dei punti di installazione delle sonde. Per considerare l'effetto di P_B, quindi, bisogna sviluppare dei modelli che operino direttamente sui valori λ_i, P_{L_i} e M_{L_i} ottenuti in condizioni di $P_B = 0$. Essendo difficile sviluppare dei modelli capaci di correlare le P_{L_i} con P_B, l'algoritmo sviluppato prevede solo la correzione di λ_i e M_{L_i}.

Per quanto riguarda gli effetti non stazionari, è complicato non solo realizzare prove in galleria del vento capaci di valutare tali effetti, ma anche lo sviluppo di modelli teorici che riescano a descrivere in maniera adeguata l'interazione tra tali non stazionarietà e le condizioni locali di flusso. Per tale motivo è stato deciso di trascurarli in questa prima fase del lavoro salvo, se necessario, reintrodurli in fase di confronto dei risultati con i dati delle prove di volo.

Si osservi, infine, che ai modelli riportati in (2.1) si devono aggiungere quelli relativi alla dinamica di rotazione con la quale la sonda si allinea alla direzione locale del flusso e quelli relativi alla dinamica dei trasduttori (angolari e di pressione). In questo modo si può tenere conto degli effetti dei transitori dinamici sulle misure delle sonde.

In base a queste considerazioni l'algoritmo di simulazione può essere schematizzato con il diagramma a blocchi di fig. 2.2. Da questo schema è possibile osservare il flusso di dati e i vari processi che compongono la procedura di simulazione e, quindi, capire come le grandezze misurate dal sistema Dati Aria ($\lambda_i, P_{front_i}, P_{slot2_i}$) sono correlate con gli *input* quali la quota, la configurazione

del velivolo *Config* in termini di deflessione delle superfici mobili, ed i valori della velocità lineare $\vec{V}_B$ e di quell'angolare $\vec{\Omega}_B$. Nello schema di fig. 2.2, il blocco 1 desume dalla conoscenza della quota *h* le caratteristiche dell'ambiente operativo in termini di pressione statica ambiente *Psa* e velocità del suono a. A tale scopo viene utilizzato il modello dell'Aria Standard per la cui descrizione si rimanda ai paragrafi successivi. Le funzioni Cp_{Li}, λ_i e M_{Li}, ottenute in condizioni stazionarie con velocità angolare nulla sono presenti nei blocchi 3, 4 e 5. Noto il coefficiente di pressione Cp_{Li}, i modelli che correlano quest'ultimo con le condizioni locali P_{L_i} di pressione (2.1b) sono contenuti nel blocco 6. I blocchi che simulano gli effetti di manovra sono il 2 ed il 7; il primo valuta gli effetti delle componenti Q_B e R_B e, quindi, in base a quanto detto precedentemente, calcola gli angoli di incidenza e di derapata ed il numero di Mach relativi alle sezioni di installazione delle sonde noti l'ambiente operativo e i valori della velocità baricentrica $\vec{V}_B$ e delle componenti di velocità angolare suddette; il secondo valuta l'effetto di P_B su M_{L_i} e λ_i (che a valle di tale correzione vengono denominati $M^*_{L_i}$ e $\lambda_i{}^*$) noti i valori λ_i, M_{L_i}, a e P_B. Infine nei blocchi 8 e 9 sono riportati il modello di funzionamento della sonda isolata (2.1c) ed i modelli relativi alla dinamica con la quale la sonda si allinea con la direzione locale del flusso ed alla dinamica dei trasduttori.

Nei paragrafi successivi si procede alla descrizione dei modelli presenti all'interno dei blocchi che costituiscono la procedura di simulazione.

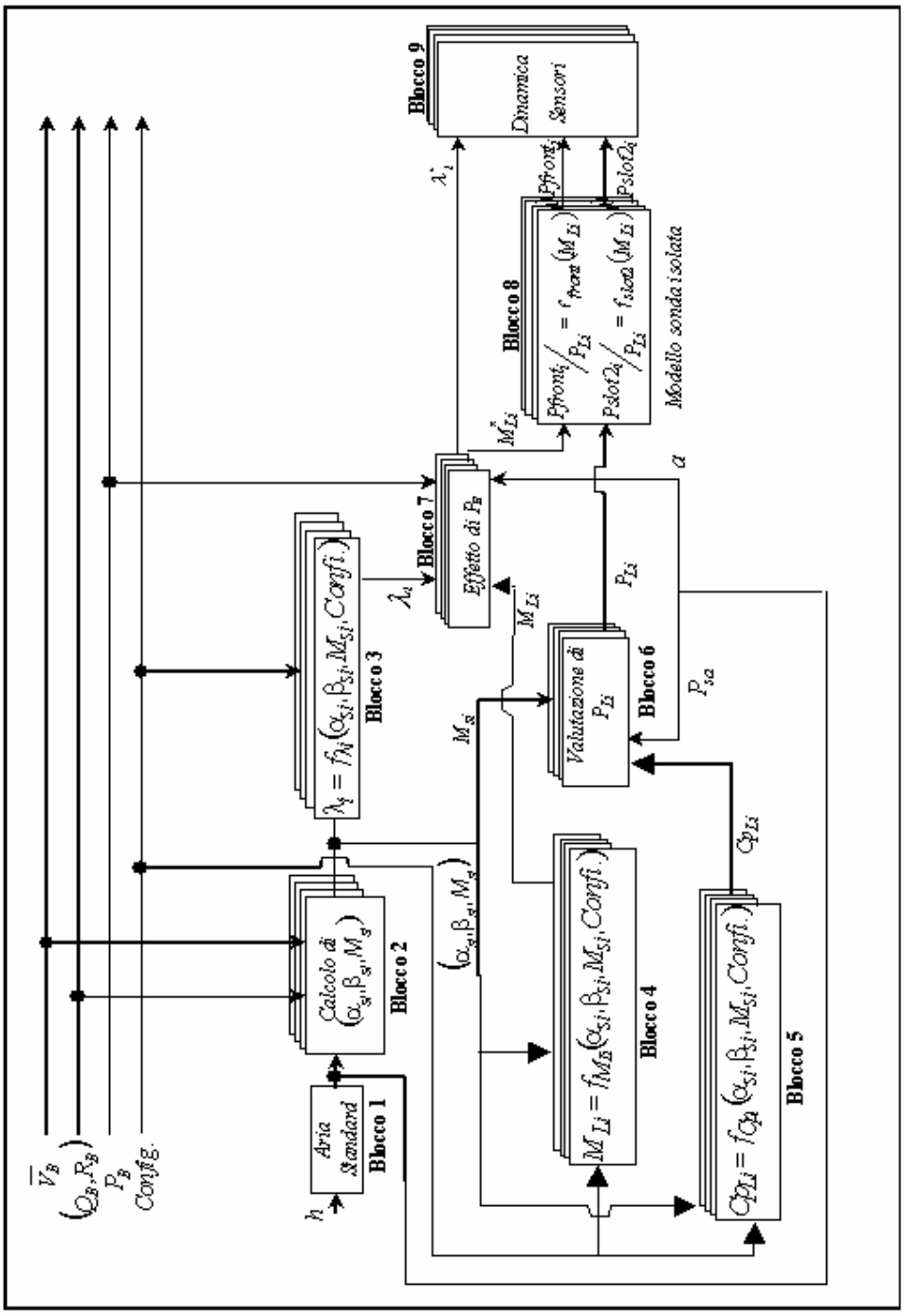

Fig. 2.2 : Algoritmo della procedura di Simulazione

2.1.1 Descrizione del blocco 1 – Aria Standard

Le caratteristiche di volo dei velivoli dipendono anche dalle caratteristiche dell'ambiente operativo ed in particolare dalla temperatura T_a (in genere chiamata "temperatura ambiente"), dalla pressione statica ambiente Psa e dalla densità ρ_a ("densità ambiente"). Queste grandezze variano in generale con la quota in modo diverso da zona a zona della superficie terrestre. Esse, infatti, dipendono anche dalla latitudine e dalle condizioni meteorologiche che interessano la zona considerata.

Nell'algoritmo di simulazione dei Dati Aria, il calcolo della Psa, T_a, e ρ_a viene effettuato considerando la sola dipendenza dalla quota. Il modello usato per esprimere tale dipendenza è definito dalle (2.2) e si riferisce alla "Aria Standard", universalmente accettata in sede internazionale. Assegnando opportuni valori di pressione, temperatura e densità a quota zero, è possibile valutare la loro variazione con la quota mediante le equazioni dei gas perfetti.

$$
\begin{cases}
T_a = T_0 - 0.0065 \cdot h & \text{(K)} \quad \text{h} < 11000\ \text{m} \\
T_a = 216.15 & \text{(K)} \quad 11000\ \text{m} < \text{h} < 20000\ \text{m} \\
P_{sa} = P_0 \cdot \left(1 - \dfrac{0.0065 \cdot h}{T_0}\right)^{\frac{1}{0.0065 \cdot \text{R}/g}} & \left(\text{N}/\text{m}^2\right) \quad \text{h} < 11000\ \text{m} \\
P_{sa} = P(11000) \cdot e^{\frac{11000-h}{\text{R}/g \cdot 216.15}} & \left(\text{N}/\text{m}^2\right) \quad 11000\ \text{m} < \text{h} < 20000\ \text{m} \\
\rho_{sa} = \dfrac{P_{sa}}{R \cdot T_a} & \left(\text{Kg} \cdot s^2/\text{m}^4\right)
\end{cases}
\qquad (2.2)
$$

con

$$
\begin{cases}
T_0 = 288\ \text{(K)},\ P_0 = 101325\ \ (\text{N}/\text{m}^2), \\
P(11000) = 86972\ \ (\text{N}/\text{m}^2),\ \text{R} = 287.25\left(\text{m}^2/\left(\text{sec}^2 \cdot \text{K}\right)\right)
\end{cases}
$$

Nota la temperatura, è possibile calcolare la velocità del suono attraverso la seguente espressione:

$$a = \sqrt{\gamma \cdot R \cdot T_a} \tag{2.3}$$

2.1.2 Descrizione del blocco 2 – Calcolo dei valori di sezione ($\alpha_{si}, \beta_{si}, M_{si}$)

In questo blocco vengono calcolati gli angoli di incidenza α_{si} e di derapata β_{si} ed il numero di Mach M_{si} relativi alle sezioni di installazione delle sonde (sezioni A-A e B-B di fig. 1.3). La presenza di componenti di velocità angolare Q_B e R_B lungo gli assi corpo Y_B e Z_B aggiunge al flusso asintotico un campo di velocità dipendente dalla sezione considerata e, quindi, crea delle differenze tra gli angoli di incidenza e di derapata ed il numero di Mach relativi al baricentro e quelli misurati in una generica sezione della fusoliera. Tali differenze non variano all'interno della sezione, cioè sono le stesse per tutte le sonde installate in quella sezione. Un'eventuale componente di velocità angolare P_B lungo l'asse X_B, invece, introduce un campo di velocità la cui intensità dipende dalla distanza dall'asse X_B. Ciò comporta che le variazioni del campo di velocità dovute a P_B siano diverse per le varie sonde, indipendentemente dalla sezione in cui sono installate.

Gli effetti della velocità angolare P_B saranno discussi nel § 2.1.7. Di seguito, invece, s'illustra il modello matematico derivante dalla schematizzazione riportata in fig. 2.2, che esprime le dipendenze di α_{si}, β_{si} e M_{si} dalle componenti della velocità baricentrica $\vec{V}_B$ e dalle componenti di velocità angolare Q_B e R_B. Tali componenti di velocità angolare introducono un campo di velocità nelle sezioni d'installazione delle sonde dato da:

$$\begin{aligned} u_{si} &= u_B \\ v_{si} &= v_B + X_{Bi} \cdot R_B \\ w_{si} &= w_B - X_{Bi} \cdot Q_B \end{aligned} \tag{2.4}$$

ed essendo:

$$
\begin{aligned}
V_{si}^2 &= \sqrt{u_{si}^2 + v_{si}^2 + w_{si}^2} \\
u_{si} &= V_{si} \cdot \cos(\beta_{si}) \cdot \cos(\alpha_{si}) \\
v_{si} &= V_{si} \cdot sen(\beta_{si}) \\
w_{si} &= V_{si} \cdot \cos(\beta_{si}) \cdot sen(\alpha_{si})
\end{aligned}
\tag{2.5}
$$

si ottiene che

$$
\begin{aligned}
\alpha_{si} &= \arctan\left(\frac{w_{si}}{u_{si}}\right) \\
\beta_{si} &= \arctan\left(\frac{v_{si}}{\sqrt{u_{si}^2 + w_{si}^2}}\right) \\
M_{si} &= \frac{\sqrt{u_{si}^2 + v_{si}^2 + w_{si}^2}}{a}
\end{aligned}
\tag{2.6}
$$

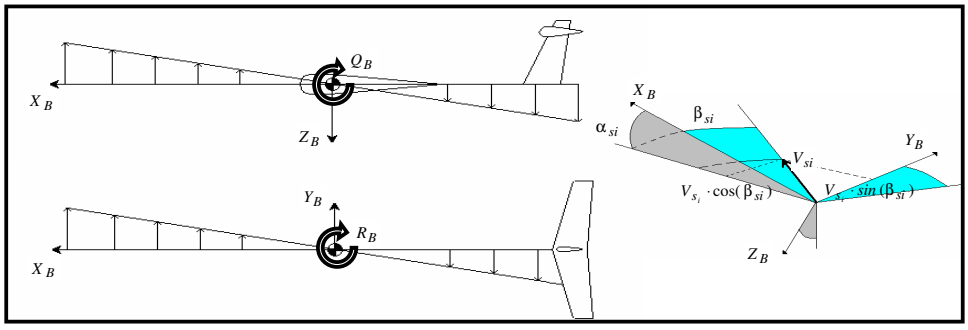

Fig. 2.3: Schematizzazione degli effetti delle velocità angolari Q_B ed R_B

2.1.3 Descrizione del blocco 3 – Calcolo degli angoli locali λ_i

Il flusso locale che si instaura in prossimità dei punti di installazione delle sonde dipende dagli angoli di incidenza α_{si} e di derapata β_{si}, dal numero di Mach M_{si} e dalla configurazione geometrica del velivolo. In questo blocco si calcolano i quattro angoli di flusso locale misurati dalle sonde.

Il calcolo viene effettuato mediante un'approssimazione ai Minimi Quadrati dei dati provenienti da prove condotte in galleria del vento transonica per la configurazione del velivolo di riferimento, che non presenta nessuna deflessione delle superfici mobili e che non prevede i carrelli rettratti, denominata di "*Cruise*" (fig. 2.4 e 2.5). Gli effetti dell'estrazione dei carrelli saranno presi in considerazione in una fase successiva quando saranno disponibili dati riguardanti le prove di volo. La procedura dell' approssimazione è descritta nel Capitolo 3.

I dati sperimentali attualmente disponibili in forma tabellare sono assimilabili a condizioni di moto rettilineo e stazionario. Questi correlano gli angoli locali λ_i con gli angoli di incidenza α e di derapata β per diversi numeri di Mach (0.20; 0.40; 0.60; 0.70; 0.80; 0.85; 0.9; 0.92; 0.95) e per diverse configurazioni del velivolo (*flaps, ailerons, nosedroop, rudder, airbrake, horizontal tail*), oltre a quella di riferimento.

Le deflessioni delle superfici introducono degli effetti (di cui si tiene conto in fase di simulazione con la costruzione delle *look-up table* di deflessione ricavate direttamente dall'interpolazione dai dati di galleria aerodinamica) sugli angoli di flusso misurati dalle sonde, espressi in termini di $\Delta\lambda_i$ rispetto alla configurazione di *Cruise* (fig. 2.6).

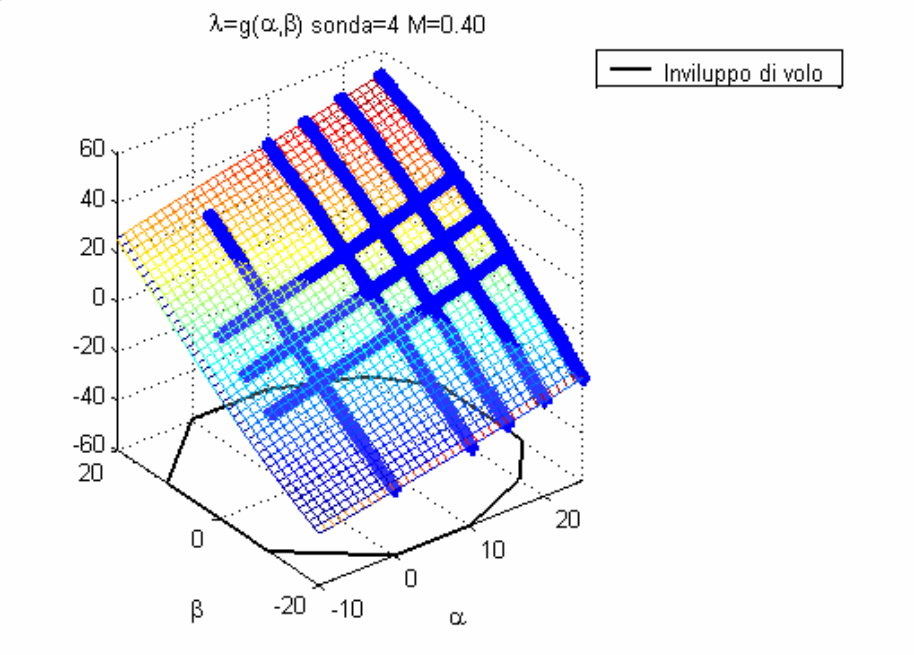

Fig. 2.4 : Angolo di flusso locale sonda4 (Config. Cruise M_∞ =0.4)

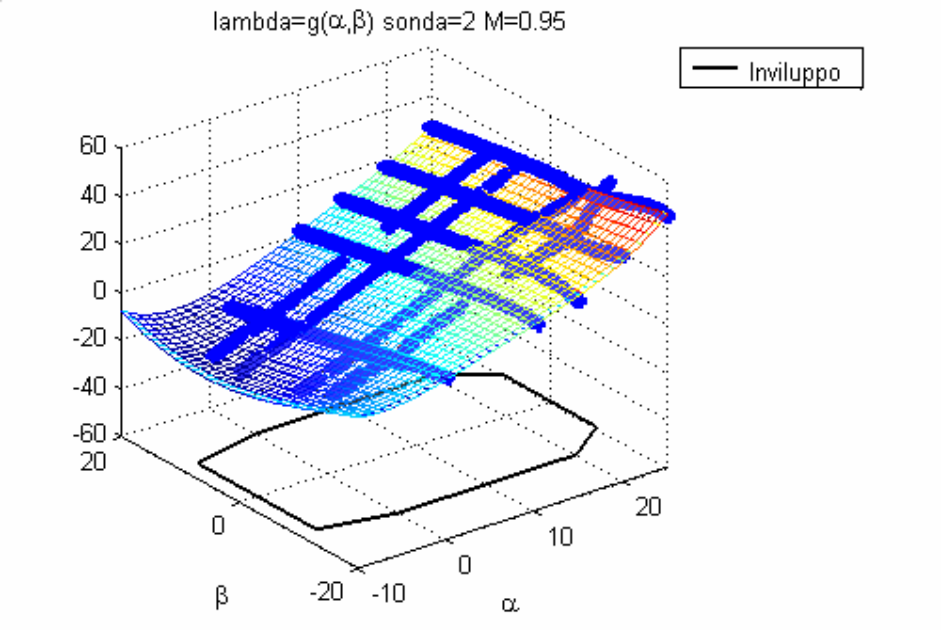

Fig. 2.5 : Angolo di flusso locale sonda2 (Config. Cruise M_∞ =0.95)

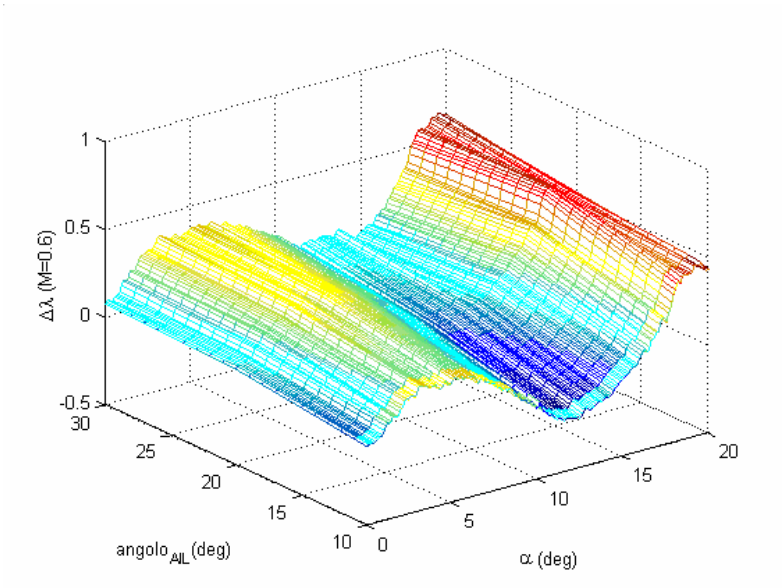

Fig. 2.6 : Variazione d' angolo di flusso locale sonda1(Config. Ailerons M_∞=0.6 $\beta = 0$)

2.1.4 Descrizione del blocco 4 – Valutazione degli M_{Li}

Il calcolo anche in questo caso, viene effettuato mediante un'approssimazione ai Minimi Quadrati, vedi Capitolo 3, di dati provenienti da prove condotte in galleria del vento transonica per la configurazione di *Cruise* (fig. 2.7 e 2.8), vedi Capitolo 3. I dati attualmente disponibili e riportati in forma tabellare come per gli angoli locali λ_i, sono assimilabili a condizioni di moto rettilineo e stazionario. Questi correlano i numeri di Mach locali M_{L_i} con gli angoli di incidenza α e di derapata β per diverse configurazioni del velivolo e numeri di Mach asintotici.

Si è tenuto conto anche degli effetti delle deflessioni delle superfici mobili in termini di ΔM_{L_i} rispetto alla configurazione di *Cruise* (fig. 2.9), con la costruzione delle *look-up table* di deflessione ricavate direttamente dall'interpolazione dai dati di galleria aerodinamica.

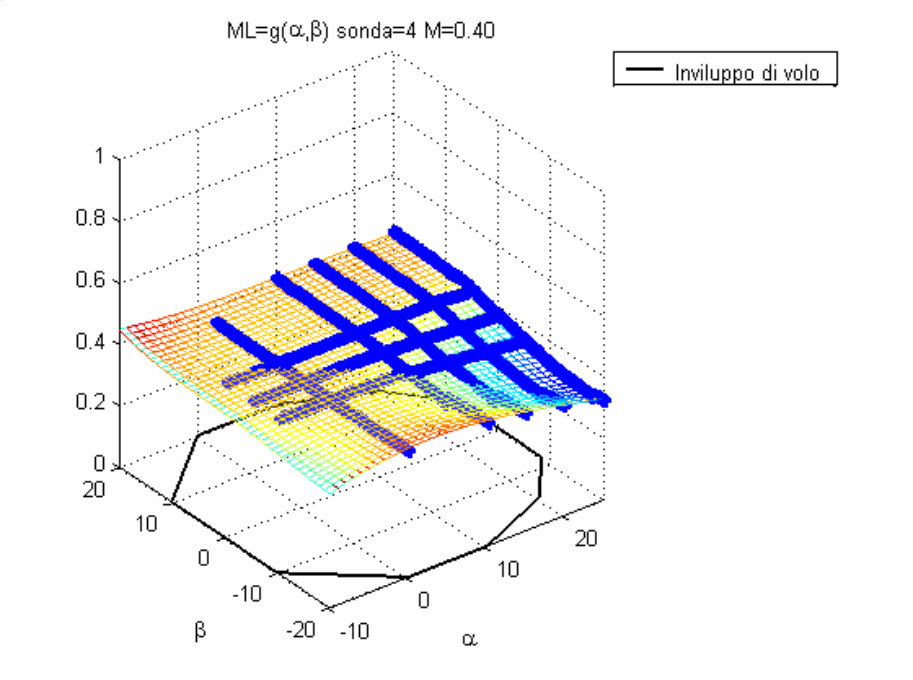

Fig. 2.7 : Mach locale sonda4(Config. Cruise M_∞ =0.4)

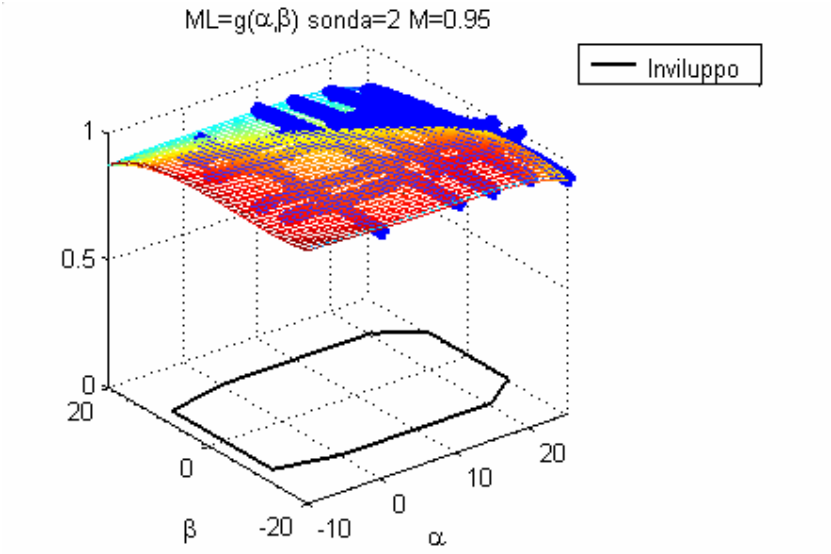

Fig. 2.8 : Mach locale sonda2(Config. Cruise M_∞ =0.95)

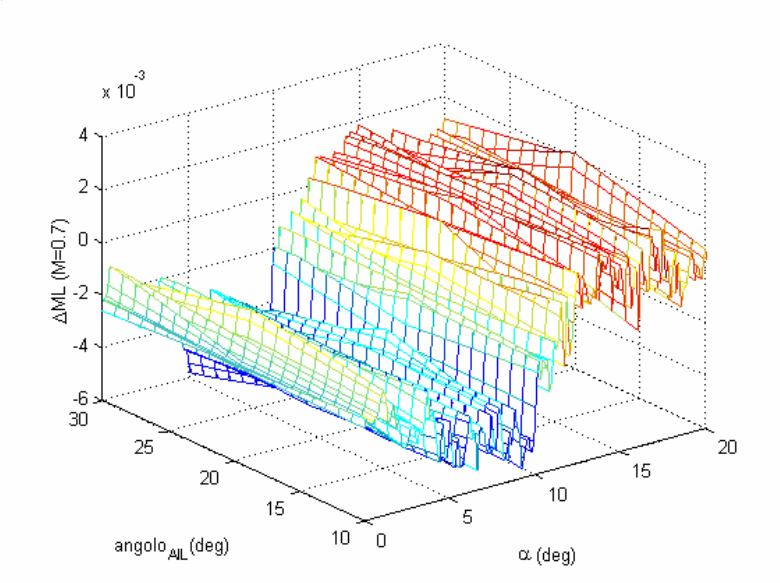

Fig. 2.9 : Variazione di Mach locale sonda1(Config. Ailerons M_∞=0.7 $\beta = 0$)

2.1.5 Descrizione del blocco 5 – Valutazione dei coefficienti di pressione Cp_{Li}

In questo blocco si calcolano i coefficienti di pressione locale in prossimità dei punti di installazione della sonda. Tali coefficienti sono definiti come:

$$Cp_{Li} = \frac{P_{Li} - P_{sa}}{\frac{\gamma}{2} \cdot P_{sa} \cdot M_{si}^2} \qquad (2.7)$$

Questi coefficienti dipendono dalle condizioni di flusso che s'instaurano nei punti di installazione delle sonde che a loro volta sono dipendenti dalla geometria del velivolo e dal flusso che investe la sezione nella quale sono situati i sensori e quindi da α_{si}, β_{si} e M_{si}.

Anche i coefficienti di pressione Cp_{Li}, come gli angoli λ_i, vengono calcolati mediante una approssimazione ai Minimi Quadrati di dati di galleria del vento (vedi capitolo 3) nella configurazione di *Cruise* (fig. 2.10 e 2.11) e mediante un'interpolazione lineare dei dati sperimentali relativi alle variazione del Cp_{Li} a seguito delle deflessioni delle superfici mobili (fig. 2.12).

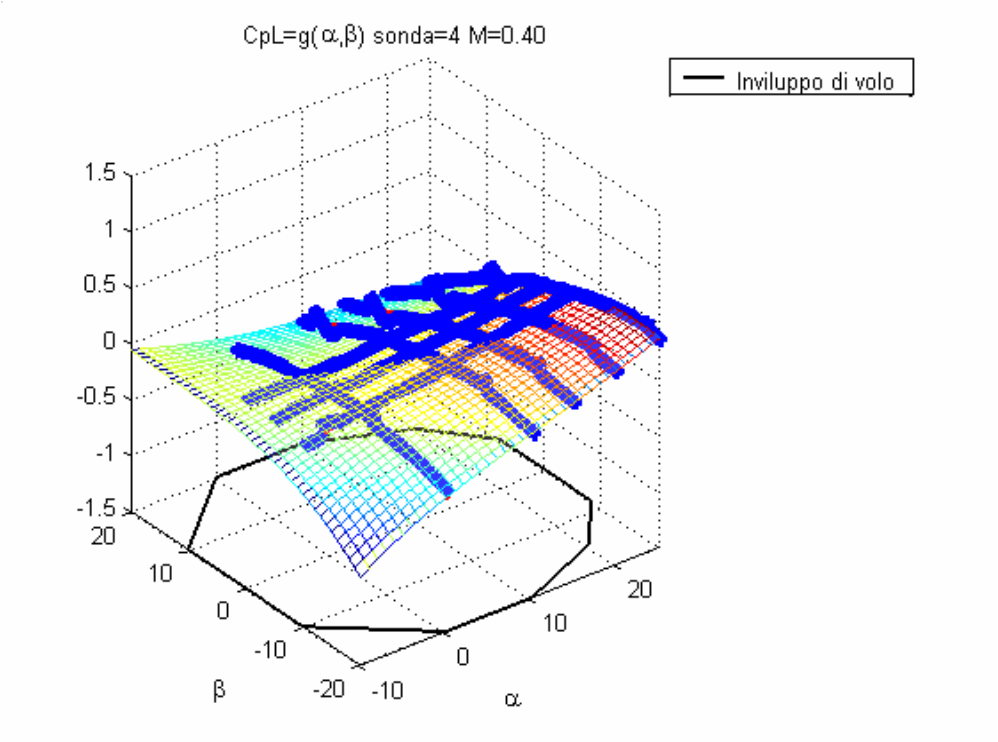

Fig. 2.10 : Cp_{Li} **sonda 4 (Config. Cruise** M_∞ **=0.4)**

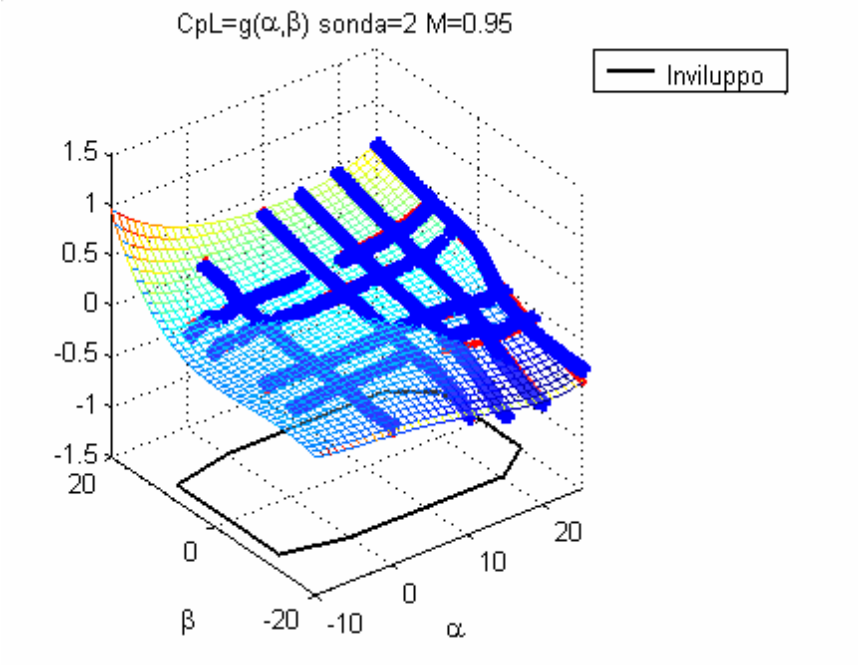

Fig. 2.11 : Cp_{Li} **sonda 2 (Config. Cruise** M_∞ **=0.95)**

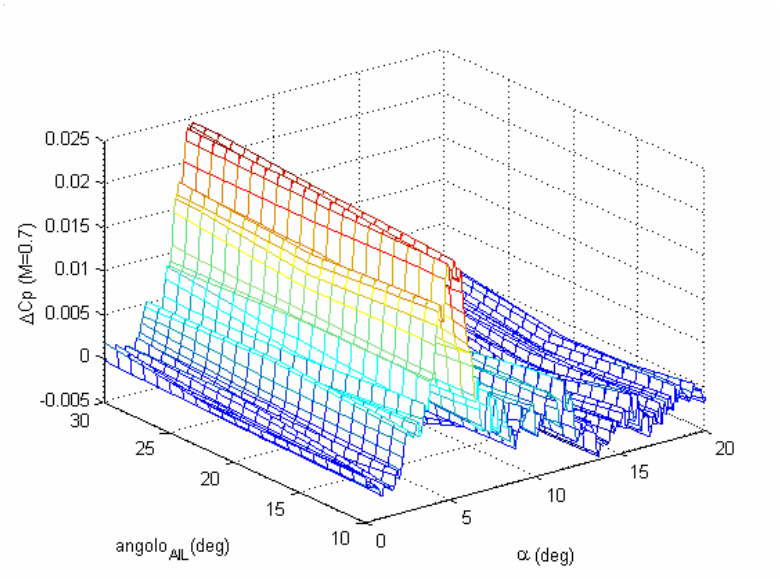

Fig. 2.12 : Variazione di Cp locale sonda 1 (Config. Ailerons M_∞ **=0.7** $\beta = 0$ **)**

2.1.6 Descrizione del blocco 6 – Valutazione di P_{Li}

In questo blocco vengono valutate le pressioni locali P_{L_i} noti i valori dei coefficienti di pressioni Cp_{Li}. La relazione che lega questi ultimi con le pressioni locali è immediatamente deducibile dalla (2.7):

$$P_{Li}(M_{si}) = P_{sa}\left(1+\frac{\gamma}{2}\cdot Cp_{Li}\cdot M_{si}^{2}\right) \tag{2.8}$$

2.1.7 Descrizione del blocco 7 – Effetto di P_B

Nel § 2.1.2 si è osservato come la presenza di componenti di velocità angolare Q_B e R_B introduca dei campi di velocità rispettivamente lungo l'asse Z_B e Y_B che provocano una differenza tra gli angoli di incidenza e di derapata relativi ad una sezione generica e quelli relativi alla sezione baricentrica. Inoltre si è osservato come tale effetto non sia prodotto dalla P_B, in altre parole, nel caso in cui si abbia solo una velocità di rotazione intorno all'asse X_B, tutte le sezioni del velivolo hanno lo stesso angolo d'incidenza e di derapata. La velocità angolare P_B, tuttavia, comporta la presenza di un campo di velocità che influenza il flusso in prossimità dei punti d'installazione delle sonde e dipendente dalla distanza, calcolata nel piano ortogonale a X_B, tra il baricentro della sezione ed il generico punto considerato.

In questo blocco si è ipotizzato che il campo di velocità indotto da P_B, comporti solo una variazione dei numeri di Mach M_{L_i} e degli angoli λ_i e non delle pressioni locali P_{L_i}. Con riferimento alla fig. 2.13 nella quale si evidenzia lo

schema utilizzato per il calcolo di M^*_{Li} e λ^*_i (che sono i valori di M_{L_i} e di λ_i comprensivi dell'effetto di P_B), si ottengono le seguenti espressioni:

$$
\begin{aligned}
V_{Li} &= M_{Li} \cdot a \\
M^*_{Li} &= \frac{\sqrt{\left(V_{Li} \cdot sen(\lambda_i) \pm P_B \cdot \sqrt{Y_{Bi}^2 + Z_{Bi}^2}\right)^2 + V_{Li}^2 \cdot \cos^2(\lambda_i)}}{a} \\
\lambda^*_i &= \arctan\left(\frac{V_{Li} \cdot sen(\lambda_i) \pm P_B \cdot \sqrt{Y_{Bi}^2 + Z_{Bi}^2}}{V_{Li} \cdot \cos(\lambda_i)}\right)
\end{aligned}
\qquad (2.9)
$$

dove il segno "+" è da applicarsi alle sonde identificate con i numeri 1 e 2, mentre il segno "–" alle sonde 3 e 4.

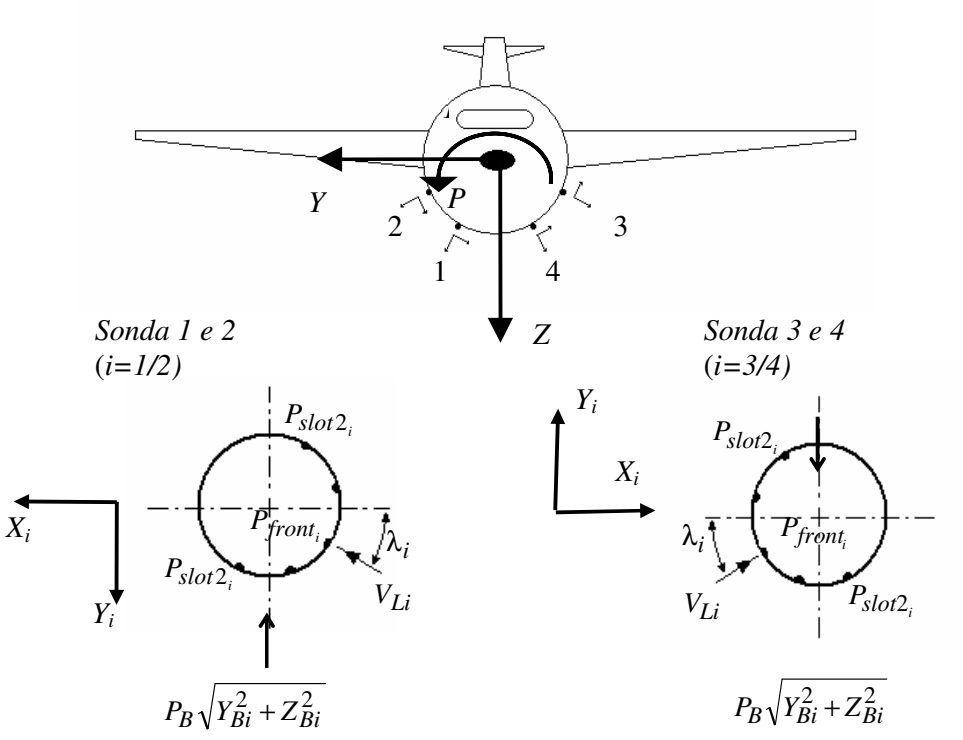

Fig. 2.13: Schematizzazione dell'effetto della velocità angolare P_B

2.1.8 Descrizione del blocco 8 – Modello sonda isolata

In questo blocco si calcolano i valori delle pressioni misurate dalle quattro sonde. Ogni sonda è dotata di cinque prese di pressione, due sono usate per orientare la sonda secondo la direzione del flusso locale e le altre tre misurano due pressioni, P_{front_i} e P_{slot2_i} (quest'ultima è misurata da due fori), che servono per effettuare una stima dei dati di volo (Mach e quota). In lavori precedenti è stato sviluppato un modello stazionario del funzionamento delle sonde in condizioni di "sonda isolata" che consente di correlare le pressioni misurate con le condizioni di flusso immediatamente a monte della sonda. Nel caso di "sonda isolata" tali condizioni coincidono con la pressione statica ambiente (P_{sa}) ed il Mach asintotico (M_∞) mentre nel caso di sonda installata in fusoliera tali condizioni sono da intendersi come quelle che si instaurano in prossimità dei punti di installazione delle sonde stesse (M_{Li}, P_{Li}), note dai dati di galleria.

Il modello sviluppato in [5] e relativo alle condizioni di sonda installata in fusoliera è costituito dalle seguenti funzioni:

$$\begin{aligned}
\frac{P_{front_i}}{P_{Li}} &= 1+\frac{\gamma}{2}\cdot M_{Li}^2\cdot\left(0.71321+0.3689\cdot M_{Li}+0.01948\cdot\frac{1}{\sqrt{1-M_{Li}^2}}\right)\\
\frac{P_{slot2_i}}{P_{Li}} &= 1+\frac{\gamma}{2}\cdot M_{Li}^2\cdot\left(-0.50517-0.01158\cdot M_{Li}-0.02509\cdot\frac{1}{\sqrt{1-M_{Li}^2}}\right)
\end{aligned} \qquad (2.10)$$

In fig. 2.14 si riportano i grafici di P_{front_i}/P_{Li} e P_{slot2i}/P_{Li} al variare del numero di Mach M_{Li}.

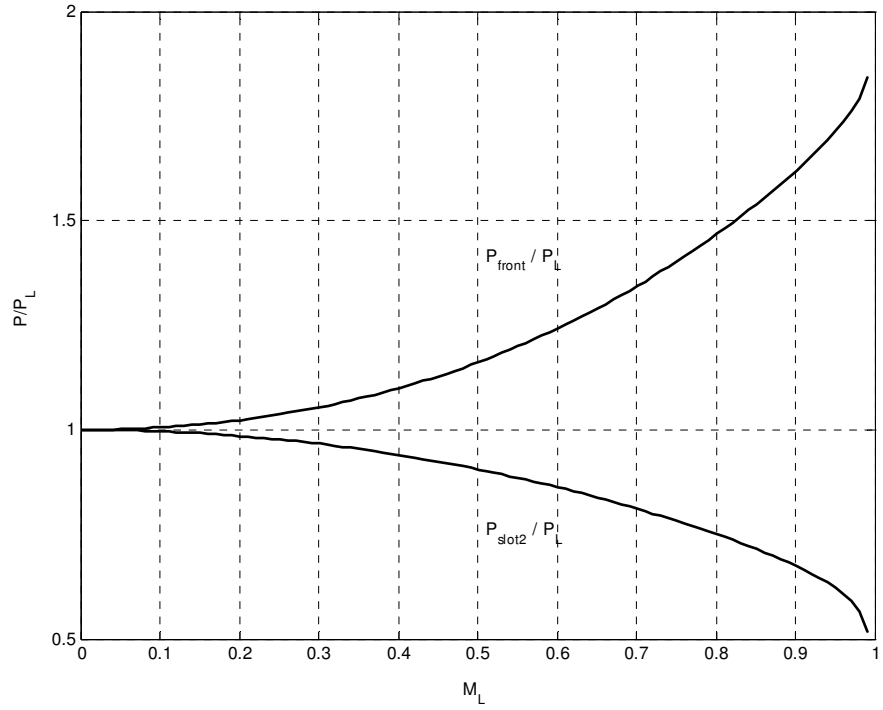

Fig. 2.14 : Modello sonda isolata

2.1.9 Descrizione del blocco 9 – Dinamica Sensori

All'interno di tale blocco è stato approssimato il modello fisico che rappresenta il funzionamento delle sonde di pressione, integrandolo con modelli capaci generare *failure* in queste.

Per la descrizione di questo blocco si rimanda al Capitolo 4.

2.2 Descrizione della procedura di simulazione del Noseboom

Il *Noseboom* (fig. 2.15) per test di volo è una sonda generalmente installata nella parte anteriore di fusoliera (fig. 2.16) che oltre ad avere le tipiche prese di pressione del *Pitot-statico* è dotato anche delle cosiddette *flow vanes* per la misurazione degli angoli (denominate in figura "*Angle of attack vane*" per la misura dell'angolo di incidenza e "*Angle of sideslipe vane*" per la misura dell'angolo di derapata).

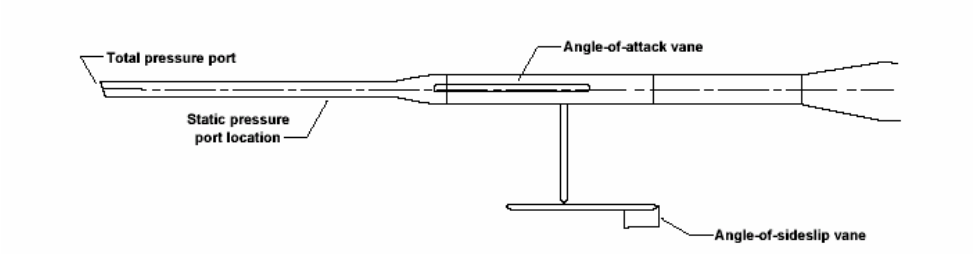

Fig. 2.15 : Noseboom

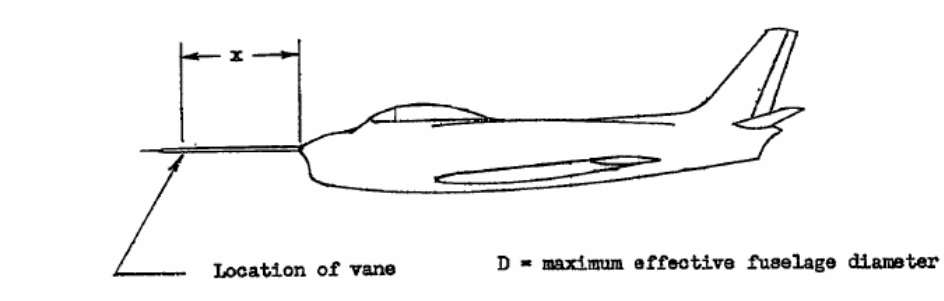

Fig. 2.16 :Installazione tipo Noseboom

L'algoritmo di simulazione dell'angolo di incidenza e di derapata misurati dal *Noseboom* è costituito da un modello che permette di ottenere gli angoli di incidenza e di derapata della sezione relativa alla posizione delle alette del *Noseboom* a partire dalle componenti del campo di velocità, analogamente a quanto fatto nel § 2.1.2 per gli angoli di incidenza e di derapata relativi alle sezioni di installazione delle sonde, α_{si}, β_{si}. Sono stati realizzati tali modelli di simulazione poiché allo stato attuale non si hanno a disposizione algoritmi di correzione che permettano di ricavare gli α e β relativi al baricentro da quelli misurati dal *Noseboom*.

Di seguito s'illustra il modello matematico derivante dalla schematizzazione che esprime la variazione di α, β in funzione delle componenti di velocità angolare Q_B e R_B. Tali componenti di velocità angolare introducono un campo di velocità nella sezione del *Noseboom* relativa alla posizione delle alette dato da:

$$\begin{aligned} u_{nb} &= u_B \\ v_{nb} &= v_B + X_{Bi} \cdot R_B \\ w_{nb} &= w_B - X_{Bi} \cdot Q_B \end{aligned} \qquad (2.11)$$

ed essendo:

$$\begin{aligned} V_{nb}^2 &= \sqrt{u_{nb}^2 + v_{nb}^2 + w_{nb}^2} \\ u_{nb} &= V_{nb} \cdot \cos(\beta_{nb}) \cdot \cos(\alpha_{nb}) \\ v_{nb} &= V_{nb} \cdot sen(\beta_{nb}) \\ w_{nb} &= V_{nb} \cdot \cos(\beta_{nb}) \cdot sen(\alpha_{nb}) \end{aligned} \qquad (2.12)$$

si ottiene che

$$\alpha_{nb} = \arctan\left(\frac{w_{nb}}{u_{nb}}\right)$$
$$\beta_{nb} = \arctan\left(\frac{v_{nb}}{\sqrt{u_{nb}^2 + w_{nb}^2}}\right) \qquad (2.13)$$

3. Ampliamento dell'inviluppo di volo

Le attività svolte hanno riguardato l'ampliamento a tutto l'inviluppo di volo α-β di interesse delle *look-up-table* di interpolazione del modello di simulazione sviluppato in tesi precedenti [4]. Il *database* aerodinamico, proveniente da prove in galleria del vento realizzare da una delle due azienze con cui il DIA collabora, fornisce le grandezze locali del flusso agenti attorno al velivolo in funzione del numero di Mach al variare degli angololi di incidenza e di derapata, ma non ricopre alcune zone del dominio di interesse. Per realizzare la simulazione dei Dati Aria in modo soddisfacente e su tutto l'inviluppo α-β di volo è stato quindi necessario interpolare i dati disponibili. Sono state sviluppate delle metologie basate sulla tecnica di Minimi Quadrati utilizzando funzioni approssimanti di tipo polinomiale. I polinomi ricavati hanno permesso di predire le grandezze di simulazione nelle zone dell'inviluppo in cui non sono disponibili dati di galleria del vento. Per poter ben vincolare i polinomi approssimanti in modo da evitare la loro divergenza in particolari zone del dominio di approssimazione sono stati introdotti ulteriori dati. Per alcuni valori del numero di Mach, infatti, i dati disponibili non sono sufficienti e sono stati aumentati utilizzando le informazioni note ad altri numeri di Mach con leggi di tipo *Prandtl-Meyer.*

L'inviluppo di volo espresso in termini di angolo di incidenza (α) e derapata (β) è stato ampliato ad un intervallo di α tra -10^o e $+25^o$ e di β tra -20^o e $+20^o$ rispetto a quello precedente di α tra 0^o e $+20^o$ e di β tra -4^o e $+4^o$, vedi fig. 3.1.

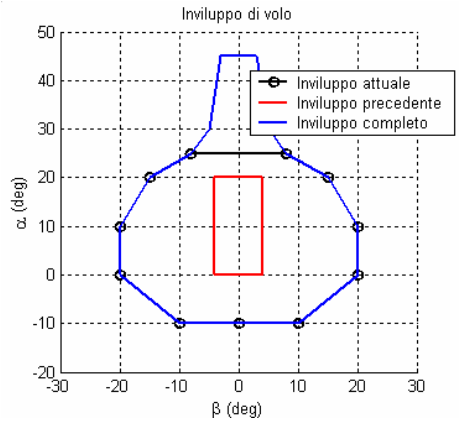

Fig. 3.1 : Inviluppo di volo di riferimento

Le grandezze necessarie alla procedura di simulazione, di cui si parla sopra, si riferiscono alla configurazione del velivolo in cui tutte le superfici di controllo presentano deflessioni nulle e i carrelli sono retratti, che corrisponde alla configurazione di Crociera.

Questa fase è stata necessaria per fissare definitivamente la banca dati aerodinamica su cui si basa il modello relativo alla procedura di simulazione del sistema Dati Aria.

3.1 Definizione del problema

Il *database* proveniente dalle prove in galleria del vento fornisce i valori sperimentali sull'inviluppo in α-β in termini di valori ad α costante e β variabile e β costante e α variabile, come si può vedere per esempio in fig. 3.2, in cui sono riportati i valori di λ della sonda 2 derivanti dalle prove in galleria del vento.

Come si può intuire dall'esempio in fig. 3.2 mancano di fatto i dati di galleria nelle zone dell'inviluppo di volo che si allontanano dall'intervallo di α tra 0^o e

+20° e di β tra -4° e +4°. Non essendoci la possibilità di ulteriori prove sperimentali, per avere dati utilizzabili su un intervallo in α-β più esteso si è pensato di ricorrere ad algoritmi di approssimazione di funzione che utilizzassero soltanto i dati a disposizione.

Con i valori sperimentali a disposizione e così distribuiti sull'inviluppo, applicando metodi di interpolazione si ottengono funzioni del tipo $f = f(\alpha, \beta)$ che non coprono bene quelle parti di inviluppo che non sono sufficientemente vicine ai dati sperimentali.

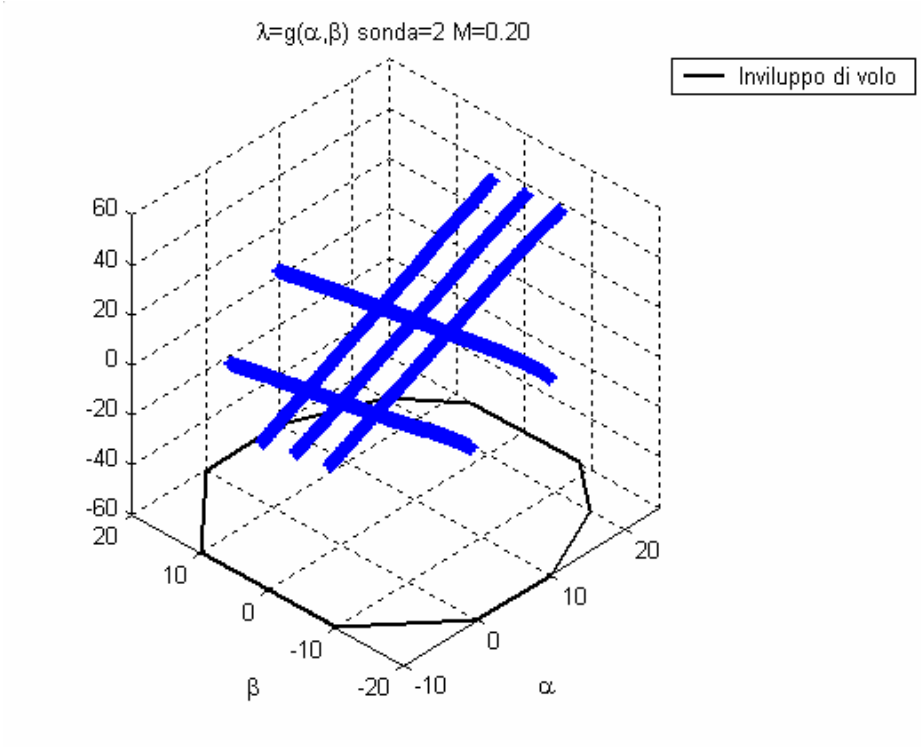

Fig. 3.2: Valori sperimentali dell'angolo di flusso locale, λ

.

Per risolvere questo problema si è dunque pensato di ricorre a tecniche di approssimazione di funzione, che permettono di filtrare i dati di galleria. Tra queste è stato scelta la tecnica di approssimazione ai Minimi Quadrati.

3.2 Problematiche relative all'approssimazione ai Minimi Quadrati

3.2.1 Problematiche a Mach minore uguale di 0.4

La campagna di prove in galleria del vento è stata effettuata a diversi numeri di Mach, in particolare a Mach 0.2, 0.4, 0.6, 0.7, 0.8, 0.85, 0.9, 0.92, 0.95. Numeri di Mach superiori non sono di interesse perché il velivolo in esame non è un velivolo supersonico.

Per Mach 0.2 e 0.4 sono state effettuate prove con il velivolo ad incidenza (α) costante facendo variare l'angolo di derapata (β) ed anche prove ad angolo di derapata (β) costante facendo variare l'incidenza (α), come si può intuire dalla fig. 3.2. Quest'ultime non sono state eseguite ai numeri di Mach superiori.

Inizialmente la tecnica di approssimazione ai Minimi Quadrati è stata usata per calcolare le grandezze necessarie alla procedura di simulazione, che si ricorda essere il coefficiente di pressione locale (Cp_{Li}), il Mach locale (M_{Li}) e l'angolo di deviazione del flusso locale (λ_i), sull' inviluppo di volo in α-β per le stazioni a Mach 0.2 e 0.4, poiché esse fornivano più dati sperimentali rispetto alle successive.

Si è scelto un polinomio del quarto grado (3.1) per approssimare con la tecnica dei Minimi Quadrati i valori di galleria a disposizione.

$$P(\alpha,\beta) = c_0 + c_1\alpha + c_2\beta + c_3\alpha\beta + c_4\alpha^2 + c_5\beta^2 + c_6\alpha\beta^2 + c_7\alpha^2\beta + c_8\alpha^3 + c_9\beta^3 + c_{10}\alpha\beta^3 + c_{11}\alpha^2\beta^2 + c_{12}\alpha^3\beta + c_{13}\alpha^4 + c_{14}\beta^4 \quad (3.1)$$

Usando tutti i valori sperimentali disponibili per la configurazione di Crociera a Mach 0.2 e 0.4, il polinomio approssimante con la tecnica dei Minimi Quadrati non risultava essere molto vincolato poiché il polinomio tendeva a divergere laddove si allontanava dai dati sperimentali.

Quindi per vincolare maggiormente il polinomio, ai precedenti detti numeri di Mach si è pensato di utilizzare un set di valori sperimentali che non si riferivano alla configurazione di Crociera, ma ad una condizione di manovra caratterizzata da una deflessione del *nose droop* piccola ma diversa da zero. E' stata fatta l'ipotesi che per piccole deflessioni del *nose droop* i valori delle grandezze locali variassero di poco rispetto alla configurazione in cui il velivolo non presenti superfici deflesse.

Sono stati utilizzati i valori sperimentali ad α costante, pari a 15^o, relativi ad una deflessione del *nose droop* di -10^o ed i valori ad α costante, pari a 20^o e 25^o relativi ad una deflessione del *nose droop* di -20^o, ottenendo una griglia di dati sufficiente su cui appoggiare il polinomio approssimante, come si vede in fig. 3.3.

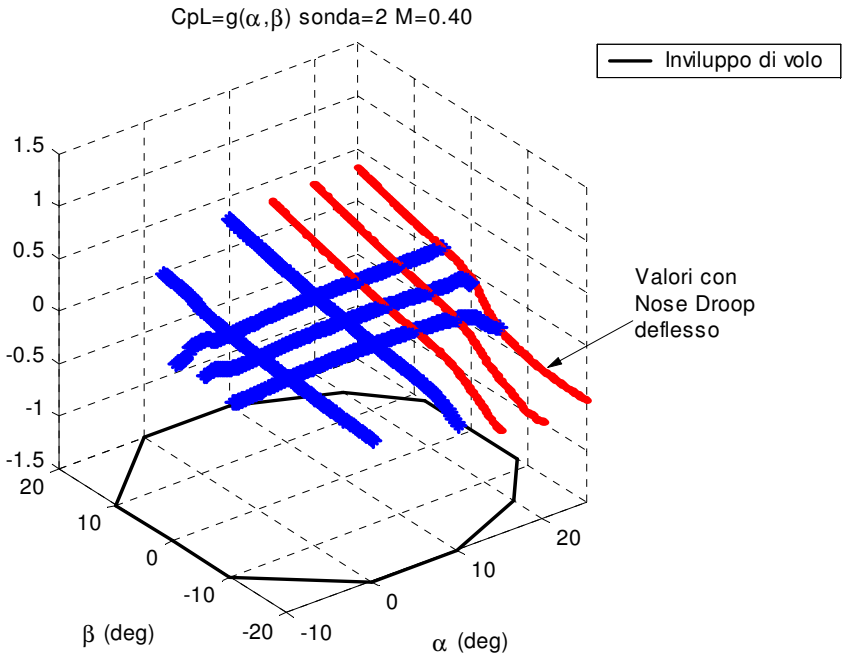

Fig. 3.3: Valori sperimentali per l'approssimazione ai Minimi Quadrati

Una scelta del genere è stata dettata dall'esigenza che ulteriori prove di galleria non potevano essere fatte e poiché alle alte incidenze ,come possono essere 15^o,20^o,25^o il pilota generalmente aziona il *nose droop*.

In questo modo l'approssimazione ai Minimi Quadrati è risultata soddisfacente con errori medi tra i valori sperimentali e i valori approssimati in tuttol'inviluppo di volo e per ogni Mach, su λ_i dell'ordine del 0.5^o ,su Cp_{Li} dell'ordine di 0.02 e sul M_{Li} dell'ordine di 0.002.

In fig. 3.4. è riportato un esempio di polinomio approssimante a numeri di Mach pari a 0.4.

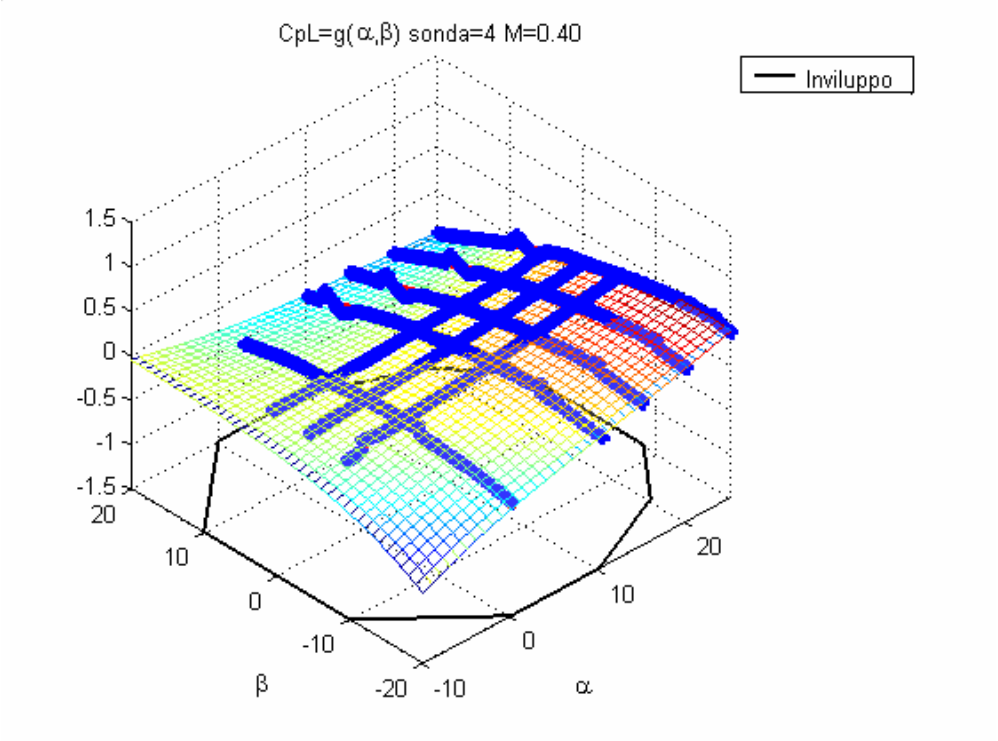

Fig. 3.4 : Coefficiente di pressione locale sonda4 (Config. Cruise M_∞ =0.4)

3.2.2 Problematiche a Mach maggiore di 0.4

Per quanto riguarda l'approssimazione ai Minimi Quadrati a valori di Mach superiori a 0.4, i dati di galleria disponibili per la configurazione di Crociera si riferiscono soltanto a prove sperimentali con velivolo ad angolo di derapata (β) costante ed incidenza (α) variabile, come si può vedere in fig. 3.5.

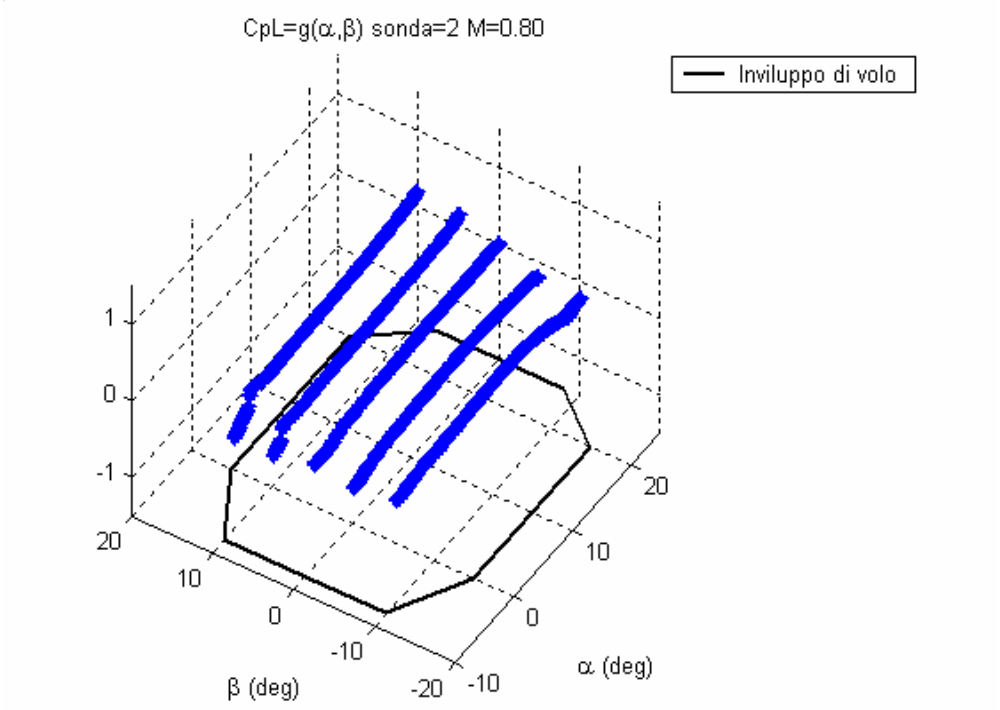

Fig. 3.5 : Valori sperimentali del coefficiente di pressione locale, Cp_{Li}

Effettuando l'approssimazione ai Minimi Quadrati, il polinomio approssimante, anche in questo caso, risultava essere poco vincolato poiché tendeva ad oscillare troppo al variare di β.

Data l'impossibilità di effettuare nuove prove di galleria del vento e non avendo a disposizione dati sperimentali ad α costante e β variabile relativi ad

angoli di *nose droop* diversi da zero per numeri di Mach superiori a 0.4, è stata presa la decisione di provare ad utilizzare i dati sperimentali ad α costante e β variabile usati per l'approssimazione ai Minimi Quadrati a Mach 0.4. I dati presi in considerazione sono quelli relativi ad α costante, pari a 0^o e 10^o, con angoli del *nose droop* uguale a 0^o, ad α costante, pari a 15^o, con angoli del *nose droop* uguale a -10^o e ad α costante, pari a 20^o e 25^o, con angoli del *nose droop* uguale a -20^o.

Questo tipo di scelta non si è rivelata inopportuna poiché è stato osservato che, per quanto riguarda λ_i e Cp_{Li} le curve di dati di galleria ad α costante e β variabile di Mach 0.4 si concatenavano abbastanza bene con le curve di valori di galleria a β costante e α variabile di Mach superiori a 0.4, come si può vedere per esempio in fig. 3.6.

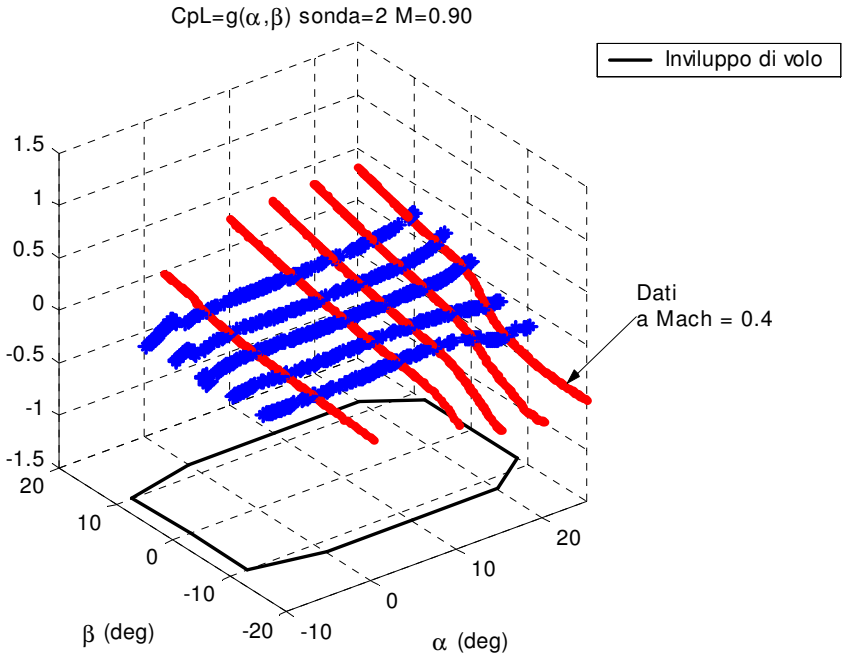

Fig. 3.6 : Valori sperimentali del coefficiente di pressione locale, Cp_{Li}

Questa ultima considerazione ha permesso di ritenere che i dati di galleria a α costante e β variabile di Mach 0.4 potevano essere effettivamente usati anche per le approssimazioni ai Minimi Quadrati a Mach maggiori.

Comportamento del tutto diverso invece, per il M_{Li} poiché esso chiaramente varia quando cambia il numero di Mach del flusso asintotico. Quindi per il M_{Li} non era possibile pensare di utilizzare semplicemente le strisciate di valori ad α costante e β variabile di Mach 0.4.

Alla luce di quanto riscontrato, la via più corretta per utilizzare i dati di galleria ad α costante a Mach 0.4 di Cp_{Li}, λ_i e M_{Li} è sembrata essere quella di valutare l'errore medio di queste grandezze nei punti di incontro tra le curve di valori sperimentali ad α variabile ai numeri di Mach maggiori di 0.4 e quelle a β variabile a Mach 0.4 e traslare quest'ultime di tale errore.

In sostanza è stata effettuata una correzione, riconducibile a quella di Prandtl - Meyer dei dati di galleria a Mach 0.4 per ottenere una buona stima degli stessi dati a numeri di Mach superiori.

Sono stati ottenuti dunque, per λ_i, Cp_{Li} e M_{Li} dei dati di galleria fittizi per i Mach maggiori di 0.4; per maggiori dettagli consultare il Capitolo 5 dove viene riportata una rapida descrizione del file *LUT_V_generator.m* che fornisce i dati di galleria virtuali.

Con questi nuovi dati il polinomio approssimante ai Minimi Quadrati è risultato soddisfacente sia per λ_i, sia per Cp_{Li} che per il M_{Li}, anche ai numeri di Mach superiori a 0.4.

Gli errori medi tra i valori sperimentali e i valori approssimati in tuttol'inviluppo di volo e per ogni Mach sono stati valutati dell'ordine di 0.5^o per λ_i,dell'ordine di 0.03 per Cp_{Li} e dell'ordine di per 0.003 per M_{Li}.

In fig. 3.7. è riportato un esempio di polinomio approssimante ai Minimi Quadrati a numeri di Mach maggiori di 0.4.

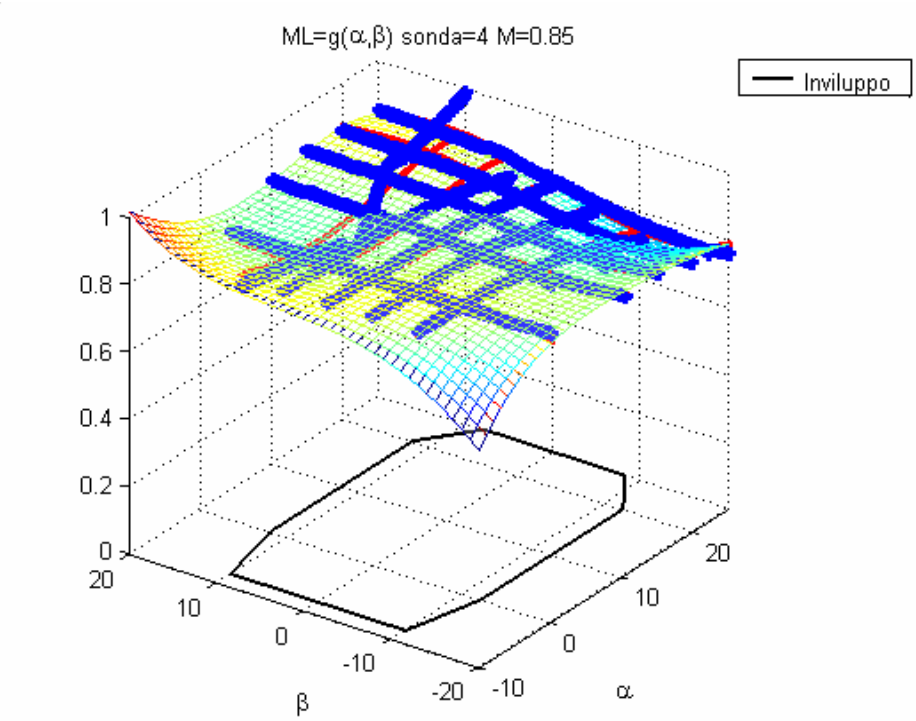

Fig. 3.7 : Mach locale sonda4 (Config. Cruise M_∞ =0.85)

3.2.3 Dati di galleria della sonda 1

Nella fase di generazione dei dati è emerso un ulteriore problema, si è osservato che i valori sperimentali della sonda 1, relativi a condizioni di α costante e β variabile presentavano andamenti anomali. Per β maggiori di $+5^{o}$ e β minori di -5^{o} i valori del Cp_{Li} presentano forti cadute di pressione di non facile spiegazione perché non riconducibili a fenomeni di distacchi del flusso nei punti di istallazione delle sonde.

In fig. 3.8 si riportano i valori sperimentali della sonda 1 e della sonda 4 opportunamente ribaltati rispetto a β per mettere in evidenza questa anomalia dei dati della sonda 1 anche da un altro punto di vista. La curva dei dati sperimentali della sonda 1 infatti dovrebbe avere un andamento non molto diverso da quello della curva dei dati sperimentali opportunamente ribaltati rispetto a β della sonda 4.

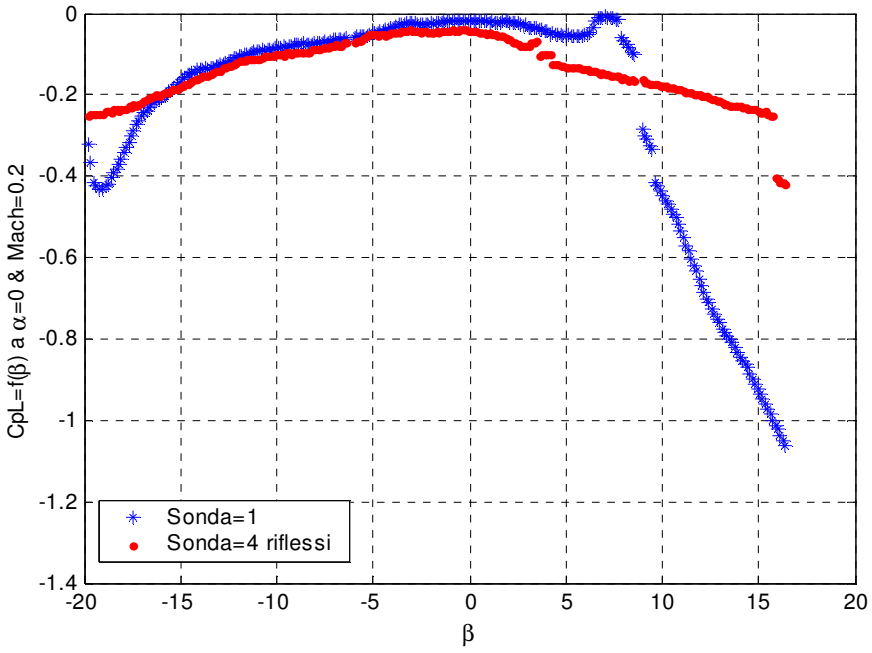

Fig. 3.8 : Valori sperimentali della sonda 1 e della sonda 4

In fig. 3.8 si nota in particolare che il Cp_{Li} ha un andamento non facilmente giustificabile dal punto di vista fisico specialmente agli estremi dell'intervallo del dominio in β poiché, per esempio a $\beta > 0^o$, è difficile pensare che la sonda uno quando si trova in luce rispetto alla direzione del flusso possa misurare una differenza di pressione $P_{Li} - P_{sa}$ che comporti un Cp_{Li} dell'ordine del meno uno.

Un comportamento del genere può essere spiegato invece, dicendo che è probabile che durante le prove di galleria la sonda uno sia stata calettata in modo errato.

Questa anomalia dei dati sperimentali della sonda 1 è stata segnalata alla industria aeronautica italiana che ha efettuato le prove di galleria e questa ha confermato le supposizioni fatte.

Data l'impossibilità di effettuare nuove prove sperimentali e stato deciso ,su suggerimento dell'azienda stessa, di risolvere il problema utilizzando per la sonda uno i dati sperimentali della sonda quattro opportunamente riflessi rispetto a β.

4. Modellizzazione *failure*

Per completare la procedura di simulazione del sistema Dati Aria, vedi fig. 2.2, è necessario considerare dei modelli capaci di simulare la presenza di *failure* nelle sonde e nel *Noseboom*.

Questi modelli sono stati integrati nel s*oftware* di simulazione in modo da simulare le perdite di prestazioni del sistema in seguito all'instaurarsi di condizioni di avaria. Tali modelli sono stati realizzati in ambiente *Matlab-Simulink®*.

4.1 Dinamica dei Sensori

I modelli in grado di generare *failure* nelle sonde sono stati implementati all'interno del blocco denominato *Dinamica dei Sensori* dell'algoritmo di simulazione rappresentato in fig. 2.2.

All'interno di tale blocco infatti, è realizzato il modello fisico che rappresenta il funzionamento di una sonda di pressione.

Una singola sonda, come già detto in precedenza, fornisce in uscita una misura angolare (λ_i) e due misure di pressione (P_{front_i} e P_{slot2_i}). La dinamica di rotazione con la quale la sonda si allinea alla direzione locale del flusso influisce inevitabilmente sulla lettura della sonda stessa ed in particolare modo quest'ultima è influenzata dal tempo che la sonda impiega per allinearsi. D'altra parte, anche gli effetti relativi alla dinamica dei trasduttori di pressione e di posizione angolare condizionano le misure rilevate dalla sonda.

Nella procedura di simulazione si è assunto che sulla misura degli angoli locali influisca solo l'effetto dinamico di rotazione della sonda (trasduttore d'angolo). Per tenere conto di quest'effetto si è approssimata tale dinamica con un modello

fisico la cui legge di moto è data da un'equazione differenziale lineare del secondo ordine (del tipo massa – molla – smorzatore):

$$\begin{aligned} &\ddot{\lambda}_{dyn_i} + 2\zeta\omega\dot{\lambda}_{dyn_i} + \omega^2 \cdot \lambda_{dyn_i} = \omega^2 \cdot \lambda_i^* \\ &\zeta = 0.8 \\ &\omega = 10 \quad \text{(rad/sec)} \end{aligned} \tag{4.1}$$

Per quanto riguarda le misure di pressione si è, invece, considerato il ritardo dovuto alla comprimibilità dell'aria nel tubicino che collega una presa di pressione con il realtivo trasduttore. Non è stato preso in considerazione il ritardo associato agli effetti dinamici del trasduttore di pressione poiché questi sono talmente piccoli da poter essere trascurati. Per tenere del ritardo pneomatico si è approssimata tale dinamica con un modello fisico la cui legge di moto è data da un'equazione differenziale lineare del primo ordine:

$$\begin{aligned} &\dot{P}_{front\_dyn_i} + \frac{1}{\tau} \cdot P_{front\_dyn_i} = \frac{1}{\tau} \cdot P_{front_i} \\ &\dot{P}_{slot2\_dyn_i} + \frac{1}{\tau} \cdot P_{slot2\_dyn_i} = \frac{1}{\tau} \cdot P_{slot2_i} \\ &\tau = 0.025 \quad \text{(sec)} \end{aligned} \tag{4.2}$$

Gli effetti degli errori di allineamento $\lambda_{dyn_i} - \lambda_i^*$ sulle pressioni P_{front_i} e P_{slot2_i} sono stati presi in considerazione avendo a disposizione le informazioni necessarie, vedi [6], per conoscere come variano le pressioni suddette con l'errore sopra specificato. Tale errore è stato considerato nella procedura di simulazione come se fosse dovuto ad una avaria del trasduttore di posizione angolare come descritto in § 4.2.3.

4.2 Failure dei sensori

Per individuare le possibili avarie della sonda multi-funzione considerata (vedi fig. 1.4) si è analizzato il suo funzionamento. Sono state identificate le avarie relative al trasduttore d'angolo, quelle del trasduttore di pressione e si è analizzato come le une e le altre potessero interagire tra di loro.

Le avarie sono state individuate andando ad analizzare quali potessero essere le possibili cause di un malfunzionamento delle sonde e come queste cause potessero influenzare i segnali di uscita dei sensori.

4.2.1 *Failure* del trasduttore d'angolo

Per analizzare le possibili avarie del trasduttore d'angolo occorre ricordare come descritto nel Capitolo 1, che la dinamica di allineamento della sonda al flusso viene indotta da una differenza di pressione rilevata dalle due *slot1* ed evolve in modo tale da ripristinare una condizione di equilibrio tra le due pressioni.

Le possibili avarie al trasduttore d'angolo introdotte nel modello sono state dedotte sostanzialmente facendo riferimento allo schema della sonda riportato in fig. 1.4, non avendo a disposizione ulteriori disegni più dettagliati e non avendo la possibilità di effettuare prove sperimentali. Nonostante ciò le *failure* prese in considerazione sembrano essere sufficienti per rappresentare le possibili perdite di prestazioni delle sonde.

Le *failure* implementate nel modello sono le seguenti:

1) *Accuracy*, inserimento di un rumore di intensità variabile sul segnale della posizione angolare ;

2) *Offset*, alterazione della posizione angolare di una quantità stabilita al momento dell'inserimento dell'avaria;

3) *Hysteresis,* introduzione di un comportamento di isteresi della posizione angolare;

4) *Departure,* raggiungimento rapido di fondo corsa (superiore o inferiore) e bloccaggio a tale valore della posizione angolare;

5) *Lock*, bloccaggio della posizione angolare al valore assunto al momento dell'inserimento dell'avaria;

6) *End-stroke limits*, modifica dei limiti di fondo corsa della posizione angolare della sonda.

7) *Frequency*, modifica della frequenza del modello del secondo ordine che approssima la dinamica del trasduttore di posizione angolare.

8) *Time costant*, modifica della costante di tempo del modello del secondo ordine che approssima la dinamica del trasduttore di posizione angolare.

9) Validity, modifica del segnale di validità associato al segnale della misura di posizione angolare.

Tali *failure* sono state introdotte nel modello tenendo conto della possibilità che una singola sonda possa essere soggetta a più *failure* contemporanee, anche se queste sonde sono state progettate con una bassa probabilità di avaria.

Di seguito è riportata una descrizione delle possibili cause che possono provocare le *failure* prese in esame. Le motivazioni relative alle varie *failure* sono il risultato di una riflessione sul funzionamento meccanico ed elettronico delle sonde.

Accuracy, l'aggiunta di un rumore di intensità variabile al segnale della posizione angolare è stata implementata per tenere conto :

- di un aumento o di una diminuzione dell'accuratezza relativa alla posizione angolare misurata dalla sonda, che di è +/- 0.36^{o};
- di una eventuale disturbo elettrico sul segnale.

Offset, l'alterazione della posizione angolare di una quantità stabilità potrebbe essere dovuta:

- ad un eventuale calettamento non corretto della sonda;
- ad un'improvvisa otturazione parziale di una o di entrambe le prese di pressione *slot1*;
- ad un'improvvisa rottura parziale del canale o dei canali che portano il flusso alla capsula del trasduttore ad esempio a causa del suo deterioramento.

Hysteresis*,* un comportamento di isteresi della posizione angolare è stato introdotto per tener conto :

- della eventuale presenza di un certo gioco meccanico;
- di un eventuale aumento di attrito per la presenza di sporcizia tra le superfici mobili o per un deterioramento dei cuscinetti su cui l'asse della sonda ruota.

Departure, avendo presente la tipologia della sonda le possibili cause che potrebbero portarla a fondo corsa rapidamente con relativo bloccaggio potrebbero essere:

- l'otturazione completa o parziale di una presa di pressione (*slot1*) con l'altra normalmente o, eventualmente, parzialmente funzionante

causata per esempio da sporcizia accumulata o depositata accidentalmente;

- la rottura interna, completa o parziale, di un canale relativo ad una *slot1* con l'altro normalmente o, eventualmente, parzialmente funzionante dovuta per esempio al deterioramento dei canali che guidano il flusso alle palette di allineamento.

***Lock*,** le possibili cause invece, che potrebbero bloccare la sonda potrebbero essere:

- l'otturazione completa o parziale di entrambe le *slot1* dovuta per esempio a sporcizia accumulata o depositata accidentalmente;
- la presenza di sporcizia accumulata o depositata accidentalmente tra la superficie esterna della sonda e la fusoliera che ne impedisce il movimento;
- la rottura delle palette di allineamento dovuta per esempio al loro deterioramento;
- la rottura dei cuscinetti su cui l'asse della sonda ruota, per esempio per deterioramento di quest'ultimi.

End-stroke limits, la modifica dei limiti di fondo corsa della posizione angolare della sonda, fissati dal costruttore in +/- 50^{o}, è stata prevista per tenere conto:

- di eventuali rotture dei fermi che limitano la massima posizione angolare della sonda dovute per esempio al loro deterioramento;
- della eventuale presenza di sporcizia o delle eventuali alterazioni dimensionali della sonda o della fusoliera che potrebbero provocare una limitazione dell'escursione della posizione angolare rispetto ai valori nominali.

Il modello della dinamica di allineamento del sensore è stato realizzato in modo da poter modificare durante una simulazione i parametri (ω,ξ) che caratterizzano il modello del secondo ordine, andando ad agire sulla frequenza e la costante di tempo che sono legati a ω,ξ dalle relazioni: $f = \omega/2\pi$ [Hz], $\tau = 1/\xi\omega$ [s].

Frequency**,** un variazione della frequenza rispetto al valore nominale (1.59 Hz) corrisponde ad una variazione della banda passante del sensore che può essere dovuta:

- all'otturazione parziale o alla rottura parziale, rispettivamente di entrambe le prese di pressione (*slot1*) o dei canali relativi a tali prese di pressione per la presenza di sporcizia o deterioramento dei canali;
- un aumento di attrito tra la superficie mobile della sonda e quella collegata rigidamente alla fusoliera dovuta per esempio alla presenza di sporcizia.

Time costant*,* un'alterazione della costante di tempo rispetto al suo valore nominale comporta un'alterazione del tempo di assestamento del sistema dinamico del secondo ordine con il quale è stato approssimato il trasduttore di posizione angolare.

Validity*,* questo tipo di *failure* permette di settare il segnale di validità associato alla posizione angolare nella modalità invalida a partire dalla modalità di *dafault* di validità e viceversa.

4.2.2 *Failure* del trasduttore di pressione

I due trasduttori di pressione, quello relativo alla P_{front} e quello relativo alla P_{slot2} sono uguali e perciò (§ 4.1) le avarie inserite nel modello sono identiche per entrambi i trasduttori.

Le *failure* implementate nel modello sia per la P_{front} che P_{slot2} sono le seguenti:

1) *Accuracy*, inserimento di un rumore di intensità variabile sul segnale della pressione;

2) *Offset*, alterazione della pressione di una quantità stabilita al momento dell'inserimento dell'avaria;

3) *Hysteresis,* introduzione di un comportamento di isteresi sul segnale della pressione;

4) *Departure,* raggiungimento rapido del fondo scala (superiore o inferiore) e bloccaggio a tale valore della pressione;

5) *Lock*, bloccaggio della pressione al valore assunto al momento dell'inserimento dell'avaria;

6) *End-stroke limits*, modifica dei limiti del fondo scala della pressione della sonda.

7) *Delay*, modifica della ritardo del modello del primo ordine che approssima il trasduttore di pressione.

8) *Validity,* modifica del segnale di validità associato al segnale della misura di pressione.

In modo analogo a quanto fatto per il trasduttore d'angolo, il modello prevede che più *failure* possano essere attivate in sequenza, simulando tutte le possibili combinazioni di avarie fisicamente possibili.

Di seguito è riportata una descrizione delle possibili cause che possono provocare le *failure* prese in esame.

Accuracy, l'aggiunta di un rumore di intensità variabile al segnale della pressione è stata implementata per tenere conto :

- di un aumento o di una diminuzione dell'accuratezza relativa alle pressioni misurate dalla sonda, che è di +/- 0.61 [Pa] per la P_{front} e di +/- 25 [Pa] per la P_{slot2};
- di una eventuale disturbo elettrico sul segnale.

***Offset*,** l'alterazione della pressione potrebbe essere dovuta:

- ad un otturazione parziale di una presa di pressione: per la P_{front} dell'otturazione parziale della sua singola presa, per la P_{slot2} dell'otturazione parziale di una sola presa o di entrambe.
- ad una rottura parziale del canale che porta il flusso alla capsula del trasduttore ad esempio a causa del suo deterioramento;

Hysteresis*,* un comportamento di isteresi della pressione è stato introdotto per tener conto di:

- eventuale presenza di un certo gioco o attrito meccanico nella capsula del trasduttore di pressione;

Departure**,** le possibili cause che potrebbero portarla a fondo corsa rapidamente con relativo bloccaggio potrebbero essere:

- l'accumulo di pressione all'interno della capsula del trasduttore senza la possibilità di scarico (fondo scala superiore);
- una perdita di pressione improvvisa dovuta alla rottura della capsula del trasduttore (fondo scala inferiore).

Lock, Il trasduttore di pressione potrebbe fornire una pressione costante per la seguenti cause:

- Otturazione completa delle prese di pressione: di entrambe le prese *slot2* per la *slot2 pressure* e della presa *front* per la *front pressure,* dovute per esempio a sporcizia accumulata o depositata accidentalmente;
- Rottura completa del canale che porta il flusso alla capsula del trasduttore ad esempio a causa del suo deterioramento.

End-stroke limits, la modifica dei limiti di fondo scala della pressione della sonda, fissati dal costruttore per la P_{front} a 5079.58 [Pa] per il limite inferiore e a 304775 [Pa] per il limite superiore, per la P_{slot2} a 2709.11 [Pa] per il limite inferiore e a 111750.84 [Pa] per il limite superiore, è stata prevista per tenere conto:

- di eventuali danneggiamenti della capsula del trasduttore di pressione;

Delay, la variazione del ritardo sul segnale delle pressione potrebbe essere dovuto:

- ad eventuali malfunzionamenti del sistema elettrico;

- ad un otturazione parziale di una presa di pressione: per la P_{front} dell'otturazione parziale della sua singola presa, per la P_{slot2} dell'otturazione parziale di una sola presa o di entrambe.

Validity, questo tipo di *failure* permette di settare il segnale di validità associato alla pressione nella modalità invalida a partire dalla modalità di *dafault* di validità e viceversa.

4.2.3 Effetti sulle pressioni delle *failure* del trasduttore d'angolo

Una caratteristica della sonda multi-funzione esaminata è che eventuali avarie che riguardano il trasduttore d'angolo, vedi § 4.2.1, influenzano inevitabilmente le misure di pressione. Il contrario non è vero come si può facilmente intuire dal funzionamento della sonda.

Per tenere conto di come variano le misurazioni di pressioni con le possibili avarie nel trasduttore d'angolo occorre conoscere qual è l'errore di pressione che la sonda commette quando non è allineata al flusso locale.

In altre parole occorre conoscere:

$$\begin{cases} \Delta P_{front} = f(\Delta\lambda, Mach) \\ \Delta P_{slot2} = f(\Delta\lambda, Mach) \end{cases} \tag{4.3}$$

dove

$$\begin{cases} \Delta P_{front} = (P_{front\_\Delta\lambda} - P_{front\_\Delta\lambda=0}) \\ \Delta P_{slot2} = (P_{slot2\_\Delta\lambda} - P_{slot2\_\Delta\lambda=0}) \end{cases} \tag{4.4}$$

in funzione dell'errore di allineamento $\Delta\lambda$:

$$\Delta\lambda = \lambda_{no\_failures} - \lambda_{failures}\,. \tag{4.5}$$

Sono disponibili delle prove sperimentali [6] che forniscono ai valori del numero di Mach pari a 0.525, 0.744, 0.908, 0.964 l'errore percentuale sulla P_{front} e sulla P_{slot2} (4.5) con errori di allineamento $\Delta\lambda$ fino a 20^o.

$$\begin{cases} \%\Delta P_{front} = \dfrac{(P_{front\_\Delta\lambda} - P_{front\_\Delta\lambda=0})}{P_{front\_\Delta\lambda=0}} * 100 \\ \%\Delta P_{slot2} = \dfrac{(P_{slot2\_\Delta\lambda} - P_{slot2\_\Delta\lambda=0})}{P_{slot2\_\Delta\lambda=0}} * 100 \end{cases} \tag{4.5}$$

Gli errori di allineamento possono essere chiaramente superiori ai 20^o e si è sopperito alla mancanza di dati assumendo che i valori delle pressioni misurate dalle sonde corrispondenti a $\Delta\lambda$ pari 20^o si mantengano costanti fino ad un massimo errore di allineamento di 40^o.

Le citate prove sono state effettuate solo per le sonde 1 e 3. Considerando che la sonda 1 e 3 sono posizionate simmetricamente rispettivamente alla 2 e alla 4 come si vede in fig. 1.3, si è assunto che il comportamento di queste ultime sia analogo a quello delle simmetriche.

I dati sperimentali sono stati forniti sotto forma di grafici di:

$$\begin{cases} \%\Delta P_{front} = f(\Delta\lambda, Mach) \\ \%\Delta P_{front} = f(\Delta\lambda, Mach) \end{cases} \tag{4.6}$$

per valori di $\Delta\lambda$ sempre positivi. Le (4.6) sono state applicate anche a $\Delta\lambda$ negativi considerando che gli effetti dei $\Delta\lambda$ negativi siano gli stessi di quelli positivi. Un comportamento del genere è logico aspettarselo data la simmetria assiale delle sonde, per cui un disallineamento positivo ha la medesima influenza sulla misura di pressione di uno negativo della stessa entità.

Questi dati sperimentali sono stati interpolati , (figg. 4.2 e 4.3) e messi sotto forma di matrici in modo da poter essere implementati nel *software* di simulazione realizzato in ambiente *Matlab-Simulink®*.

Non essendoci i valori sperimentali, per numeri di Mach inferiori a 0.525, si è realizzato una estrapolazione lineare dei dati disponibili.

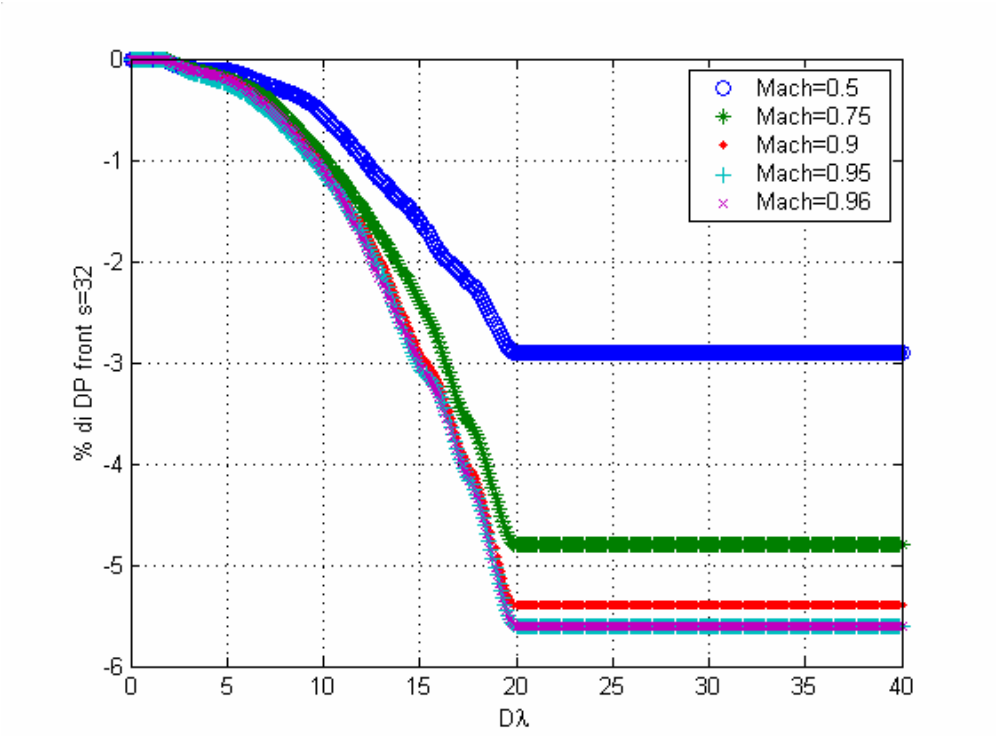

Fig. 4.2 : $\%\Delta P_{front} = f(\Delta\lambda)$ **sonde 2 e 3**

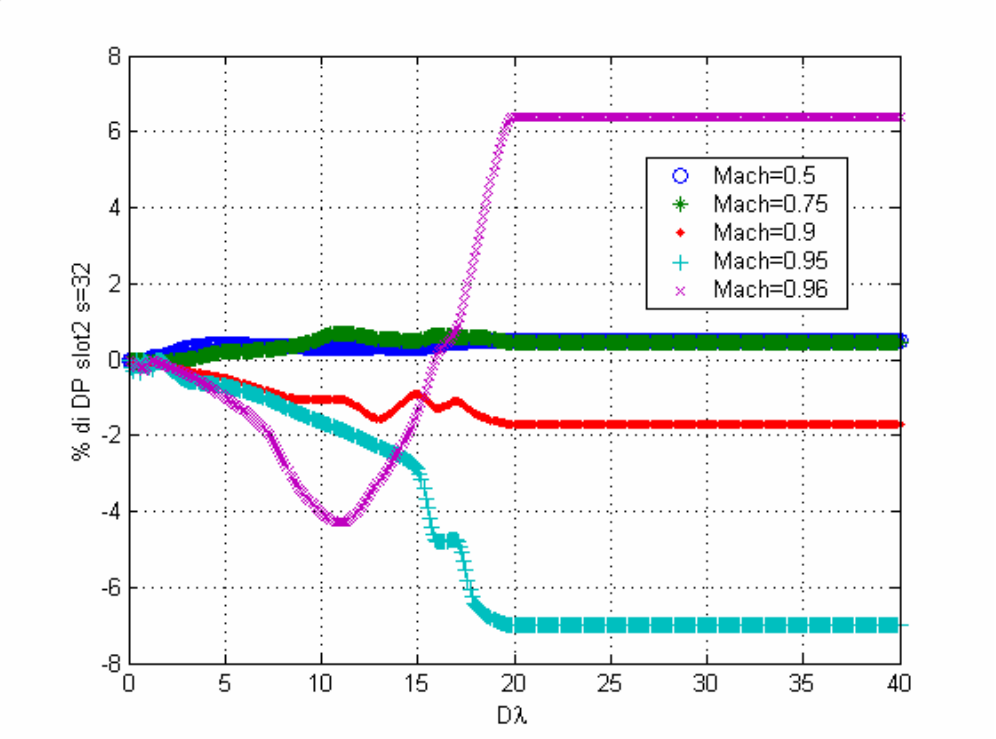

Fig. 4.3 : $\%\Delta P_{slot2} = f(\Delta\lambda)$ **sonde 2 e 3**

4.3 Dinamica *Noseboom*

La dinamica del *Noseboom* è stata approssimata con un modello che introduce solo un ritardo associato agli effetti dinamici.

Si è dunque schematizzato la dinamica del *Noseboom* con un'equazione differenziale del primo ordine:

$$\dot{\alpha}_{dyn} + \frac{1}{\tau}\cdot\alpha_{dyn} = \frac{1}{\tau}\cdot\alpha$$
$$\dot{\beta}_{dyn} + \frac{1}{\tau}\cdot\beta_{dyn} = \frac{1}{\tau}\cdot\beta \qquad (4.7)$$
$$\tau = 0.025 \quad (\text{sec})$$

4.4 Failure Noseboom

Per quanto riguarda le possibili *failure*, non conoscendo in dettaglio lo strumento adoperato effettivamente dal velivolo, si è pensato di implementare nel modello di simulazione le medesime avarie introdotte per il trasduttore di posizione angolare, ipotizzando che potessero essere sufficienti per rappresentare un comportamento anomalo dello strumento.

Si è ipotizzato inoltre, che la tipologia di avaria sia per α che per β sia la stessa.

Le *failure* implementate nel modello per α e β sono le seguenti:

1) *Accuracy*; inserimento di un rumore di intensità variabile sul segnale della posizione angolare .

2) *Offset*; alterazione della posizione angolare di una quantità stabilita al momento dell'inserimento dell'avaria;

3) *Hysteresis;* introduzione di un comportamento di isteresi della posizione angolare;

4) *Departure;* raggiungimento rapido di fondo corsa e bloccaggio a tale valore della posizione angolare;

5) *Lock*; bloccaggio della posizione angolare al valore assunto al momento dell'inserimento dell'avaria;

6) *End-stroke limits*; modifica dei limiti di fondo corsa della posizione angolare;

7) *Delay*; modifica della ritardo del modello del primo ordine che approssima il trasduttore d'angolo.

8) *Validity;* modifica del segnale di validità associato al segnale della misura di posizione angolare.

In modo analogo a quanto fatto per il trasduttore d'angolo il modello prevede che le *failure* possano essere attivate più di una in sequenza, simulando tutte le possibili combinazioni di avarie fisicamente possibili.

5. Descrizione del *software* di simulazione

Nel presente capitolo si fornisce una descrizione del *software* realizzato in ambiente *Matlab®*, *Matlab-Simulink®* e *LabVIEW®* relativo alla procedura di simulazione del sistema Dati Aria.

In particolare viene riportata una descrizione dei *file* con estensione ".m" che hanno permesso di ampliare la banca dati aerodinamica relativa alla simulazione e la descrizione dei *file* con estensione "*.mdl*", *ADS_mode_RT.mdl* e *ADS_mode_TESTS.mdl,* che sono due versioni diverse dello stesso modello di simulazione dei Dati Aria. La prima versione è stata usata per far girare la simulazione in tempo reale; la seconda ha permesso di effettuare vari tipi di test di simulazione con relativo salvataggio dei risultati. Entrambi i modelli contengono gli stessi algoritmi discussi nel Capitolo 2. Di seguito vengono descritti in particolare i blocchi *Sensors Dinamics e Noseboom Dinamics* che sono relativi all'attività svolta nella presente tesi.

Inoltre viene riportata una descrizione del *file Control_Pannel.vi* che rappresenta il pannello di controllo, realizzato nell'ambito di questa tesi, per l'inserimento delle avarie e la visualizzazione delle uscite del modello di simulazione predisposto per il *Real Time* rappresentato dal *file ADS_mode_RT.mdl.*

5.1 File relativi all'ampliamento dell'inviluppo di volo

5.1.1 DATI_Galleria_Array.m

Questo file, riportato in Appendice A, utilizza i dati di galleria di λ_i, Cp_{Li} e M_{Li} relativi alla configurazione di crociera che in un lavoro precedente a questa tesi sono stati oggetto di un controllo e sono stati disposti sotto forma di matrici meglio gestibili rispetto ai dati originali.

Il file limita le curve dei dati sperimentali nell'intervallo di α compreso tra -10° e +25° e di β compreso tra -20° e 20°, come precedentemente detto al Capitolo 3, ed in seguito carica questi sotto forma di *Array-Strutturato.*

5.1.2 LUT_V_Generator.m

Il file, riportato in Appendice A, realizza la procedura che permette di ottenere gli andamenti delle grandezze di simulazione (λ_i, Cp_{Li} e M_{Li}) ad α costante e β variabile per Mach superiori a 0.4.

Il *database* aerodinamico non contiene queste informazioni e quindi è stata realizzata una procedura che a partire dai dati disponibili riesce a stimare le curve prima citate. Il modello sviluppato si basa su una procedura iterativa che calcola le differenze tra le curve ad α costante (per α pari a 0°, 10°, 15°, 20°, 25°) e Mach pari a 0.4 e le curve ad α variabile e Mach superiori di 0.4 nei punti corrispondenti agli stessi valori di α. Successivamente le curve citate per prime vengono traslate di una quantità pari alla differenza media e vengono nuovamente calcolate le differenze. Il processo descritto viene ripetuto fino a quando la differenza non è inferiore di un fissato valore . Si osserva che sono sufficienti due iterazioni per avere la convergenza della procedura iterativa .

Alla fine del procedimento iterativo i nuovi dati vengono salvati per la successiva approssimazione ai Minimi Quadrati.

5.1.3 LUT_MinQuad.m

Il *file LUT_MinQuad,* riportato in Appendice A effettua l'approssimazione ai Minimi Quadrati sui dati derivanti dai *file DATI_Galleria_Array.m* e *LUT_V_Generator.m* ed infine genera le *look-up table* necessarie al modello di simulazione, ovvero ai *file ADS_model_RT.mdl* e *ADS_model_TEST.mdl*.

5.2 Modelli di simulazione

Come già accennato sono state realizzate due versioni del modello di simulazione, una compatibile con l'ambiente di lavoro *Matlab-Simulink-xPC Target* ® per le simulazioni in tempo reale (*Real Time)*, l'altra utilizzata per effettuare vari tipi prove di simulazione. Queste prove hanno permesso di confrontare le uscite di simulazione con i dati di galleria a disposizione e quindi validare la procedura di simulazione.

La presenza di due versioni si è resa necessaria soprattutto perché la compilazione del modello di simulazione da far girare in tempo reale richiede un tempo abbastanza lungo stimabile in 30-40 minuti, quindi si è ritenuto di non appesantire troppo il modello per la simulazione in *Real Time* con funzioni non strettamente necessarie alla simulazione dei Dati Aria, ma utili per le visualizzazioni e il salvataggio dei dati. Il modello per il *Real Time* è stato affiancato inoltre da un pannello di visualizzazione e di inserimento delle avarie realizzato in ambiente *LabVIEW*® per poter effettuare prove di simulazione con la possibilità di inserimento delle avarie in modo interattivo.

5.2.1 ADS_input.m

Questo *file* contiene le caratteristiche geometriche e dinamiche del sistema Dati Aria. In particolare sono presenti i seguenti dati:

- la posizione delle sonde sul velivolo in assi corpo;
- la costante di tempo dei trasduttori di pressione;
- la pulsazione ed il coefficiente di smorzamento della risposta dinamica di rotazione della sonda;
- la posizione delle alette del *Noseboom* in assi corpo;
- la costante di tempo della risposta dinamica del *Noseboom*;
- i dati relativi all' "Aria Standard";
- i dati aerodinamici di galleria e approssimati in forma matriciale;
- i dati relativi all'errore di pressione in funzione dell'errore di allineamento della sonda al flusso locale.

In riferimento al penultimo punto vengono caricate le matrici tridimensionali generate dal *file LUT_MinQuad.m*, per le *look-up-table* di interpolazione relative ai λ_i, Cp_{Li}, e M_{Li} della configurazione di *Cruise*.

In riferimento all'ultimo punto il *file* calcola sotto forma di matrici le relazioni che rappresentano l'errore sulle pressioni in funzione dell'errore di allineamento della sonda al flusso locale, come precedentemente spiegato nel § 4.2.3.

5.2.2 Manovre.m

Questo *file* permette di caricare gli input di simulazione relativi alla manovra che si vuole simulare. Gli input di simulazione sono la quota, il Mach, gli angoli di incidenza e derapata, le componenti di velocità angolare del velivolo e le deflessioni delle superfici mobili del velivolo. La manovra desiderata viene scelta tra un set di manovre che sono state fornite dall'azienda aeronautica italiana con la quale il DIA collabora.

5.2.3 ADS_model_RT.mdl

La descrizione seguente si riferisce al modello predisposto per il *Real-Time*, successivamente invece, viene riportata una descrizione del modello predisposto per i test di simulazione mettendo in evidenza soprattutto le differenze poiché è sostanzialmente lo stesso modello.

Questo *file* (fig. 5.1) costituisce la parte centrale del *software* di simulazione dei Dati Aria perché contiene i modelli descritti nei Capitoli 2 e 4. Tenendo conto della complessità del programma, si è preferito dare allo stesso una struttura ad albero, che da una parte faciliti l'interfaccia utente, e dall'altra consenta di comprendere meglio la disposizione logica dei vari componenti.

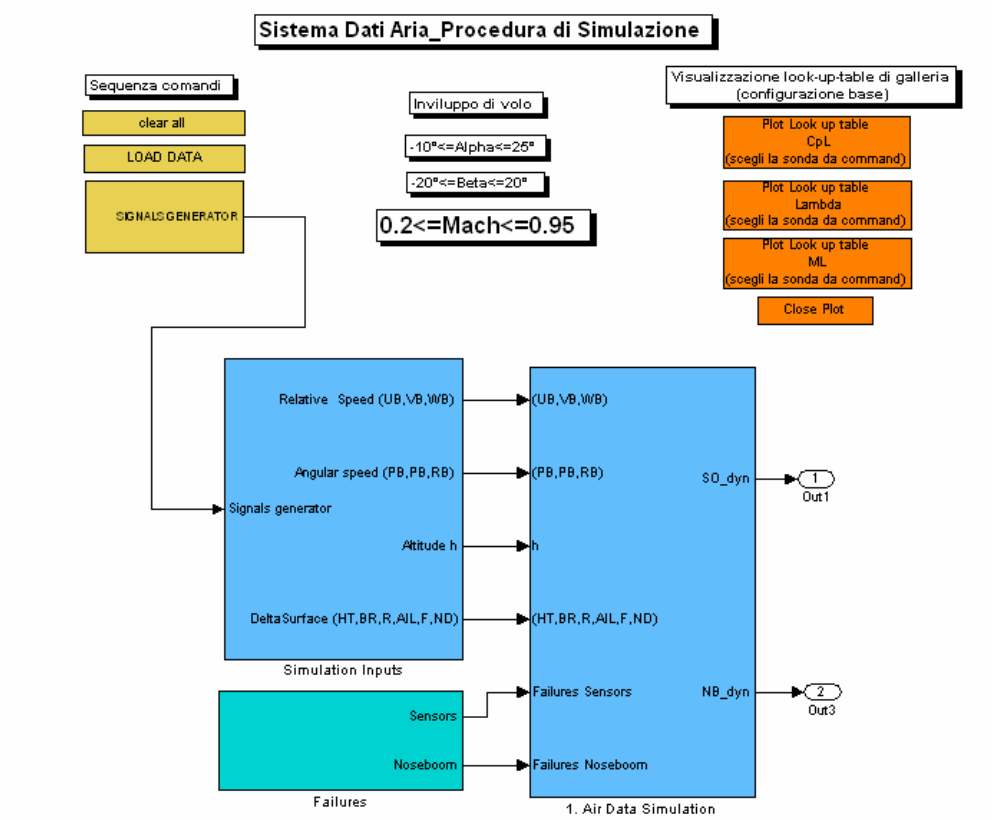

Fig. 5.1 : ADS_model_RT – level 0

Dalla fig. 5.1 si può osservare che al *level 0* è possibile caricare il *file ADS_input.mat* e mandare in esecuzione il *file Manovre.m* mediante il tasto "*LOAD DATA*".

Inoltre al *level 0* vi è la possibilità di attivare (mediante i tasti "*Plot look-up-table CpL*", "*Plot look-up-table lambda*", "*Plot look-up-table ML*") le *routine* con estensione ".m" *Plot_CpL_LookupTable, Plot_Lamb_LookupTable,* e *Plot_ML_LookupTable* che permettono di visualizzare i grafici inerenti i dati delle varie *look-up-table*.

In fig. 5.2 s'illustra la struttura ad albero (*model tree*) del *software* realizzato:

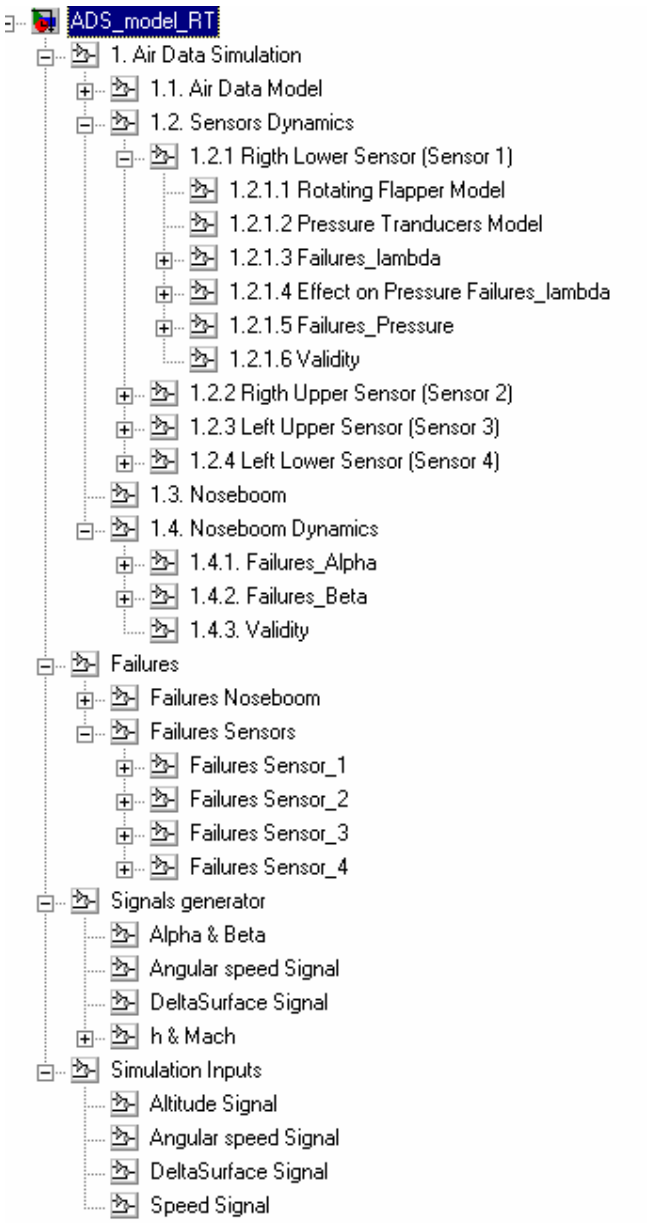

Fig. 5.2 : Model tree del software Adm_model_RT.mdl

Dal *model tree* si può osservare che il *software* realizzato è suddiviso in blocchi interagenti tra loro ed organizzati per livelli. Al *level 0* (Fig. 5.2) sono presenti quattro blocchi: il blocco *Signals Generator* (fig . 5.3) che genera i segnali di *inputs* (possono essere costanti, sinusoidali, a rampa) o carica gli *input* dai dati relativi ad una manovra. All'interno di tale blocco sono presenti i blocchi chiamati: *h & Mach, Alpha & Beta, Angular speed signal e Delta Surface signal*, i primi due sono riportati in figg. 5.4 e 5.5 . Il blocco *Simulation Inputs* ordina e ridefinisce i segnali di ingresso alla procedura di simulazione, il blocco *Failures* (fig. 5.6) riceve e gestisce le avarie inviate dal pannello di controllo realizzato in *LabVIEW®* e il blocco *1. Air Data Simulation* contiene all'interno i vari modelli analitici e dinamici del sistema Dati Aria ed invia le uscite di simulazione al pannello di controllo.

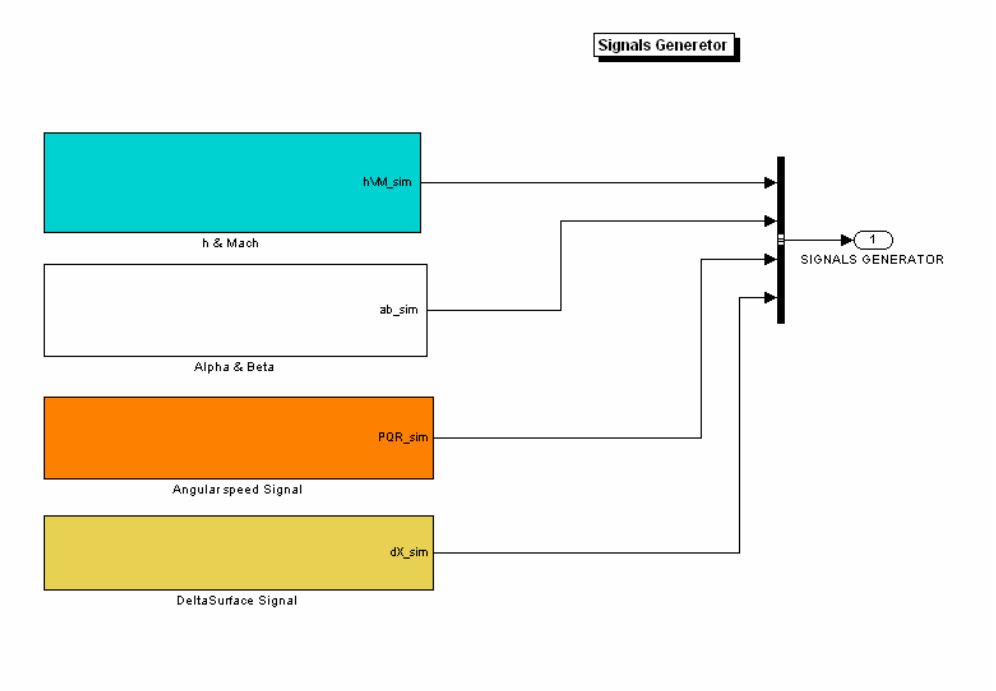

Fig. 5.3 : blocco Signals Generator

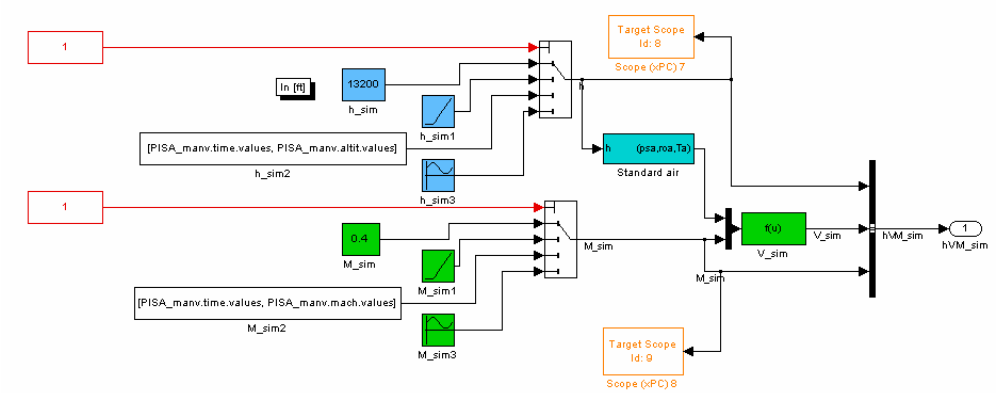

Fig. 5.4 : blocco h e Mach

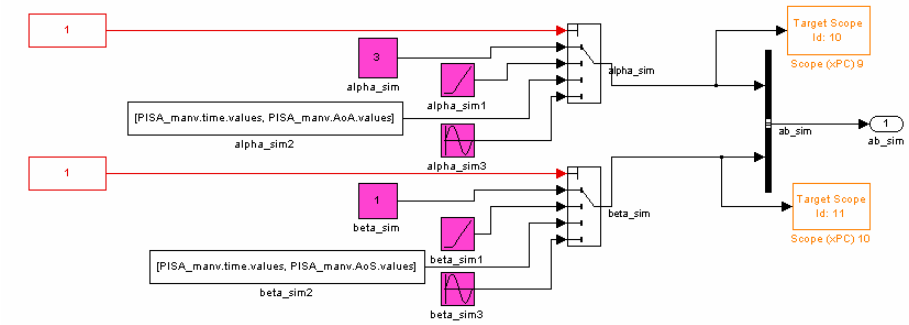

Fig. 5.5 : blocco Alpha e Beta

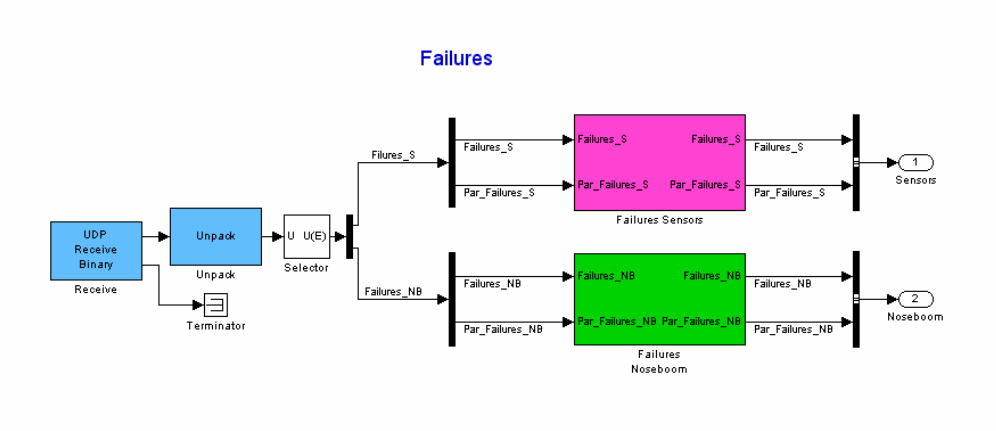

Fig. 5.6 : Blocco Failures

In fig. 5.6 si osserva la presenza del blocchetto *UDP Receive Binary* (fig. 5.7) che è quel blocco che permette al modello *Matlab-Simulink®* di ricevere i segnali relativi alle avarie trasmessi dal pannello di controllo realizzato in ambiente *LabVIEW®*.

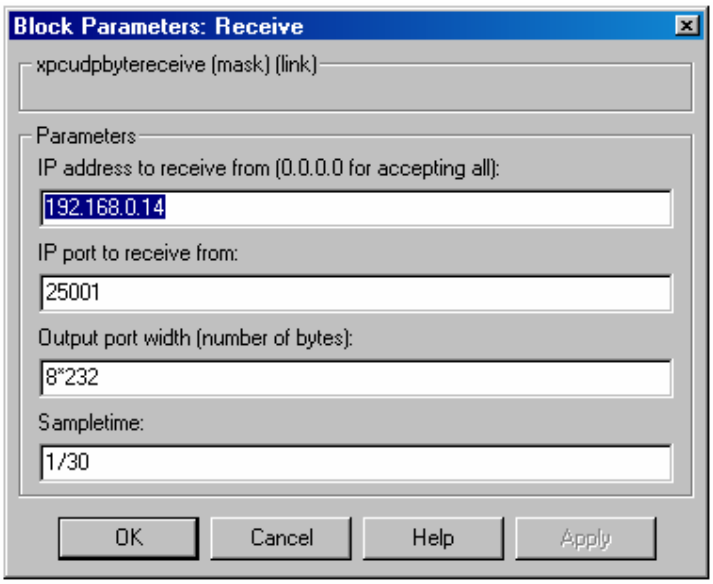

Fig. 5.7 : *UDP Receive Binary*

La fig. 5.7 presenta la maschera del blocchetto *UDP Receive Binary* nella quale si possono settare i suoi parametri. La prima voce indica l'indirizzo del computer in rete locale a cui devono essere inviati i segnali di avaria, in questo caso l'indirizzo del computer (*Target)* su cui si fa girare il modello in tempo reale utilizzando l'ambiente di lavoro *Matlab-Simulink-xPC Target ®. L*a seconda voce indica una porta IP libera del *Target* che riceve i segnali, la terza voce indica il numero di bit moltiplicato per il numero dei segnali di avaria che il *Target* riceve ed in fine l'ultima voce indica il tempo di campionamento dei segnali ricevuti.

Ritornando alla descrizione della fig. 5.1, il blocco 1 è suddiviso in quattro blocchi (fig. 5.8): il blocco *1.1 Air Data Model* contenente i modelli analitici del sistema Dati Aria relativi alle procedure di simulazione, il blocco *1.2 Sensors Dinamics* contenente i modelli dinamici delle sonde e i modelli realizzati per la generazione di *failure* nelle sonde,il blocco *1.3 Noseboom* contenente la procedura per il calcolo dell'angolo di incidenza e di derapata misurati dal *Noseboom* ed infine il blocco *1.4 Noseboom Dinamics* che contiene il modello dinamico del *Noseboom* e i modelli per la generazione di *failure,* in modo analogo a quanto fatto nel blocco 1.2.

I primi due blocchi sono strutturati al loro interno in modo da rendere visibile la suddivisione tra le quattro sonde (figg. 5.9, 5.10). Negli ultimi due blocchi invece è visibile la suddivisione tra l'angolo di incidenza e l'angolo di derapata (figg. 5.11, 5.12).

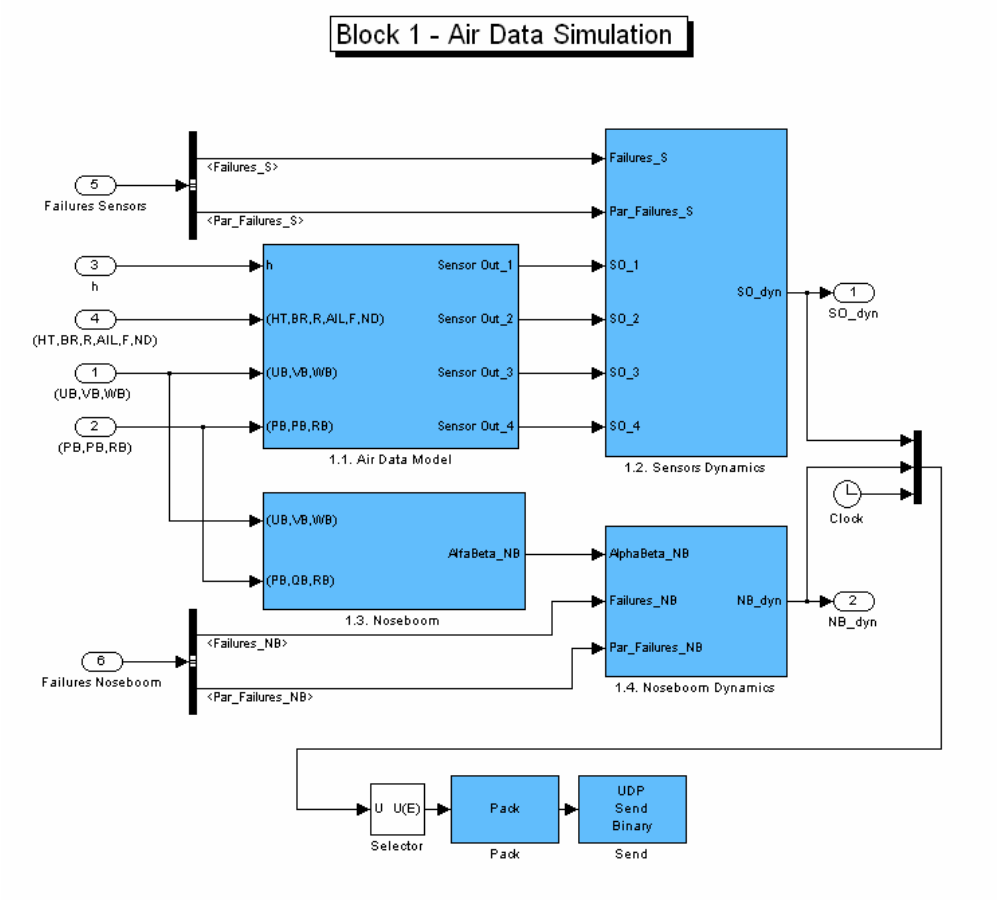

Fig. 5.8 : Blocco 1 "Air Data Simulation"

In fig. 5.8 si osserva il blocchetto *UDP Send Binary* (fig. 5.9) che permette al modello *Matlab-Simulink®* di inviare i segnali di uscita al pannello di controllo realizzato in ambiente *LabVIEW®*.

In fig. 5.9 è illustrata la maschera del blocchetto prima citato. La prima voce indica l'indirizzo del computer in rete locale a cui devono essere inviate le uscite di simulazione, in questo caso l'indirizzo del computer su cui sta girando il panello di visualizzazione e di inserimento delle avarie realizzato in *LabVIEW®*. La seconda voce indica una porta IP libera del *Target* per inviare le uscite di

simulazione al pannello *LabVIEW®*, quindi deve avere un numero diverso da quella di ricezione. La terza voce indica il tempo di campionamento dei segnali da inviare.

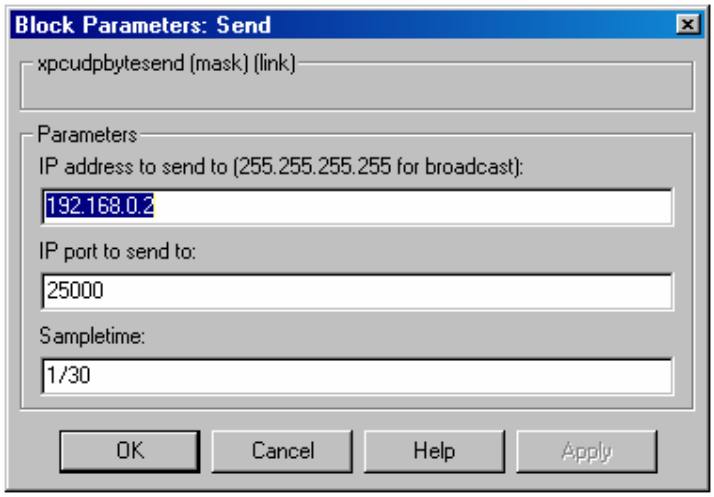

Fig. 5.9 : ***UDP Send Binary***

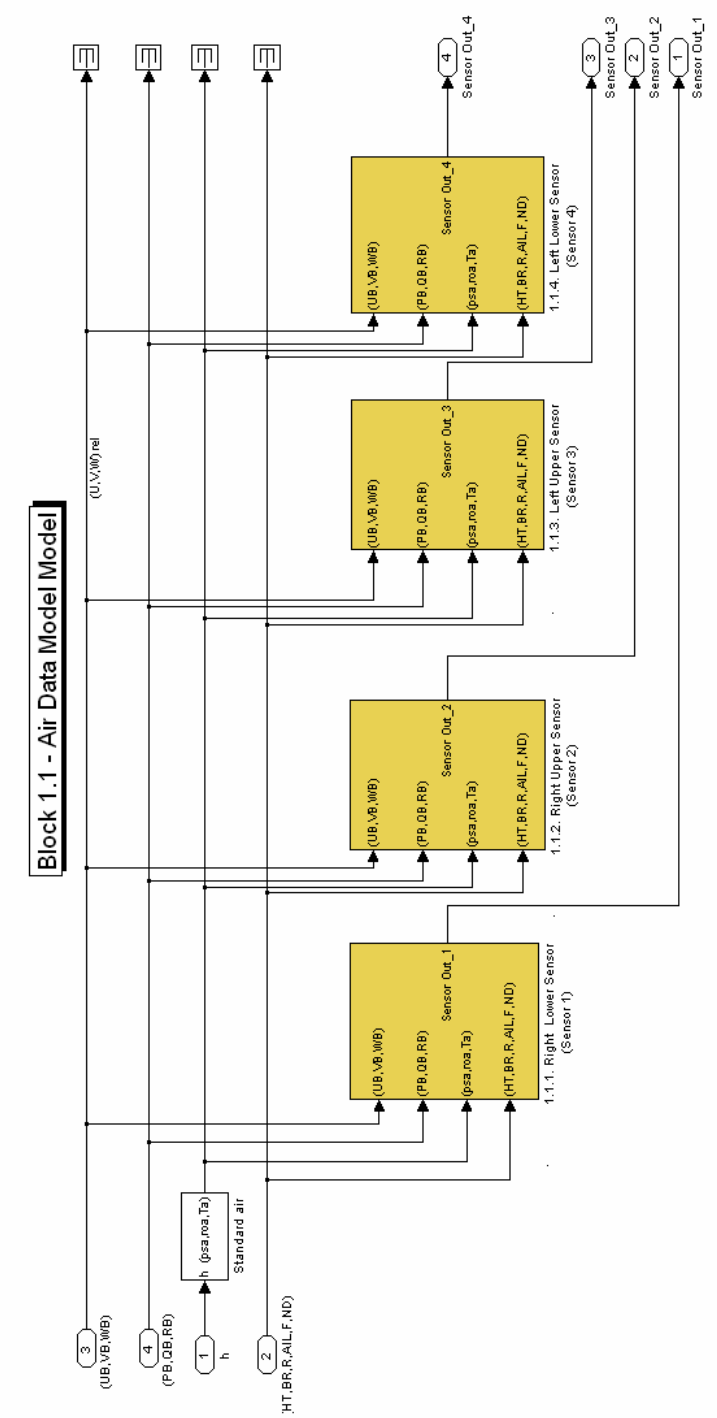

Fig. 5.10 : Blocco 1.1 Air Data Model

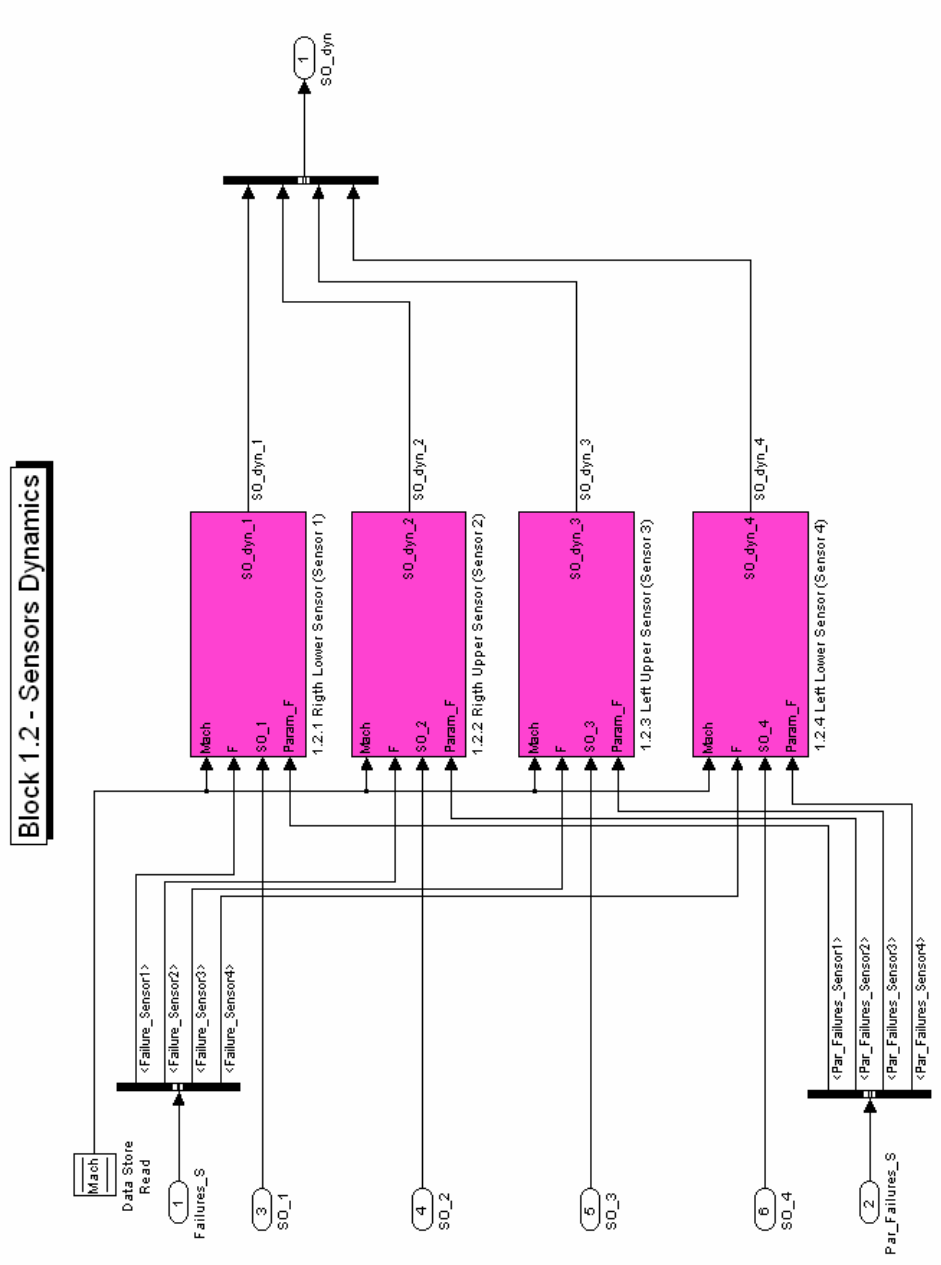

Fig. 5.11 : Blocco 1.2 Sensors Dinamics

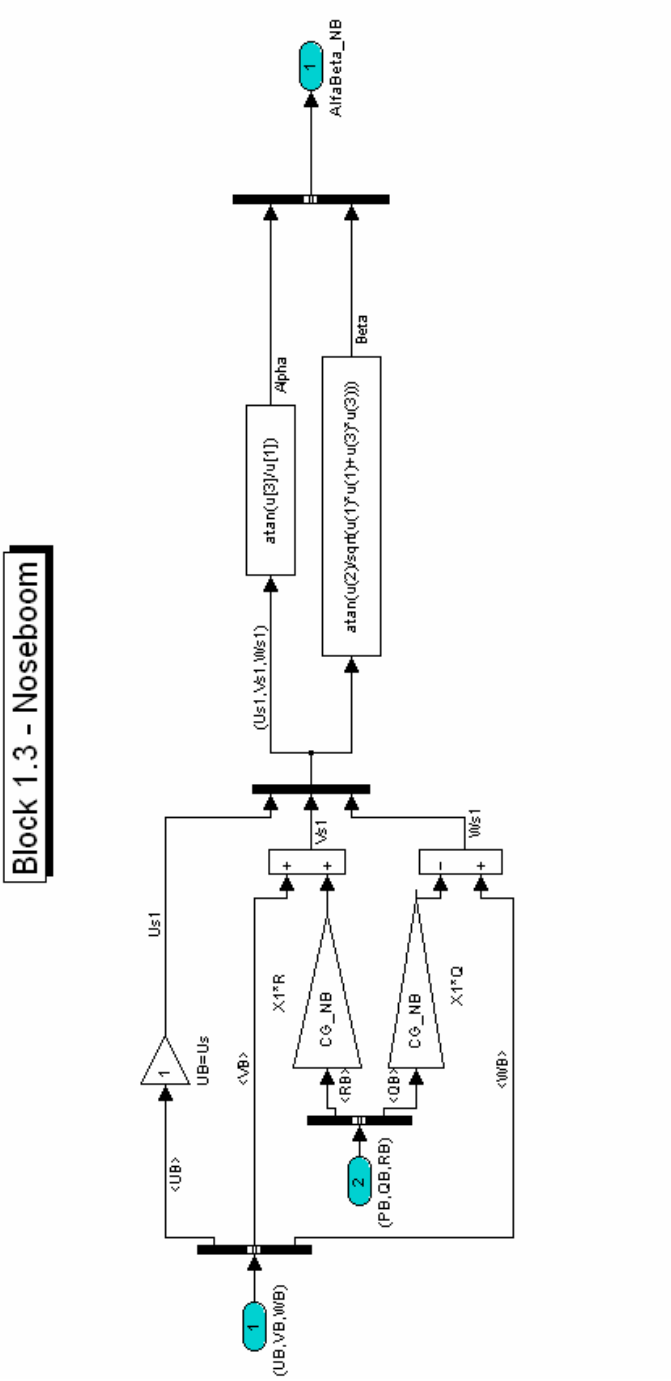

Fig. 5.12 : Blocco 1.3 Noseboom

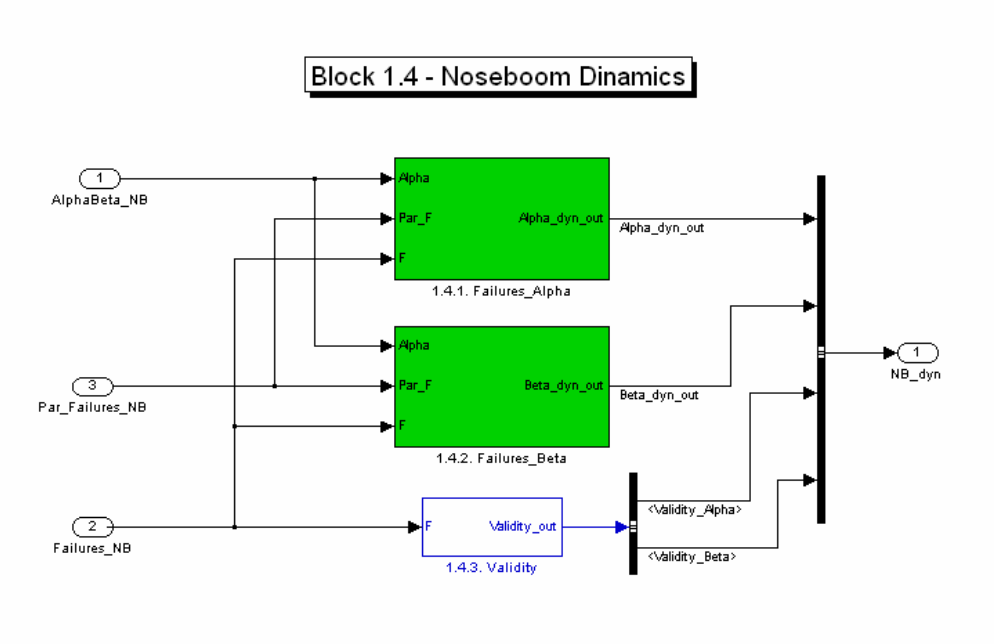

Fig. 5.13 : Blocco 1.4 Noseboom Dinamics

Si riportano di seguito, a titolo d'esempio, solo lo schema del blocco 1.1.1 (fig. 5.14) relativo alla sonda 1 (*Right Lower Sensor*), in quanto la sua struttura è analoga a quella dei blocchi 1.1.2, 1.1.3, 1.1.4 relativi alle rimanenti sonde. Il blocco 1.1.1 è stato strutturato in modo da essere fedele allo schema concettuale di fig. 2.2, fatta eccezione per il blocco 1 che, essendo uguale per tutte le sonde, è stato posto al livello superiore (blocco *Standard air* in fig. 5.10) e fatta eccezione per il blocco 9 (*Sensors Dinamics*) in fig. 2.2 che si trova nel blocco 1.2 (fig. 5.11).

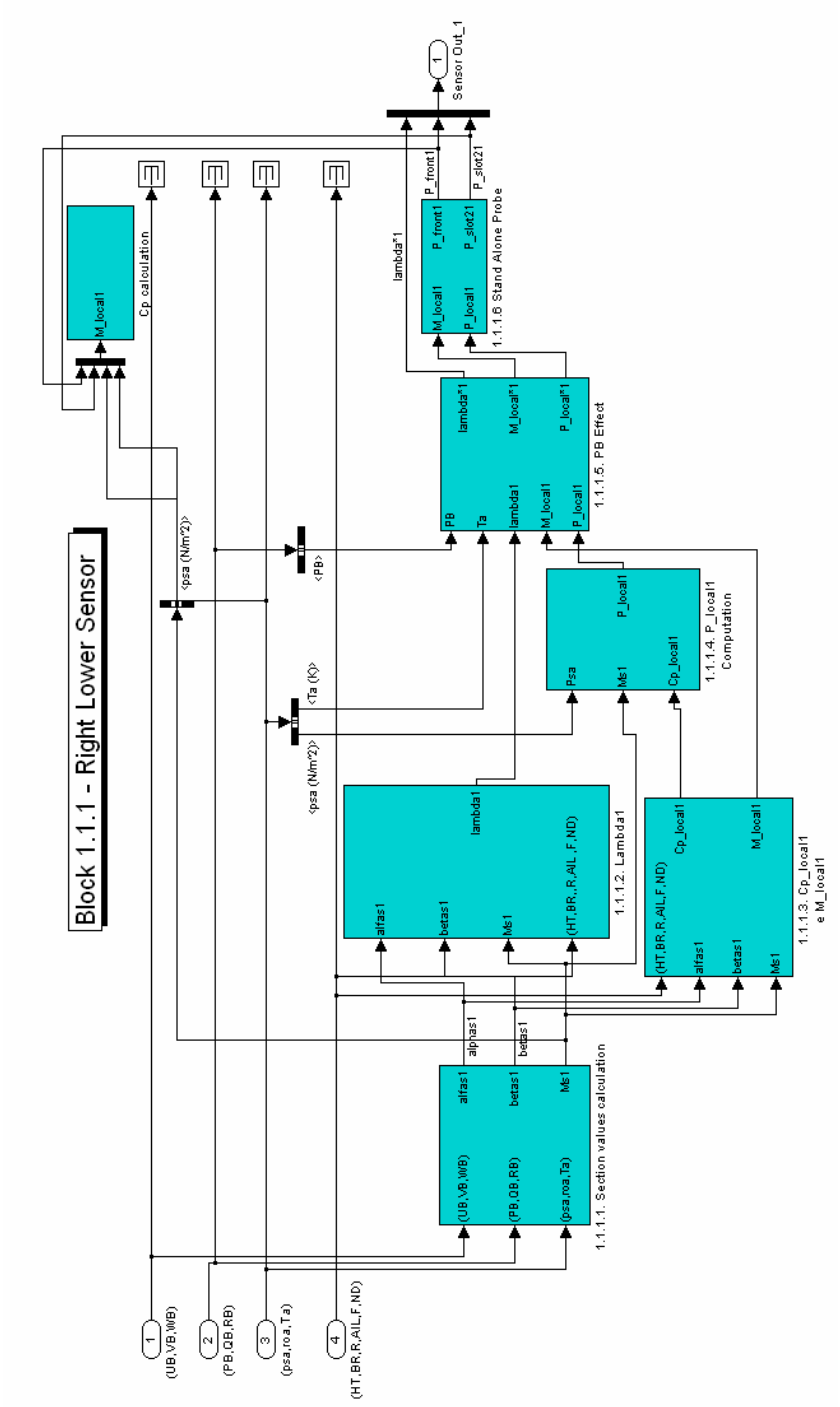

Fig. 5.14 : Blocco 1.1.1 "Right Lower Sensor"

Dato che la presente tesi si è concentrata in particolare sulla parte relativa alla dinamica dei sensori e all'implementazione delle *failure,* vengono riportati di seguito gli schemi a blocchi e le descrizioni relative a questa parte.

In fig. 5.15 è riportato soltanto lo schema del blocco (1.2.1) che descrive la dinamica della sonda 1 (*Right Lower Sensor*), in quanto la sua struttura è analoga a quella dei blocchi 1.2.2, 1.2.3, 1.2.4 relativi alle rimanenti sonde.

In fig. 5.15 si trovano i blocchi relativi alla dinamica della sonda, suddivisa in quella del trasduttore di posizione angolare e quella del trasduttore di pressione (blocchi 1.2.1.1 e 1.2.1.2) come descritto in § 4.1. Si trovano inoltre i blocchi 1.2.1.3, 1.2.1.5 e 1.2.1.6 (figg. 5.16, 5.17, 5.18) relativi rispettivamente, alla realizzazione delle *failure* sulla misura della posizione angolare, delle *failure* sulle misure di pressione e delle *failure* di *validity* analogamente a quanto descritto in § 4.2.1 e § 4.2.2. Il blocco 1.2.1.4 (figg. 5.19) invece, contiene i modelli che permettono di simulare le variazioni delle misure di pressione a seguito di *failure* sul trasduttore d'angolo, come descritto in § 4.2.3.

Secondo lo schema di fig. 5.15 al segnale di ingresso λ viene applicata una dinamica del secondo ordine nel blocco 1.2.1.1, in seguito nel blocco 1.2.1.2 può essere attivata una avaria su tale segnale. Nel blocco 1.2.1.2 si notano le *failure* precedentemente descritte in § 4.2.1, le avarie: *Frequency, Time costant, Accuracy, Offeset, Hysteresis, Departure, Lock e Endstroke limit*. Le uscite scaturite dai blocchi 1.2.1.1 e 1.2.1.2 entrano nel blocco 1.2.1.3 insieme all'ingresso λ per determinare l'errore di allineamento $\Delta\lambda$ (definito in § 4.2.3). All'interno di quest'ultimo blocco l'errore di allineamento $\Delta\lambda$ serve per determinare le variazioni di pressioni da sommare ai segnali di pressione. Dopodiché nel blocco 1.2.1.5 ai segnali di pressione vengono applicate le proprie dinamiche e possono essere soggetti alle avarie descritte in § 4.2.2: *Delay, Accuracy, Offeset, Hysteresis, Departure, Lock e Endstroke limit*.

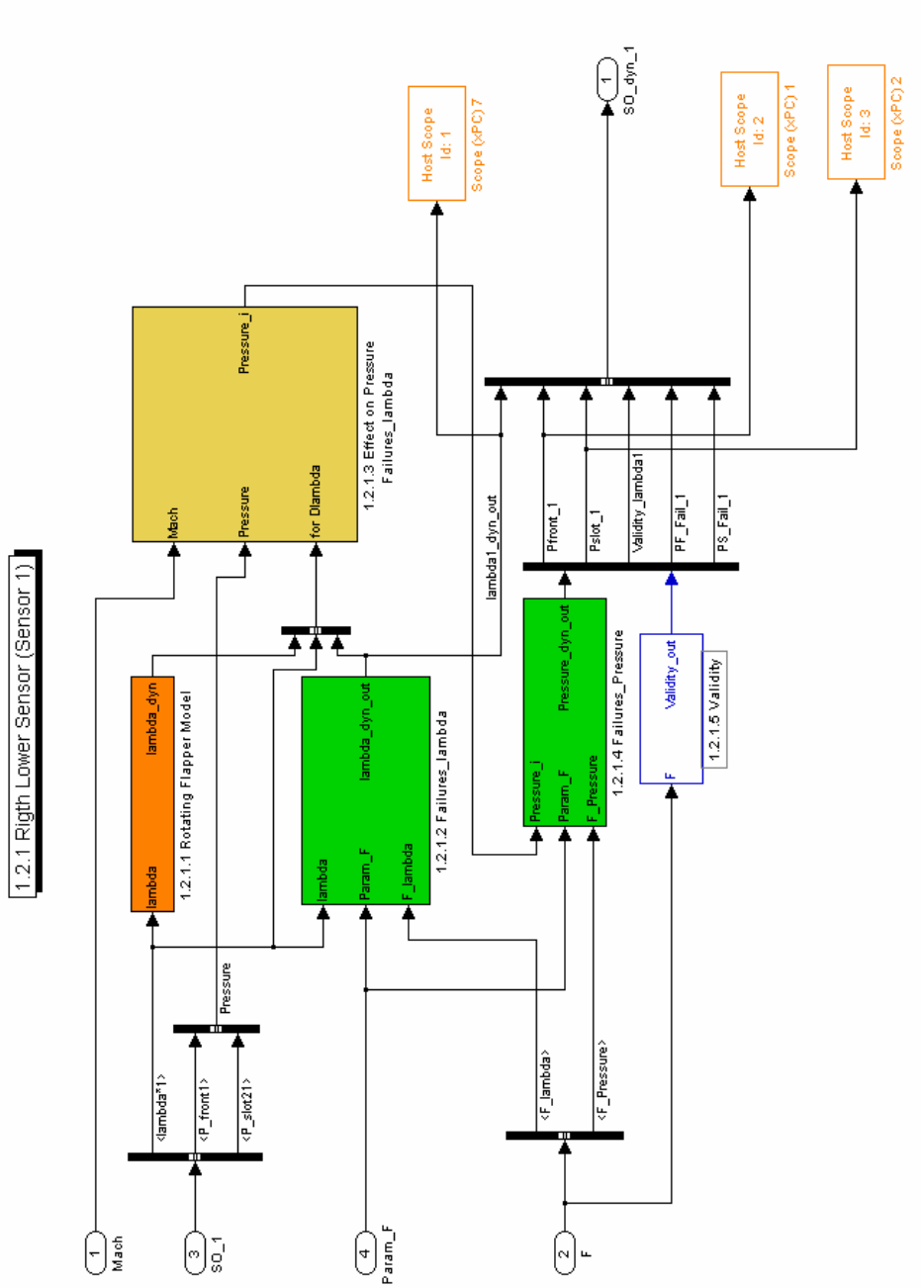

Fig. 5.15 : Blocco 1.2.1 "Right Lower Sensor"

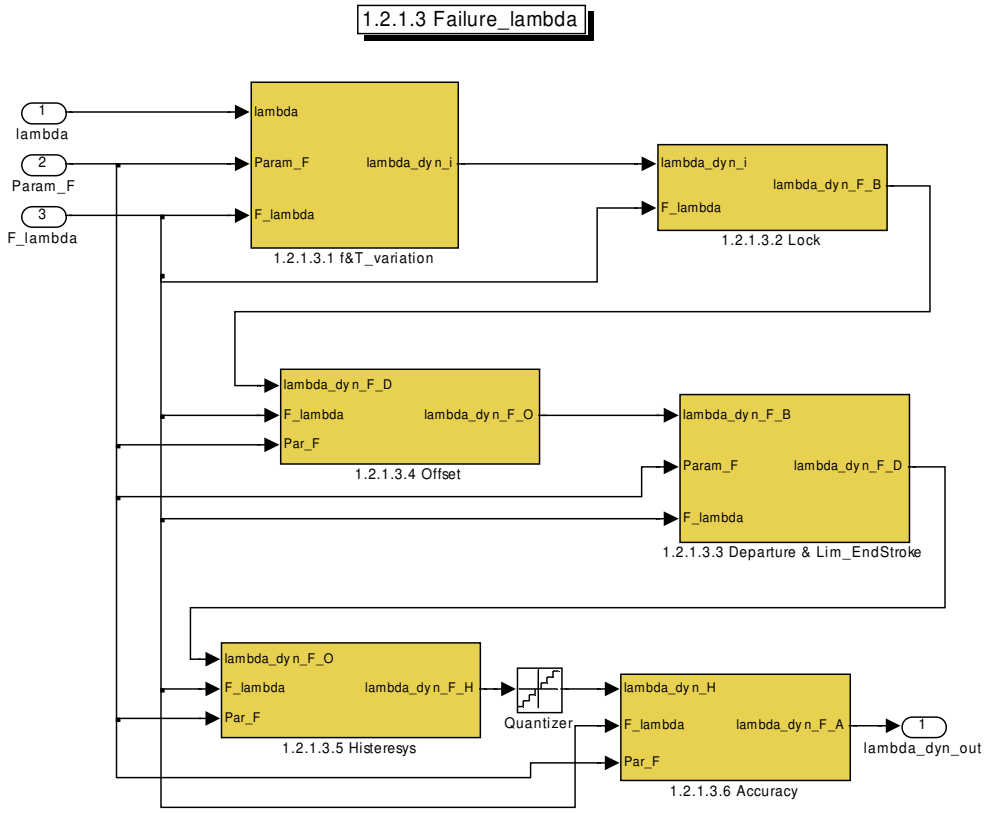

Fig. 5.16 : Blocco 1.2.1.3 Failure lambda

In fig. 5.16 si nota il percorso del segnale della posizione angolare attraverso i blocchi che rappresentano le possibili avarie a cui può essere soggetto, come indicato § 4.2.1.

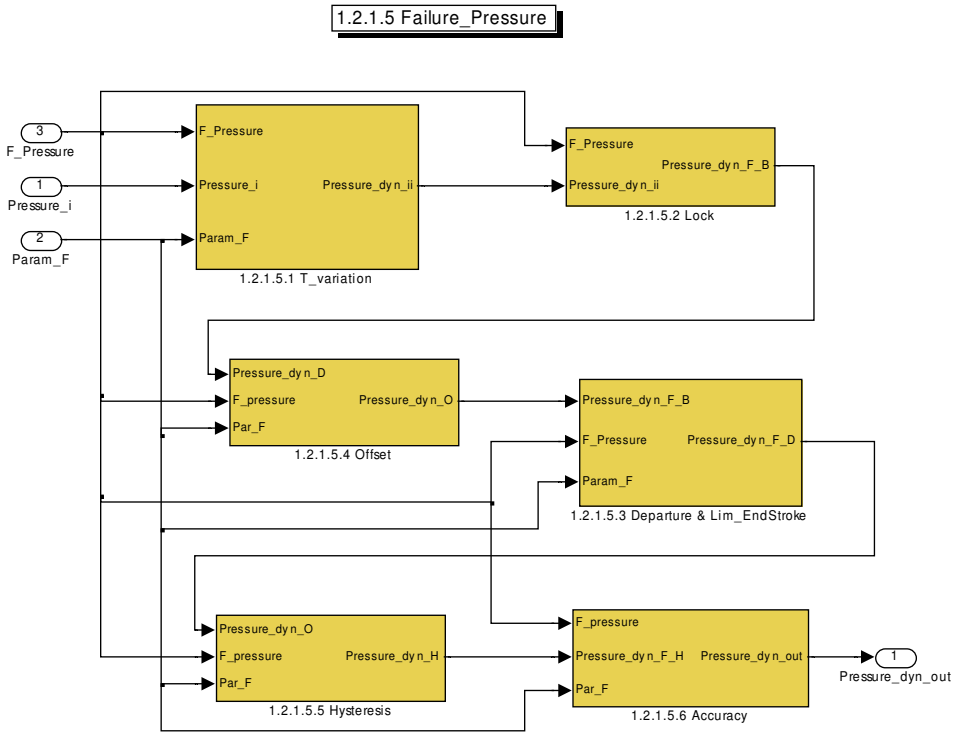

Fig. 5.17 : Blocco 1.2.1.5 Failure Pressure

In fig. 5.17,analogamente alla fig. 5.18 si nota il percorso dei segnali della misure di pressione attraverso i blocchi che rappresentano le possibili avarie a cui possono essere soggette, come indicato § 4.2.2. All'interno del blocco 1.2.1.5.1 è presente la dinamica dei trasduttori di pressione, la quale può essere variata modificando il valore del ritardo.

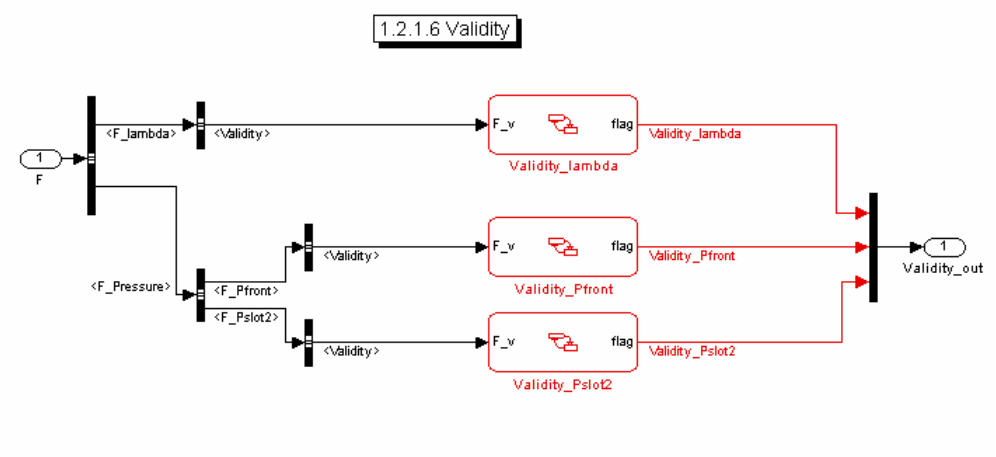

Fig. 5.18 : Blocco 1.2.1.8 Validity

In fig. 5.18 è raffigurato l'interno del blocco che permette ai segnali di *validity* associati ad ogni misura delle sonde di passare da zero ad uno.

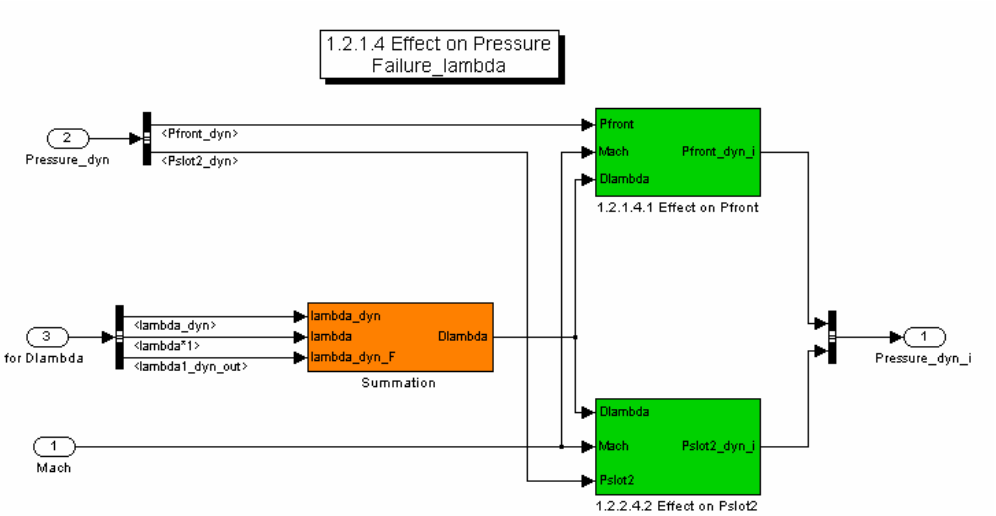

Fig. 5.19 : Blocco 1.2.1.4 "Effect on Pressare Failure lambda"

In fig. 5.19 si nota l'interno del blocco che permette di modificare i segnali di pressione a seconda dell'errore di allineamento delle sonde, $\Delta\lambda$.

5.2.4 Control_Pannel.vi

uesto *file* (fig. 5.20), realizzato con il *software LabVIEW®,* permette di gestire le *failure* e di visualizzare le uscite del modello di simulazione predisposto per girare in tempo reale rappresentato dal *file ADS_model_RT.mdl*.

Come si osserva in fig. 5.20 il pannello è costituito da cinque finestre di interfaccia (*Insert Sensors-Failures, Visualization Sensors-Outputs, Validity Sensors-Outputs, Notice e Noseboom*) che possono essere aperte cliccando con il *mouse* sul nome delle finestra desiderata.

La prima finestra, *Insert Sensors-Failures*, (fig. 5.20) rappresenta l'ambiente predisposto al settaggio dei parametri relativi alle avarie delle sonde multifunzione e all'inserimento di queste. Si fa notare che nonostante il numero elevato di pulsanti (200), questi sono stati disposti in modo tale da poter gestire l'inserimento di tutte le *failure* delle quattro sonde in una sola schermata. In basso a sinistra in fig. 5.20 si trovano i pulsanti *STOP, RESET FAILURES*, *RESET PARAMETERS* e un indicatore del tempo di simulazione del modello *Matlab-Simulink®*. Il primo pulsante serve per disattivare dalla simulazione il pannello che deve essere avviato separatamente dal modello *Matlab-Simulink®* attraverso un suo pulsante di "*run*". Il secondo pulsante serve per riportare ai valori di *default* i parametri delle avarie e il terzo per disattivare i pulsanti delle avarie.

In fig. 5.20 si nota inoltre, per esempio per la sonda 1, che tutte le avarie citate in § 4.2.1 e § 4.2.2 sono presenti e sono rappresentate dai pulsanti: *Frequency, Delay, Time costant, Accuracy, Offeset, Hysteresis, Departure +, Departure -, Lock, Up-limit, Low-limit e Validity.* Sono presenti anche dei *display* che permettono di settare i parametri relativi alle varie avarie, i quali sono settati di *default* sui rispettivi valori nominali. I *display* sopra citati riportano il nome del parametro da settare (*Frequency, Delay, Time costant, Accuracy, Offset, Deadband dell' Hysteresis, Up-limie e Low-limit*) e la sua unita di misura.

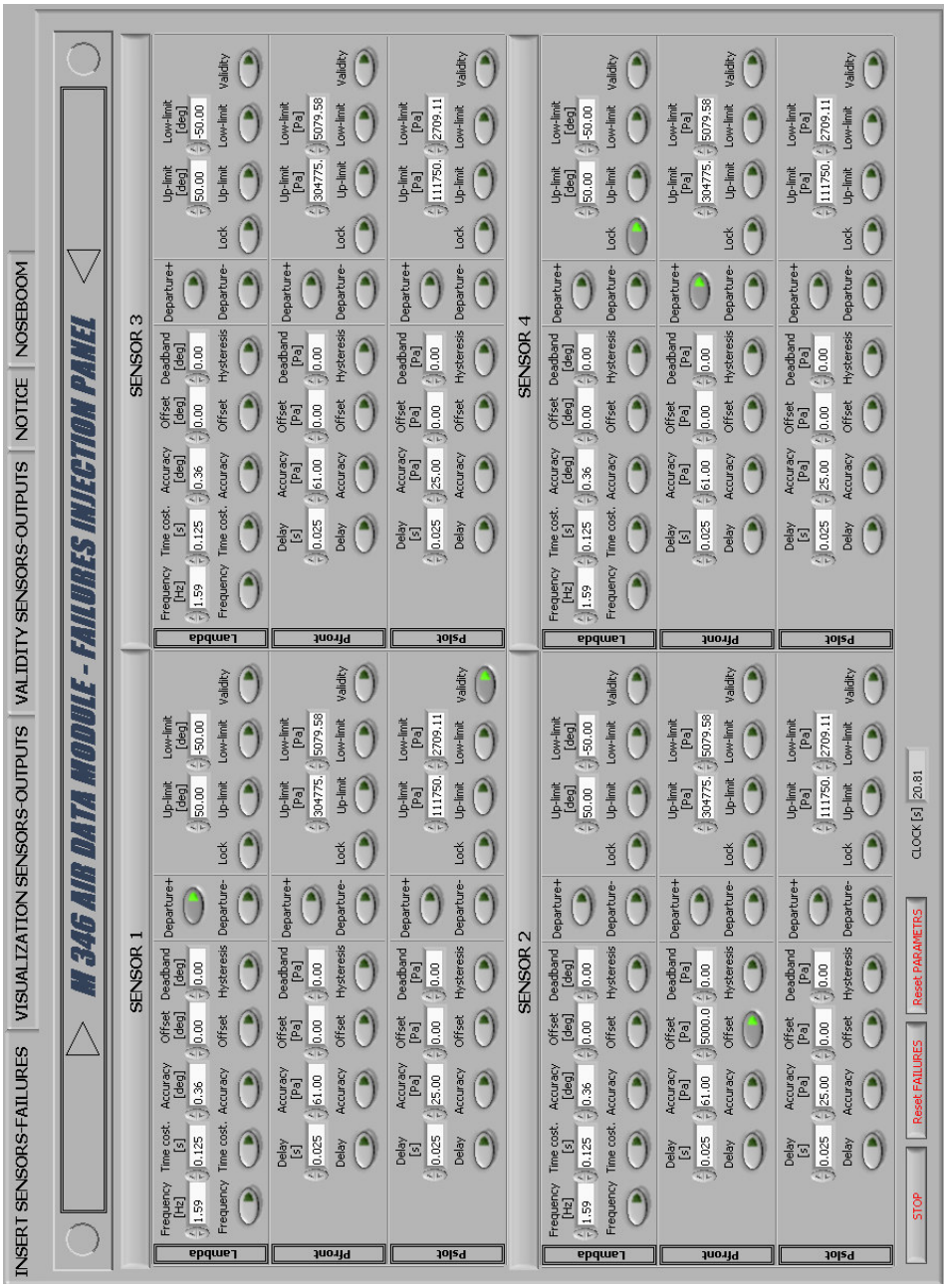

Fig. 5.20 : Control_Pannel.vi (1' finestra)

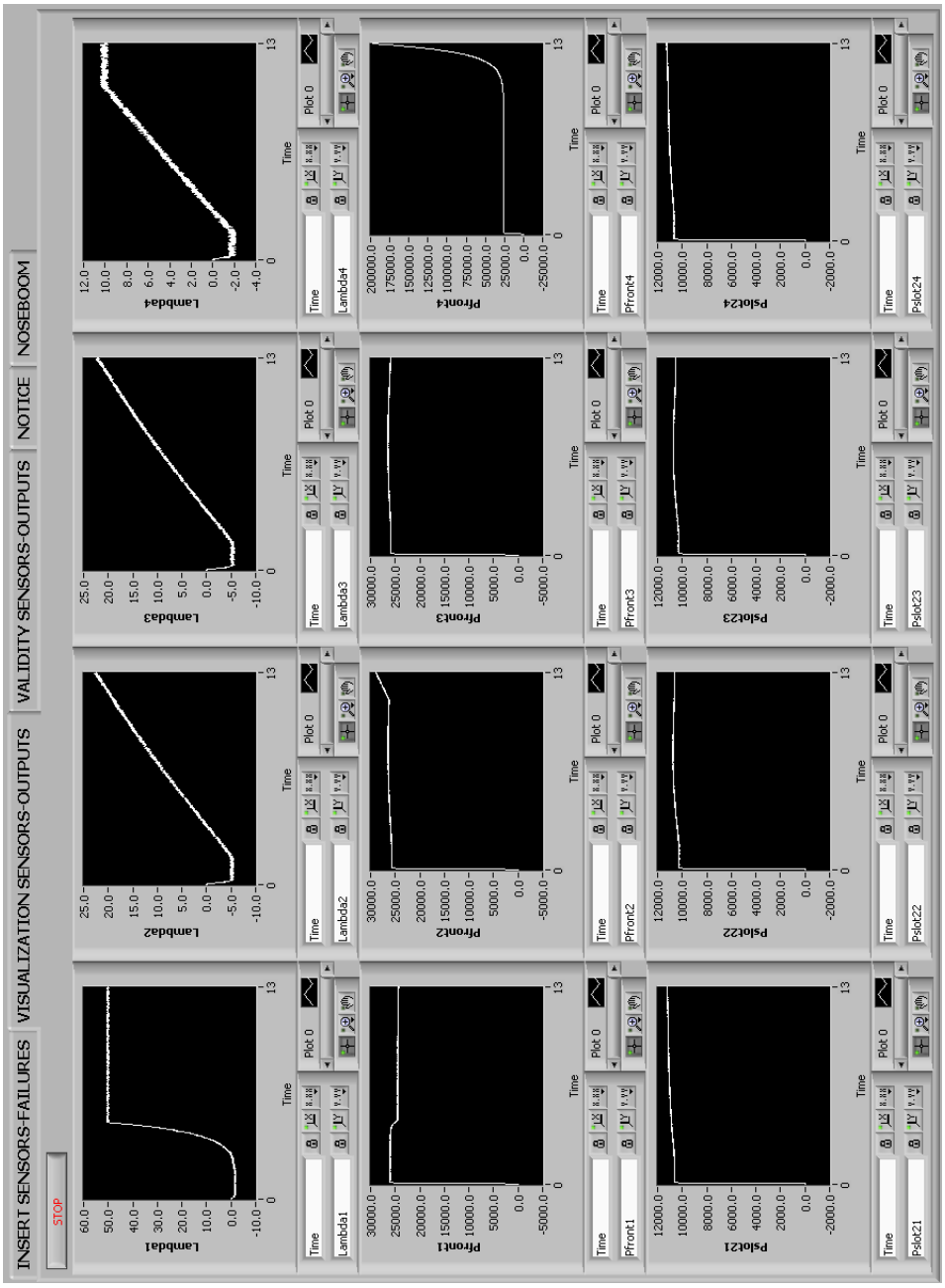

Fig. 5.21 : Control_Pannel.vi (2' finestra)

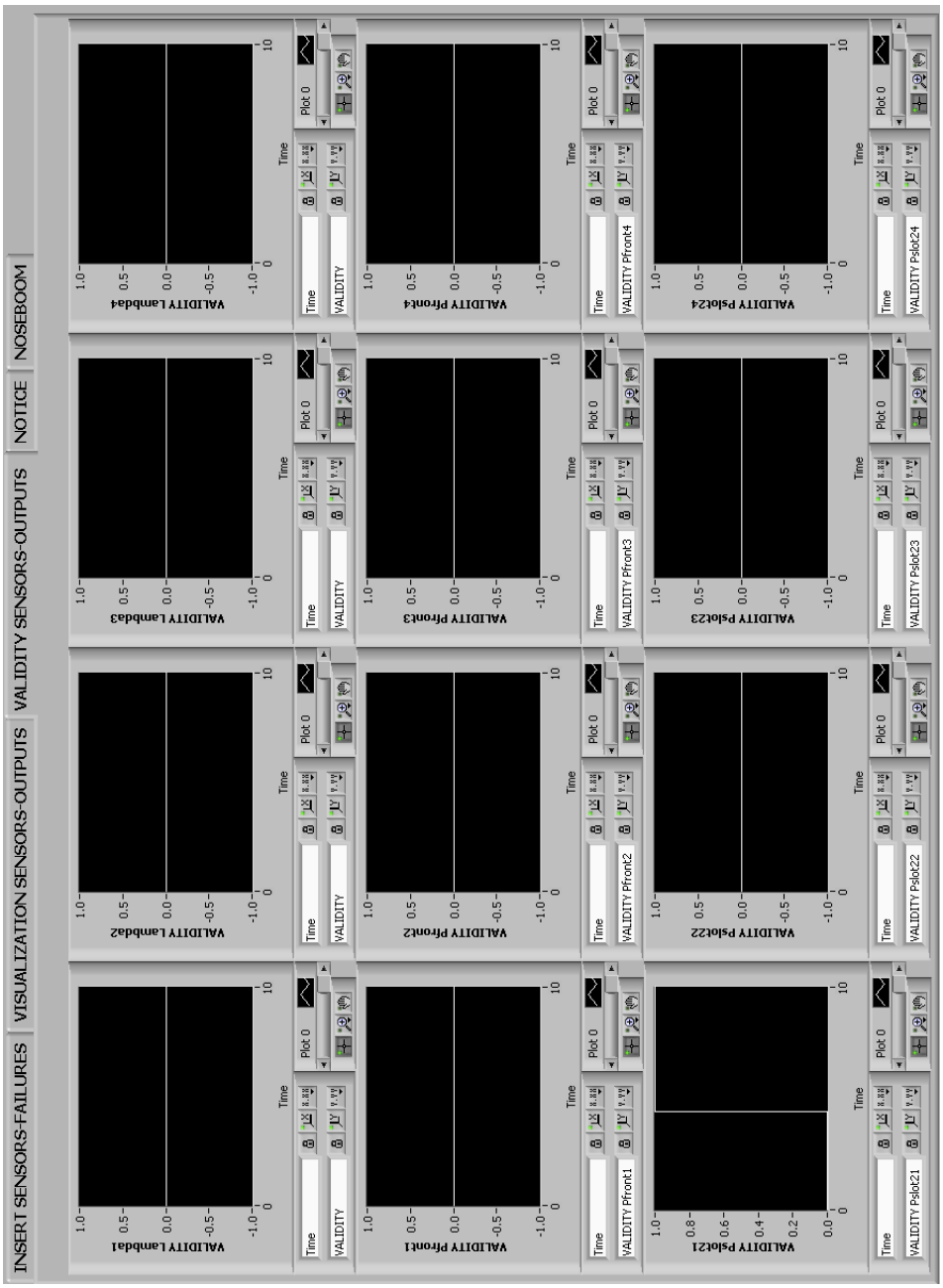

Fig. 5.22 : Control_Pannel.vi (3' finestra)

La seconda finestra, *Visualization Sensors-Outputs*, (fig. 5.21) mostra i grafici relativi alle uscite della quattro sonde, la terza finestra, *Validity Sensors-Outputs,* (fig. 5.22) mostra i grafici relativi ai segnali di validità associati ad ogni uscita di simulazione.

La quarta finestra, *Notice*, (fig. 5.23) è dedicata ai commenti sui parametri delle avarie e infine la quinta finestra, *Noseboom*, (fig. 5.24) gestisce l'inserimento delle avarie e la visualizzazione delle uscite di simulazione del *Noseboom.*

In fig. 5.24 si nota analogamente a quanto descritto per la fig. 5.20 che tutte le avarie citate in § 4.4 sono presenti e sono rappresentate dai pulsanti: *Delay, Accuracy, Offset, Hysteresis, Departure +, Departure -, Lock, Up-limit, Low-limit e Validity.* Sono presenti i *display* che permettono di settare i parametri relativi alle varie avarie, i quali sono settati di *default* sui rispettivi valori nominali. I *display* sopra citati riportano il nome del parametro da settare (*Delay, Accuracy, Offeset, Deadband dell' Hysteresis, Up-limie e Low-limit*) e la sua unita di misura.

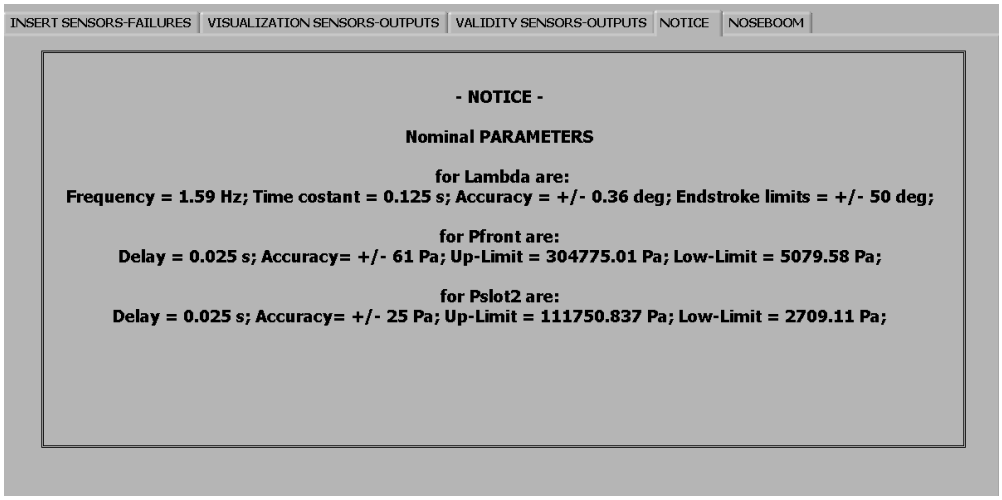

Fig. 5.23 : Control_Pannel.vi (4' finestra)

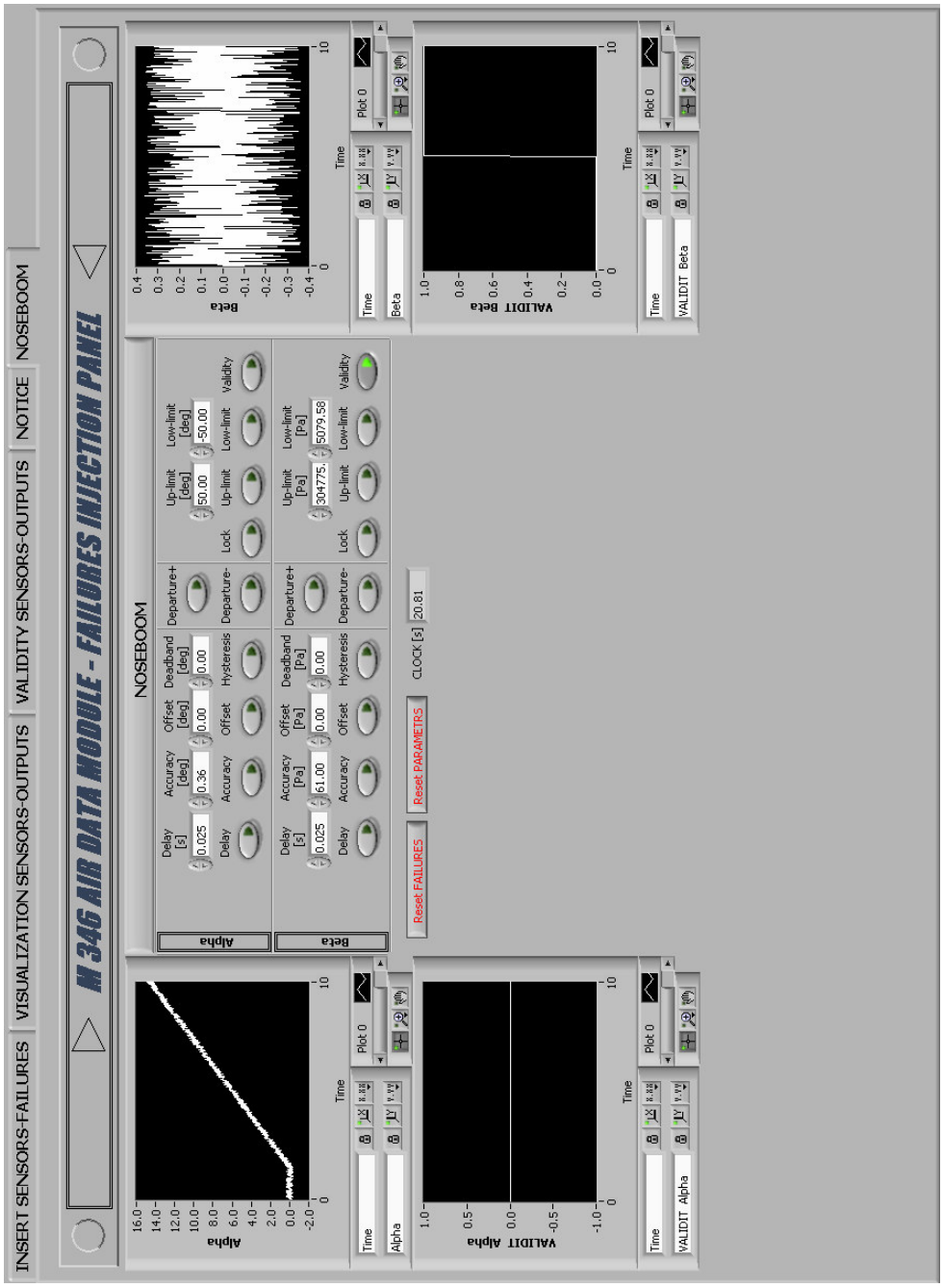

Fig. 5.24 : Control_Pannel.vi (5' finestra)

Ogni *file.vi* è costituito da due ambienti di lavoro, il primo di interfaccia rappresentato per esempio dalla fig. 5.20 chiamato *Front Panel* ed l'altro di costruzione del codice a blocchi del *Front Panel* chiamato *Block Diagram*. Nel *Block Diagram* sono presenti i blocchetti di ogni indicatore o controllo posto sul *Front Panel*, disposti in modo tale da realizzare un codice a blocchi.

Il *Block Diagram* del *file Control_Pannel.vi* è di dimensioni tali da non poter essere visualizzato tutto in una sola schermata a causa dall'elevato numero di blocchetti relativi ai vari pulsanti e display grafici. L'alto numero di blocchetti lo rende anche abbastanza ripetitivo. Per questi motivi vengono di seguito riportate immagini che rappresentano le parti più significative del *Block Diagram* del *file Control_Pannel.vi.*

In fig. 5.25 è riportato il codice a blocchi relativo alla trasmissione dei parametri e delle avarie al modello *Matlab-Simulink®* attraverso i blocchetto *UDP write* e *UDP open.* In fig. 5.25 si osserva inoltre che i segnali relativi alle avarie vengono raggruppati attraverso blocchetti multipli *Insert into Array,* in seguito vengono converti in formato di stringa attraverso il blocchetto *Type cast* come richiede *UDP write*, dopodiché vengono disposti in ordine inverso attraverso il blocchetto *Reverse string* per poter essere letti tutti ed infine si collegano al *UDP write* che spedisce i dati all'indirizzo del computer ricevente. Il tutto è inserito all'interno di una cornice che rappresenta un ciclo *while* in modo tale da mandare i segnali di avaria in maniera ripetitiva per tutta la durata della simulazione.

In fig. 5.26 è riportato invece il codice a blocchi relativo alla ricezione e alla visualizzazione delle uscite del modello *Matlab-Simulink®* attraverso i blocchetti *UDP open* e *UDP read*. Si osserva che i dati ricevuti sottoforma di stringa vengono disposti in ordine inverso per poter essere letti tutti, vengono convertiti in formato di *Array-double* attraverso il blocchetto *Type cast,* vengono separati attraverso il blocchetto *Index Arrray* ed infine vengono inviati ai display di visualizzazione grafica. Il tutto all'interno di un ciclo *while* analogamente al *softawre* di ricezione. All'esterno del ciclo *while* è presente un codice a blocchi

che permette di cancellare i display di visualizzazione grafica ad ogni avvio del *file*.

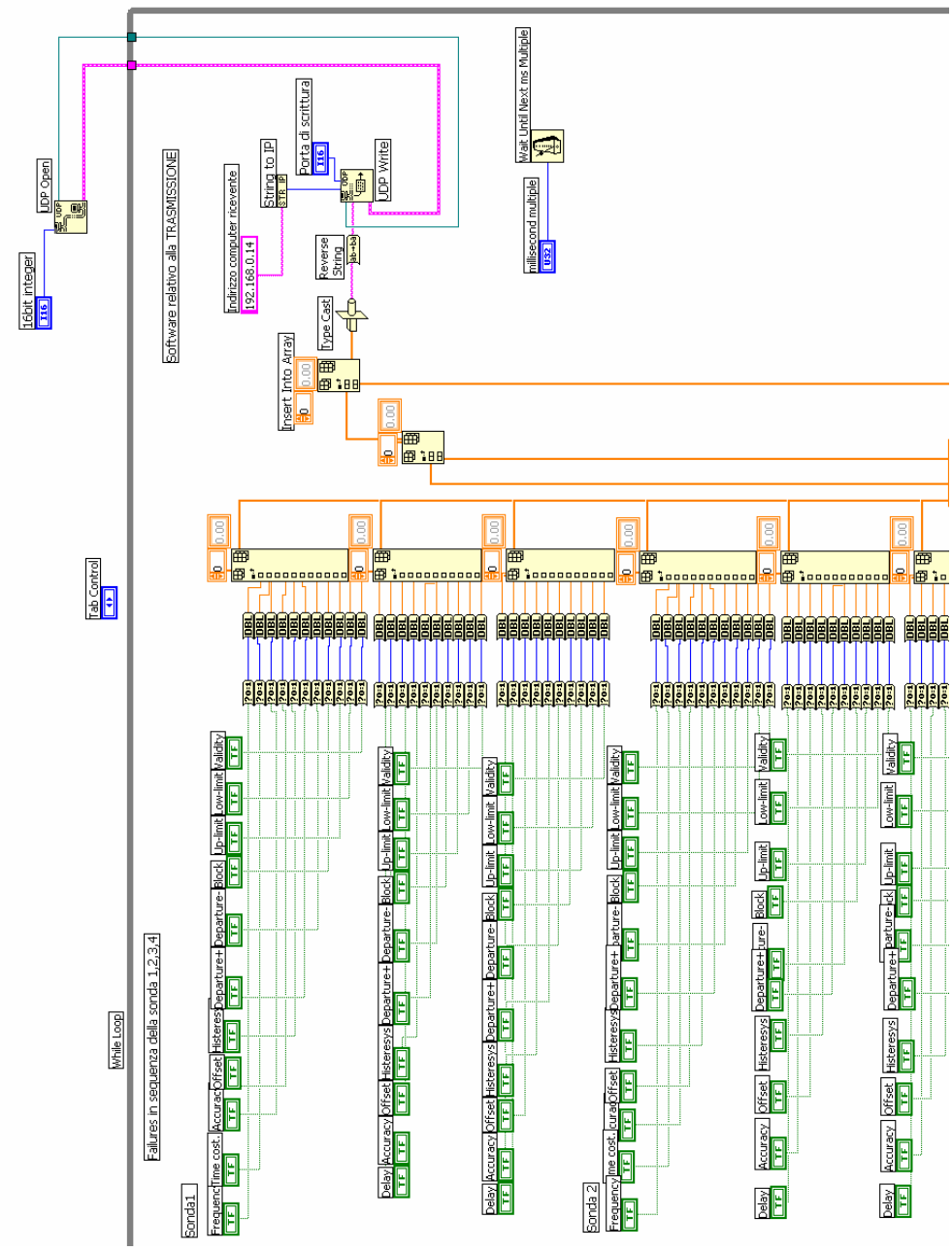

Fig. 5.25 : Particolare del Control_Pannel.vi Diagram (Trasmissione)

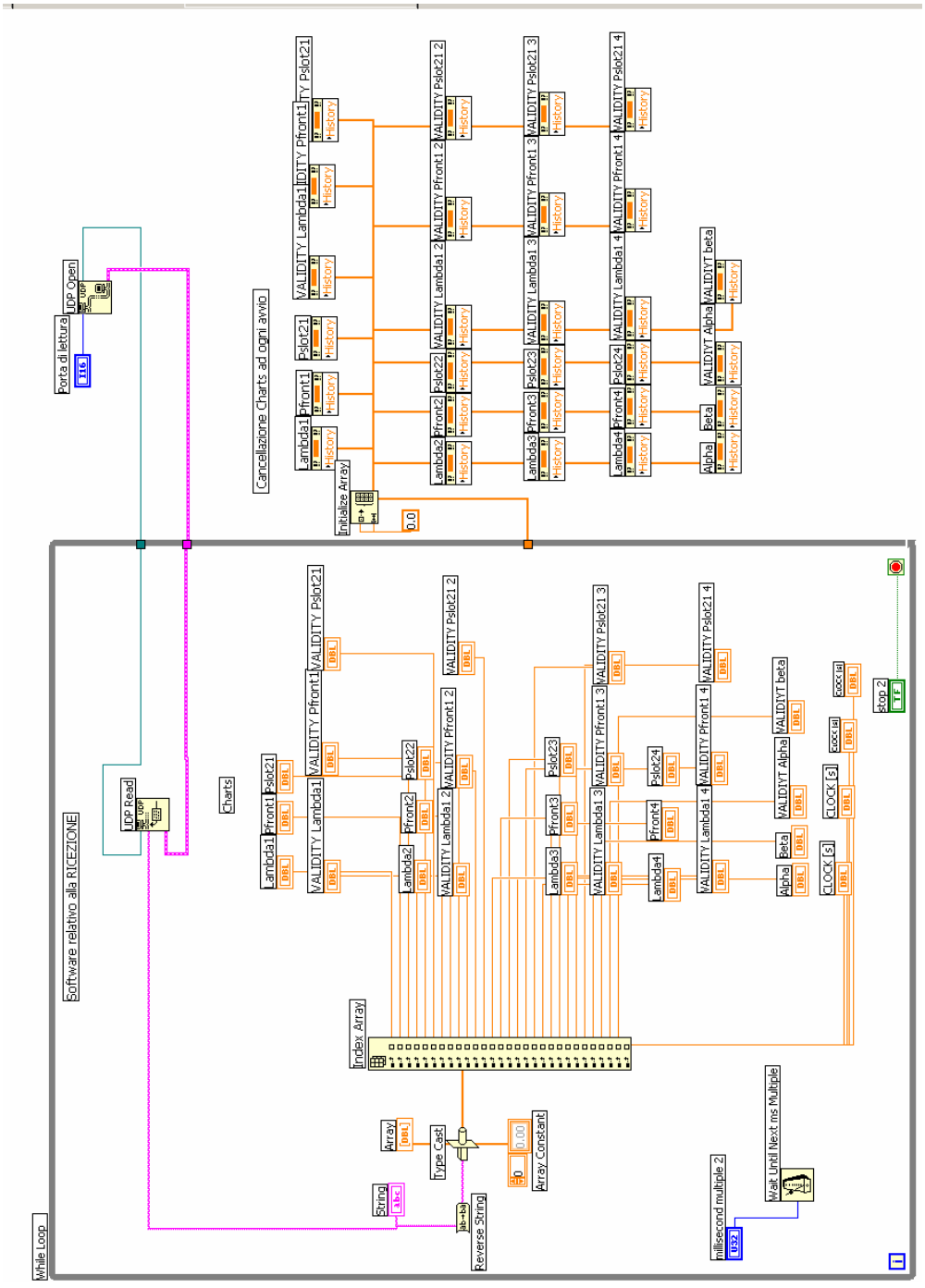

Fig. 5.26 : Particolare del Control_Pannel.vi Diagram (Ricezione)

5.2.5 Data_test.m

Questo *file* è strettamente legato al modello di simulazione predisposto per effettuare i test, contiene al suo interno i dati necessari per effettuare i test statici (per maggiori dettagli si rimanda al Capitolo 6), integra al suo interno il *file* precedentemente descritto *Manovre.m* e permette di settare il tipo di test da effettuare.

5.2.6 ADS_model_TEST.mdl

Come già accennato questo *file* (fig. 5.27) è soltanto una versione diversa dello stesso modello che è predisposta per la visualizzazione e il salvataggio dei risultati di simulazione dei vari tipi di test effettuati non in tempo reale.

Le differenze principali con il *file ADS_model_RT.mdl* consistono nel fatto che il modello non si collega al pannello di visualizzazione e inserimento di avarie realizzato in *LabVIEW®*, ma le *failure* e le visualizzazioni sono gestite interamente in *Matlab-Simulink®*, inoltre non girando in tempo reale è possibile utilizzare dei blocchetti (*To workspace*) di *Matlab-Simulink®* molto utili per salvare i dati di simulazione direttamente in *Matlab®*.

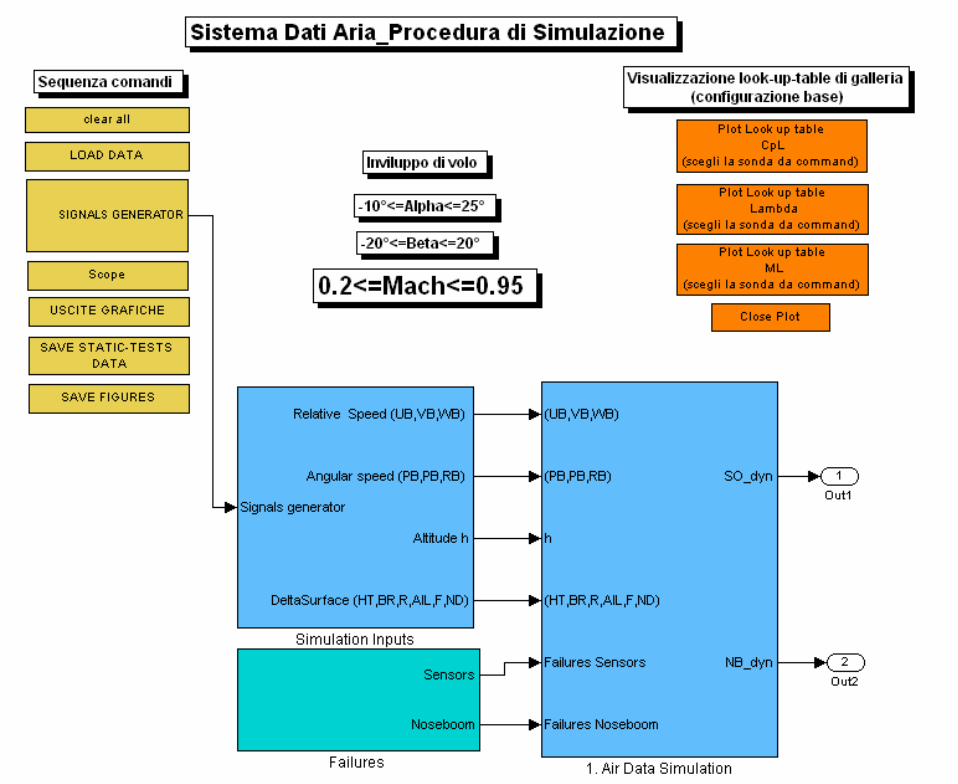

Fig. 5.27 : ADS_model_TEST.mdl

Oltre a queste differenze, al *level 0* del modello in fig. 5.27 sono presenti vari tasti in più rispetto al *file ADS_model_RT.mdl* che permettono sostanzialmente di visualizzare le grandezze di simulazione e di salvare quest'ultime sottoforma di dati o figure.

Il tasto *Scope* apre la finestra di fig. 5.28 che permette di premere i tasti relativi alle visualizzazioni delle grandezze di simulazione delle quattro sonde e del *Noseboom.* Questi tasti mandano in esecuzione rispettivamente le *routine Openscope_SIM.m, Openscope_NB.m e Closescope.m.*

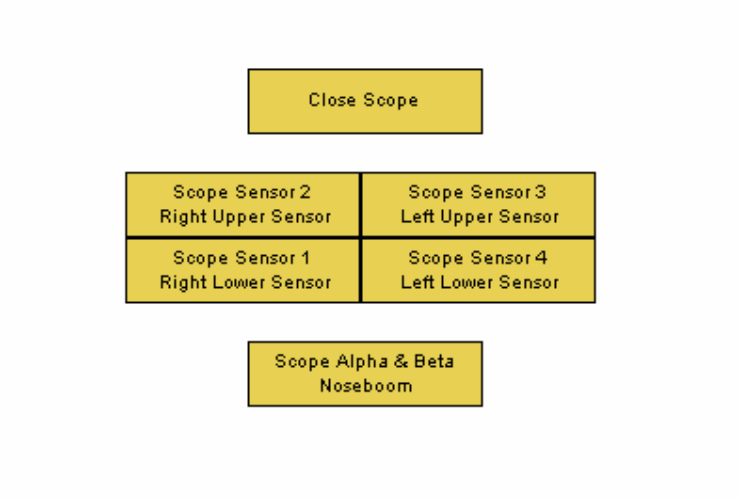

Fig. 5.28 : Scope

Il tasto *Uscite grafiche* manda in esecuzione il *file Grafici.m* che permette di visualizzare le uscite di simulazione nel formato *Matlab® ".fig"*.

Il tasto *Save figure* manda in esecuzione il *file Save_fig.m* che permette di salvare le figure aperte nelle cartelle *Save_Test_Man, Save_Test_Dyn o Save_Test_Failure* a seconda del tipo di test effettuato.

Infine il tasto S*ave Static-Test Data* manda in esecuzione il *file Save_Data_test.m* che permette di salvare i dati di simulazione in matrici in formato ASCII nella cartella *Save_Test_Static.*

6. Test di simulazione

Sono state effettuate vari tipi di prove di simulazione:

- **Test statici**, prove per esaminare la correttezza dei risultati ottenuti con i valori sperimentali a disposizione;
- **Test dinamici**, prove di simulazione con variazione dinamica dei parametri di ingresso alla procedura;
- **Test failure**, prove con *failure;*
- **Test con manovre**, prove relative a manovre realmente effettuate dal velivolo e fornite dall' azienda aeronautica italiana con cui il DIA collabora;
- **Test in Real-Time**, prove in tempo reale per confrontare gli output di simulazione di tali test con quelli relativi a simulazioni non in tempo reale.

6.1 Test statici

I test statici hanno permesso di documentare le grandezze di simulazione escludendo la dinamica dei sensori. Tali prove hanno interessato vari punti dell'inviluppo in α-β ed alcuni valori del Mach e della quota. I dati acquisiti oltre a rappresentare una documentazione, sono stati utili per effettuare dei confronti con i valori sperimentali a disposizione, quindi per validare la procedura di simulazione.

I test sono stati effettuati in automatico facendo in modo che il modello di simulazione utilizzasse gli ingressi di quota, Mach, α, e β predisposti in maniera opportuna del *file Data_test.m.* Tale *file* genera i vettori degli ingressi, ordinati in

modo tale da simulare tutte le combinazioni di quota, Mach, α e β scelti. Le simulazioni sono state effettuate considerando quattro valori quota: 0, 4000, 8000, 13200 m, nove valori di Mach pari a 0.2, 0.4, 0.6, 0.7, 0.8, 0.85, 0.9, 0.92, 0.95 e campionando l'inviluppo di volo in α-β con curve ad α costante e β variabile e β variabile e α costante spaziate di 5^o sia in α che in β.

I risultati di simulazione sono stati salvati in matrici in formato ASCII ed in Appendice B vengono riportati i dati di alcune simulazioni sottoforma di tabelle.

Di seguito sono riportati invece, alcuni risultati dei confronti tra i valori di galleria a disposizione opportunatamente interpolati e i valori simulati dalla procedura per le grandezze λ_i, Cp_{fronti}, e Cp_{sloti} (fig. 6.1). La fig. 6.1 rappresenta un esempio dove si osservano gli errori tra i valori di uscita dalla procedura di simulazione e i valori sperimentali opportunamente interpolati in modo da poter calcolare la differenze in determinati punti α-β dell'inviluppo di volo.

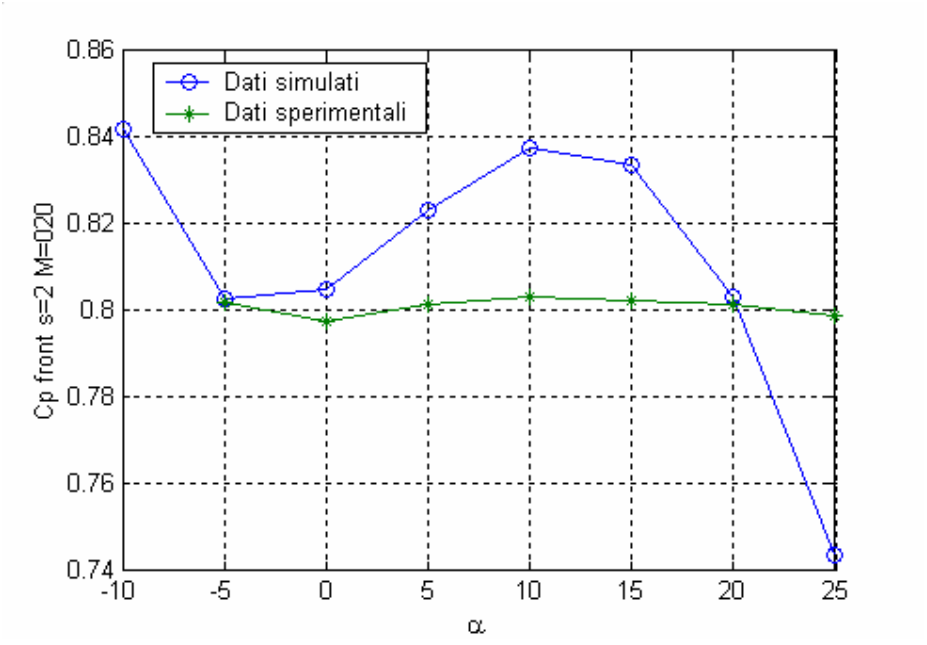

Fig. 6.1 : Cp_front sonda=2 Mach=0.2 β=0^o

Sono state confrontate quelle quantità che non dipendono dalla quota poiché i dati galleria a disposizione sono stati forniti per una sola pressione della camera di prova.

Nelle tabelle seguenti vengono riportate le suddette differenze in valore assoluto per le curve ad α costante e β variabile o le curve ad α variabile e β costante sull'inviluppo di volo in α-β per un data sonda e un fissato numero di Mach. Si riporta anche, per ogni tabella la deviazione standard σ di λ_i, Cp_{front_i} e Cp_{slot_i} calcolata tra i valori di galleria e i valori approssimati su quest'ultimi, con la tecnica dei Minimi Quadrati per l'intero inviluppo di volo ad un dato numero di Mach. Occore precisare che i valori di Cp_{front_i} e Cp_{slot_i} utilizzati per determinare la deviazione standard sono stati ricavati dalle grandezze sperimentali di flusso locale (Cp_{Li}, M_{Li}) attraverso le realzioni della sonda isolata (2.10) e la relazione che definisce il coefficiente di pressione locale Cp_{Li} (2.7).

La deviazione standard suddetta dovrebbe rappresentare un termine di paragone per valutare se gli errori commessi dalla procedura di simulazione si mantengono all'interno del campo di incertezza dei valori sperimentali stessi rappresentato da σ.

Si osservi che nelle tabelle seguenti i valori delle differenze per α = -10^o e β = +/- 20^o non sono presenti perché manca il valore sperimentale.

- Caso: Sonda 2, Mach=0.2, α variabile, $\sigma_{\lambda} = 0.407$, $\sigma_{Cp\_front} = 0.0352$, $\sigma_{Cp\_slot} = 0.0409$:

α [deg]	β [deg]	$\Delta\lambda_i$ [deg]	ΔCp_{front_i}	ΔCp_{slot_i}
-5.0000	0	0.4191	0.0009	0.0640
0	0	0.1795	0.0077	0.0019
5.0000	0	0.0055	0.0219	0.0129
10.0000	0	0.3353	0.0343	0.0276
15.0000	0	0.2744	0.0310	0.0167
20.0000	0	0.2572	0.0015	0.0350
25.0000	0	0.2182	0.0554	0.0886

Tab. 6.1 : Sonda 2, Mach=0.2, α variabile

- Caso: Sonda 3, Mach=0.4, β variabile, $\sigma_{\lambda} = 0.572$, $\sigma_{Cp\_front} = 0.0341$, $\sigma_{Cp\_slot} = 0.0402$:

α [deg]	β [deg]	$\Delta\lambda_i$ [deg]	ΔCp_{front_i}	ΔCp_{slot_i}
0	-15.0000	0.0383	0.0208	0.0788
0	-10.0000	0.5464	0.0084	0.0495
0	-5.0000	0.6407	0.0024	0.0021
0	0	0.3962	0.0110	0.0042
0	5.0000	0.4205	0.0262	0.0048
0	10.0000	0.0618	0.0166	0.0028
0	15.0000	0.8650	0.0138	0.0025

Tab. 6.2 : Sonda 3, Mach=0.4, β variabile

- Caso: Sonda 4, Mach=0.6, α variabile, $\sigma_\lambda = 0.552$, $\sigma_{Cp\_front} = 0.0366$, $\sigma_{Cp\_slot} = 0.0328$:

α [deg]	β [deg]	$\Delta\lambda_i$ [deg]	ΔCp_{front_i}	ΔCp_{slot_i}
-5.0000	-10.0000	0.6316	0.0001	0.0003
0	-10.0000	0.8472	0.0350	0.0191
5.0000	-10.0000	0.7686	0.0139	0.0058
10.0000	-10.0000	0.9621	0.0330	0.0302
15.0000	-10.0000	1.2753	0.0546	0.0383
20.0000	-10.0000	1.0934	0.0763	0.0316
25.0000	-10.0000	1.1268	0.0722	0.0541

Tab. 6.3 : Sonda 4, Mach=0.6, α variabile

- Caso: Sonda 3, Mach=0.7, α variabile, $\sigma_\lambda = 0.595$, $\sigma_{Cp\_front} = 0.0528$, $\sigma_{Cp\_slot} = 0.0383$:

α [deg]	β [deg]	$\Delta\lambda_i$ [deg]	ΔCp_{front_i}	ΔCp_{slot_i}
-5.0000	0	0.7029	0.0393	0.0999
0	0	0.6360	0.0630	0.0061
5.0000	0	0.3213	0.0803	0.0070
10.0000	0	0.0935	0.0695	0.0016
15.0000	0	0.6601	0.0428	0.0176
20.0000	0	1.4067	0.0118	0.0248
25.0000	0	0.5310	0.0389	0.0463

Tab. 6.4 : Sonda 3, Mach=0.7, α variabile

- Caso: Sonda 1, Mach=0.8, α variabile, $\sigma_{\lambda} = 0.688$, $\sigma_{Cp\_front} = 0.0453$, $\sigma_{Cp\_slot} = 0.0476$:

α [deg]	β [deg]	$\Delta\lambda_i$ [deg]	ΔCp_{front_i}	ΔCp_{slot_i}
-5.0000	0	0.7491	0.1543	0.0492
0	0	0.5956	0.1227	0.0161
5.0000	0	0.3136	0.0981	0.0039
10.0000	0	0.3584	0.0714	0.0048
15.0000	0	0.0273	0.0519	0.0013
20.0000	0	0.2487	0.0386	0.0112
25.0000	0	0.2198	0.0028	0.0151

Tab. 6.5 : Sonda 1, Mach=0.8, α variabile

- Caso: Sonda 2, Mach=0.85, α variabile, $\sigma_{\lambda} = 0.628$, $\sigma_{Cp\_front} = 0.0636$, $\sigma_{Cp\_slot} = 0.0345$:

α [deg]	β [deg]	$\Delta\lambda_i$ [deg]	ΔCp_{front_i}	ΔCp_{slot_i}
-5.0000	0	0.2200	0.0377	0.0625
0	0	0.0628	0.0004	0.0315
5.0000	0	0.6547	0.0407	0.0606
10.0000	0	0.6144	0.0543	0.0664
15.0000	0	0.1613	0.0306	0.0530
20.0000	0	1.3169	0.0013	0.0640
25.0000	0	0.5328	0.0075	0.0580

Tab. 6.6 : Sonda 2, Mach=0.85, α variabile

- Caso: Sonda 3, Mach=0.9, α variabile, $\sigma_\lambda = 0.712$, $\sigma_{Cp\_front} = 0.0723$, $\sigma_{Cp\_slot} = 0.0314$:

α [deg]	β [deg]	$\Delta\lambda_i$ [deg]	ΔCp_{front_i}	ΔCp_{slot_i}
-5.0000	5.0000	0.3660	0.1766	0.0577
0	5.0000	0.2006	0.1470	0.0885
5.0000	5.0000	0.5549	0.1109	0.1171
10.0000	5.0000	0.4582	0.1476	0.0903
15.0000	5.0000	0.6761	0.2135	0.0600
20.0000	5.0000	1.0000	0.2394	0.0803
25.0000	5.0000	3.7042	0.1806	0.2006

Tab. 6.7 : Sonda 3, Mach=0.9, α variabile

- Caso: Sonda 2, Mach=0.95, α variabile, $\sigma_\lambda = 0.763$, $\sigma_{Cp\_front} = 0.0797$, $\sigma_{Cp\_slot} = 0.0327$:

α [deg]	β [deg]	$\Delta\lambda_i$ [deg]	ΔCp_{front_i}	ΔCp_{slot_i}
-5.0000	10.0000	0.0184	0.2206	0.2097
0	10.0000	0.8089	0.2477	0.1396
5.0000	10.0000	0.5970	0.2158	0.1523
10.0000	10.0000	0.3671	0.1777	0.1687
15.0000	10.0000	0.3357	0.1435	0.1684
20.0000	10.0000	0.6747	0.0960	0.1618
25.0000	10.0000	1.1145	0.0022	0.1101

Tab. 6.8 : Sonda 2, Mach=0.95, α variabile

Innanzi tutto va detto che gli errori che si osservano nelle precedenti tabelle è probabile che siano causati da più fattori. Si ricorda che i dati aerodinamici inseriti nello modello di simulazione sono il risultato di una approssimazione ai Minimi Quadrati dei dati originali di galleria, come descritto nel Capitolo 3. Inoltre va precisato che per effettuare i confronti precedentemente detti in particolari punti dell'inviluppo, è stato necessario interpolare i dati originali di galleria. Un altro aspetto da tenere in considerazione è che agli estremi dell'inviluppo di volo i polinomi approssimanti i dati sperimentali iniziano a discostarsi sensibilmente dai dati stessi.

Gli errori consistenti che si osservano ai numeri di Mach maggiori o uguali di 0.9 sono probabilmente dovuti al fatto che la procedura di approssimazione ai Minimi Quadrati filtra delle curve di dati di galleria che hanno un andamento fortemente oscillante. Un altro motivo potrebbe essere che a questi valori di Mach la correzione delle curve sperimentali a β variabile e α costante di Mach pari a 0.4 realizzata per ottenere le stesse ai Mach superiori descritta in § 3.2.2 inizi a comportare degli errori consistenti.

Si osserva inoltre che i valori degli errori calcolati per numeri di Mach minori di 0.9 sono dello stesso ordine delle deviazioni standard calcolate tra i valori di galleria e i valori approssimati ricavati da quest'ultimi. Per valori di Mach superiori o uguali a 0.9 gli errori valutati sono superiori alla deviazione standard suddetta. Tale risultato potrebbe essere dovuto a più fattori. Un primo motivo potrebbe essere legato al fatto che il modello della sonda isolata (2.10) a Mach vicini e superiori a 0.9 inizia a commettere degli errori non trascurabili. Un secondo motivo potrebbe essere legato al fatto che gli errori dati dall'approssimazione ai Minimi Quadrati sulle grandezze di flusso locale (Cp_{Li} e M_{Li}), vedi Capitolo 3, si amplificano attraverso l'algoritmo di simulazione e quindi comportino degli errori elevati sulle misure di pressione di uscita dalla

procedura. Questo comportamento si nota non solo per i casi rappresentati dalle tabelle precedenti, ma è una tendenza generale per ogni condizione di volo.

6.2 Test dinamici

I test dinamici hanno riguardato prove di simulazione con gli ingressi (quota, Mach, α, e β) variabili linearmente. I risultati di simulazione vengono proposti in tale paragrafo sottoforma di grafici. In figg. 6.2, 6.3 e 6.4 vengono riportate le misure della sonda 1 nel caso di α variabile da 0^o a 25^o dopo un ritardo 2.5 secondi, β=0, M=0.4, h=13200 m per una simulazione della durata di 30 secondi. Nelle figure sopra citate venie riportata le grandezza a cui è stata applicata una dinamica e la stessa grandezza senza dinamica per mettere in risalto le differenze.

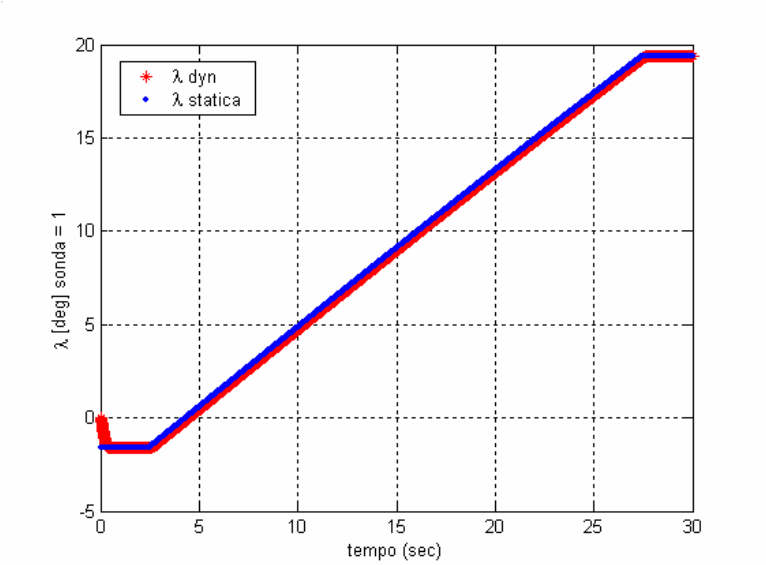

Fig. 6.2 : λ_1, α= 0^o÷25^o, β=0^0, M=0.4, h=13200 m

Facendo uno zoom di fig. 6.2 relativo all'intervallo di tempo compreso tra 0 e 5 sec. si può osservare che la risposta è quella tipica di un sistema del secondo ordine.

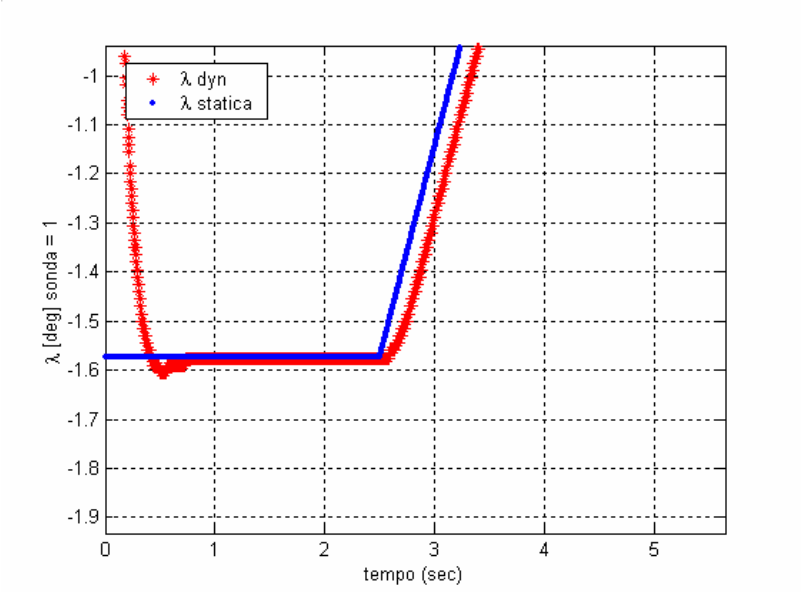

Fig. 6.3 : λ_1, α= 0°÷25°, β=0°, M=0.4, h=13200 m

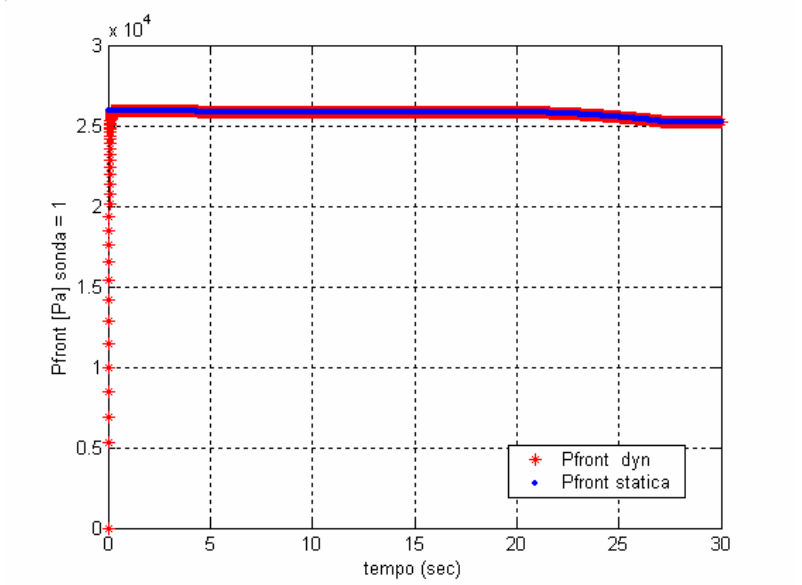

Fig. 6.4 :Pfront_1, α= 0°÷25°, β=0°, M=0.4, h=13200 m

Facendo uno zoom di fig. 6.4, in fig. 6.5 si osserva una classica risposta di un sistema del primo ordine come è quello che descrive la dinamica del misura pressione, P_{front_i}.

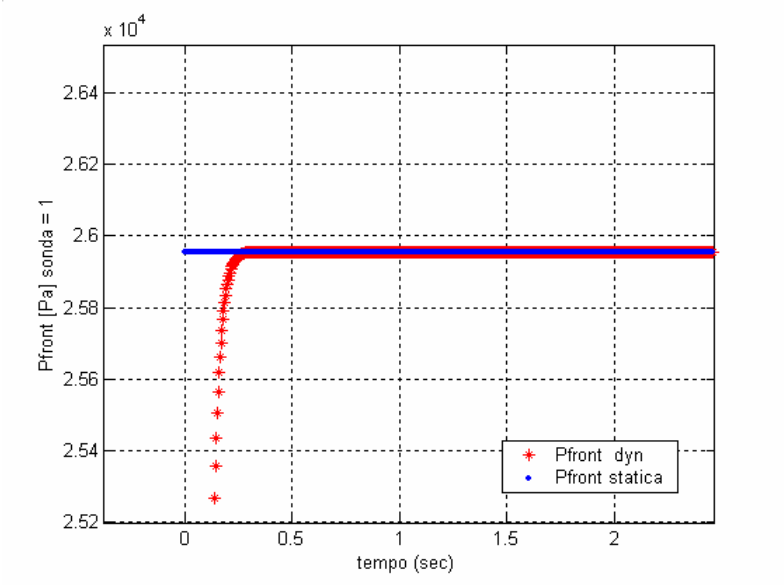

Fig. 6.5 :Pfront_1, α= 0°÷25°, β=0°, M=0.4, h=13200 m

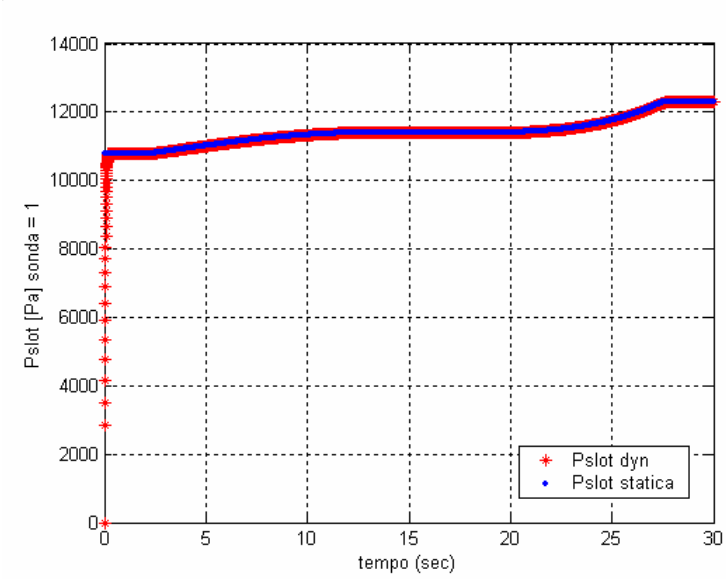

Fig. 6.6 :Pslot_1, α= 0°÷25°, β=0°, M=0.4, h=13200 m

Facendo uno zoom di fig. 6.6, in fig. 6.7 si osserva una classica risposta di un sistema del primo ordine come è quello che descrive la dinamica del misura pressione, P_{slot2_i}.

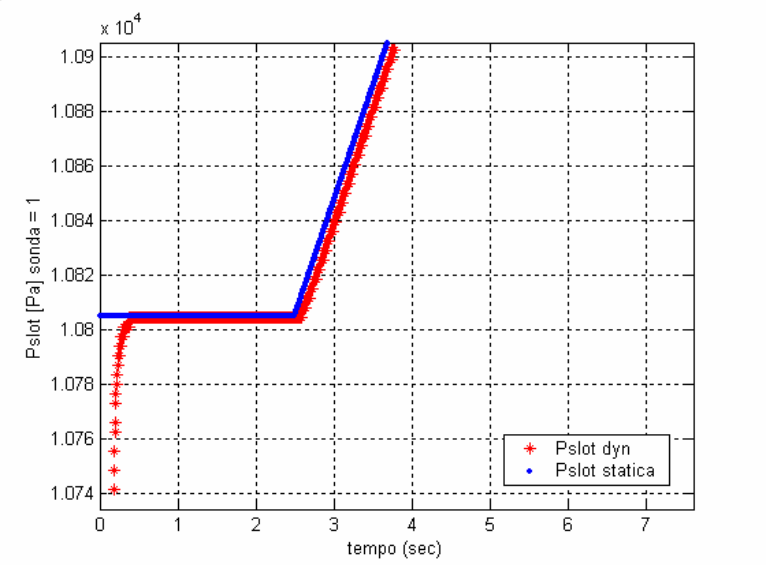

Fig. 6.7 :Pslot_1, α= 0°÷25°, β=0⁰, M=0.4, h=13200 m

6.3 Test *failure*

Con il nome "test *failure"* si indicano quelle simulazioni che prevedono l'inserimento di una avaria durante la simulazione stessa. Le condizioni di avaria sono documentate di seguito sottoforma di grafici che rappresentano le misure delle sonde in presenza di una *failure*. Per la quasi totalità delle avarie documentate si è previsto l'attivazione della *failure* dopo 10 secondi di simulazione, sempre per la sonda 1, nel caso di α variabile da 0^o a 25^o dopo un ritardo 2.5 secondi, β=0, M=0.4, h=13200 m, per una simulazione della durata di 30 secondi. In caso contrario sono state specificate le caratteristiche di simulazione.

6.3.1 Failure relative alle misure angolari

Vengo di seguito riportate le figure che documentano gli effetti delle avarie sul trasduttore di posizione angolare come descritto in § 4.2.1.

Si riporta per brevità il nome della *failure* inserita, una rapida descrizione e la figura che la rappresenta.

- *Accuracy,* inserimento di tale *failure* al decimo secondo di simulazione. L'*accuracy* passa dal suo valore nominale di 0.36° ad un valore di 2°.

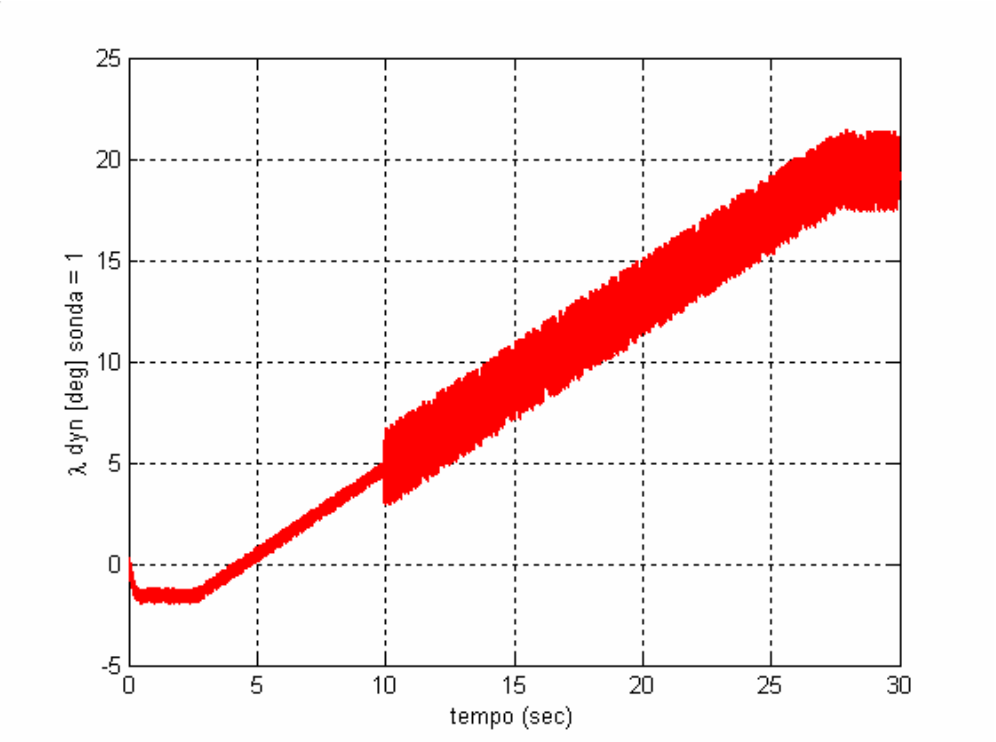

Fig. 6.8 : Failure Accuracy

- *Offset, failure* inserita al decimo secondo di simulazione. Il segnale di posizione angolare attraverso un transitorio lineare viene traslato di $+10^{o}$. Tale avaria è stato realizzata sommando al segnale in ingresso una rampa che si blocca quando raggiunge il valore dell'offset impostato.

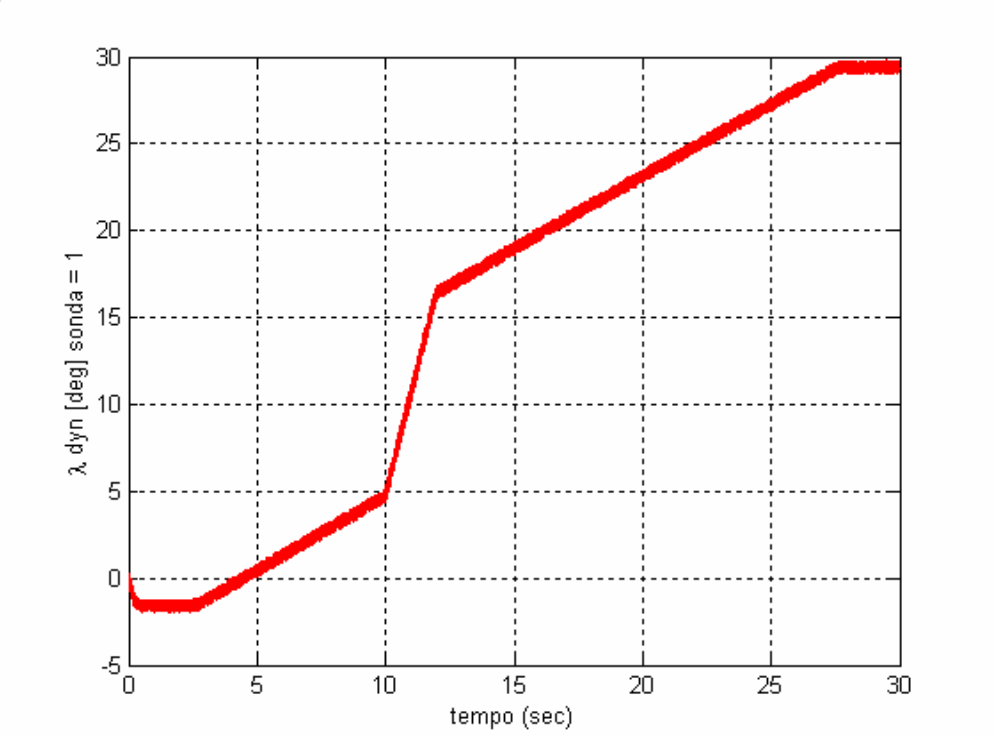

Fig. 6.9 : Failure Offset

- *Hysteresis, failure* inserita al decimo secondo di simulazione. In questo caso l'ingresso di α è stato preso sinusoidale con un ampiezza di 10^o in modo da mettere in evidenza gli effetti di tale avaria. In fig. 6.6 vengono riportate le curve del segnale della posizione angolare all'ingresso e all'uscita delle blocco *Sensors Dinamics* di fig. 5.8 che racchiude i modelli delle avarie. La *failure* di *Hysteresis* è stata realizzata andando a realizzare un codice che descrivesse il comportamento del blocchetto della libreria *Matlab-Simulink®* chiamato *Backlash* (per maggiori informazioni vedi l'*help* di *Matlab-Simulink®*) poiché questo non permetteva di variare il suo parametro *Deaband* durante la simulazione. La *Deaband* è stata variata da zero a 10^o.

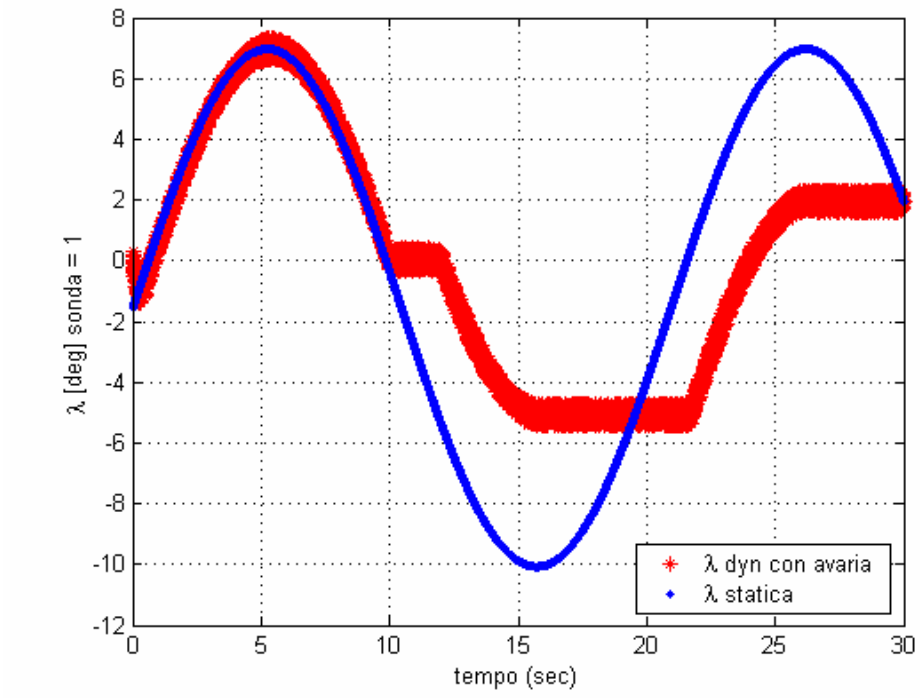

Fig. 6.10 : Failure Hysteresis

- *Departure to Upper limit, failure* inserita al decimo secondo di simulazione. La sonda raggiunge la posizione di fondo corsa di $+50^{o}$ attraverso un transitorio di andamento esponenziale. Tale avaria è stata realizzata andando a sommare al segnale di ingresso un esponenziale che raggiunto il valore di fondo corsa si blocca.

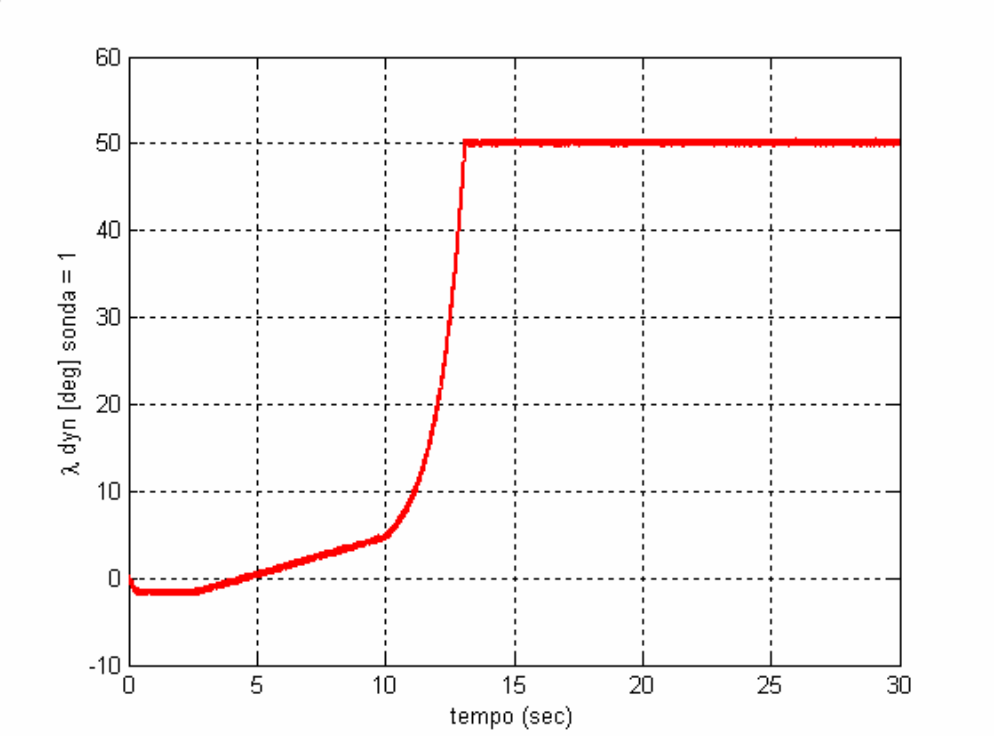

Fig. 6.11 : Failure Departure to Upper limit

- *Departure to Lower limit, , failure* inserita al decimo secondo di simulazione. Il comportamento è del tutto analogo al caso precedente con la differenza che la pendenza dell'esponenziale è negativa.

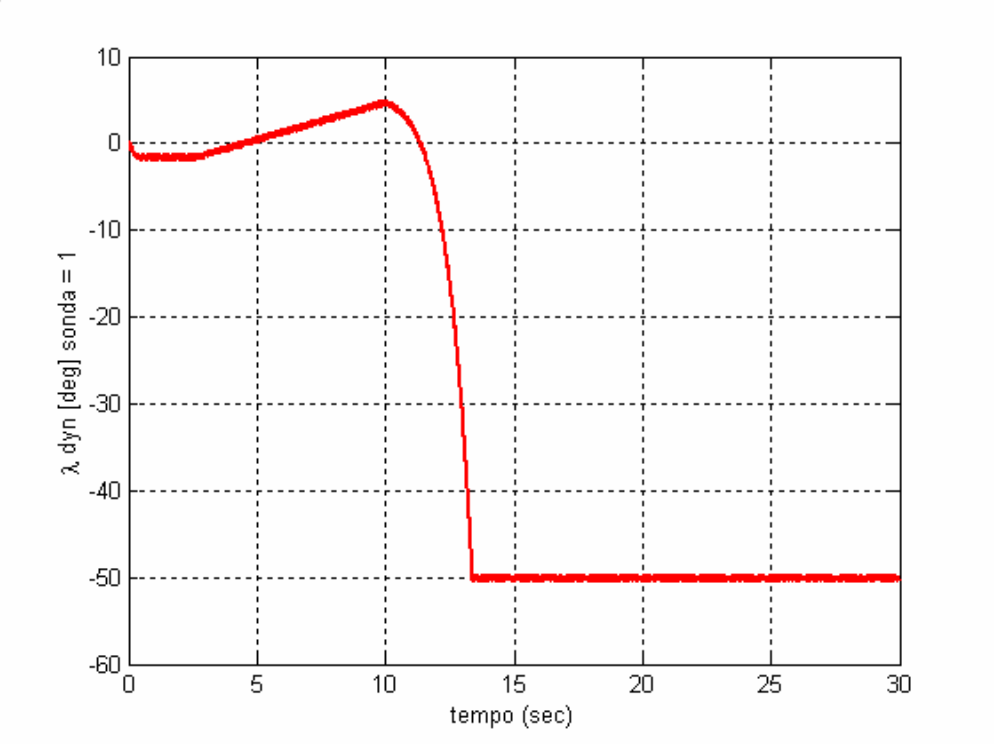

Fig. 6.12 : Failure Departure to Lower limit

- *Lock, failure* inserita al decimo secondo di simulazione. L'avaria consiste nel congelamento della posizione angolare nell'istante di attivazione della *failure.*

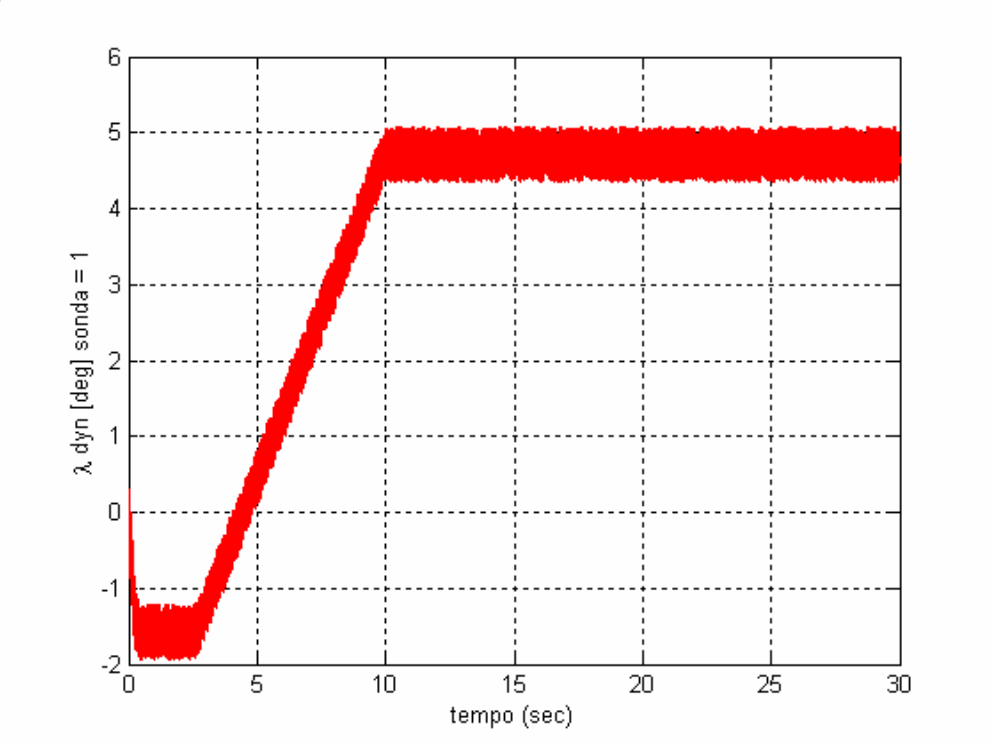

Fig. 6.13 : Failure Lock

- *Upper limit Endstroke e Departure to Upper limit.* E' un caso di più *failure* contemporanee sulla stessa sonda. L'avaria che riguarda la variazione del limite di fondo corsa è stata attivata prima dell' avaria *Departure to Upper limit,* che è stata inserita dopo 10 sec di simulazione . Il limite di fondo corsa è stato variato da $+50^{o}$ a $+30^{o}$.

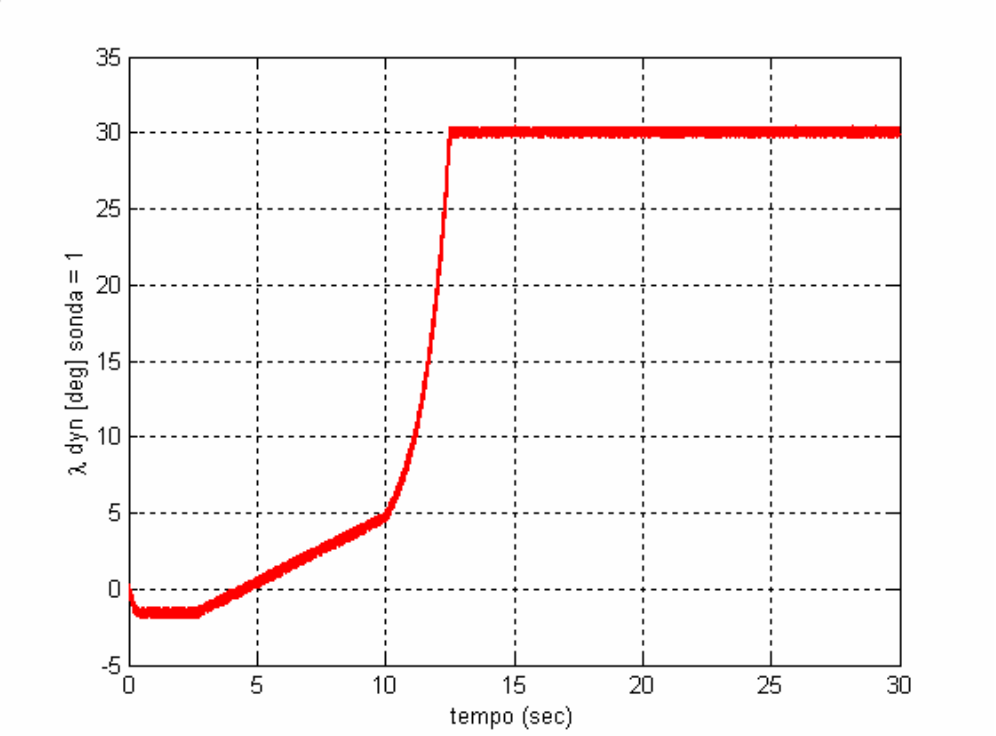

Fig. 6.14 : Failure Upper limit Endstroke

- *Lower limit Endstroke e Departure to Lower limit,* Caso sostanzialmente analogo al precedente. Il limite di fondo corsa è stato variato da -50° a -70°.

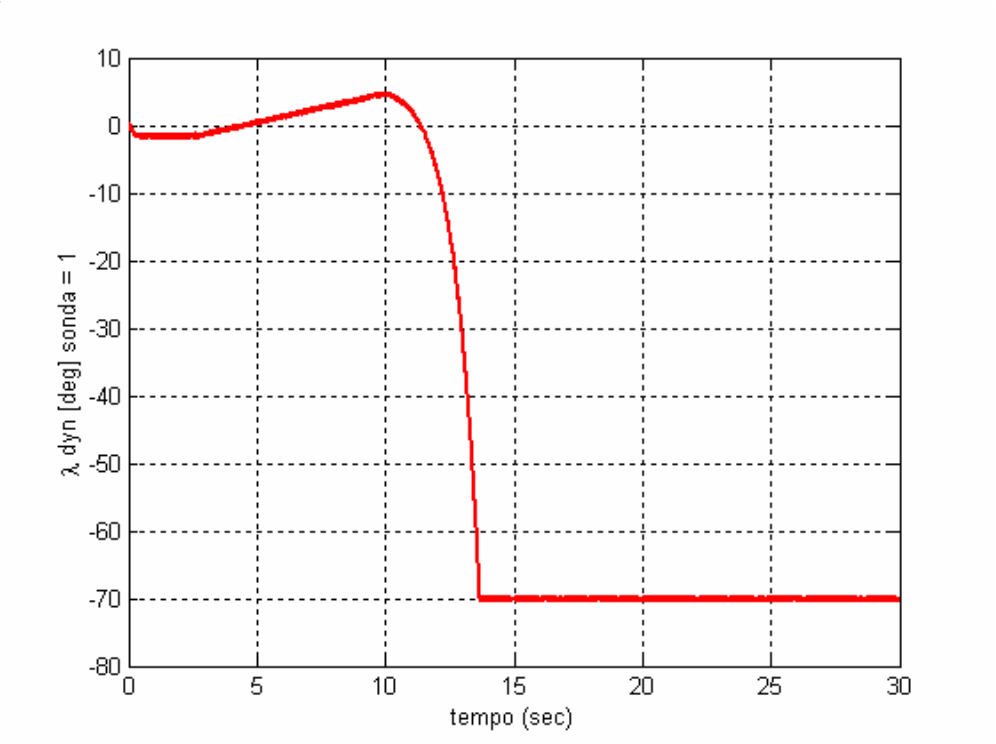

Fig. 6.15 : Failure Lower limit Endstroke

- *Frequency, failure* inserita al decimo secondo di simulazione.
  L'ingresso di α è un doppio gradino che varia da 0^o a 2^o e da 2^o a 4^o in modo da mettere in evidenza gli effetti di tale avaria. In questo caso l'avaria inserita modifica il valore della frequenza del sistema dinamico del secondo ordine che rappresenta il trasduttore di posizione angolare dal valore nominale di 1.59 Hz ad un valore di 1 Hz, a parità di costante di tempo. In fig. 6.15 viene riportato uno zoom significativo che mostra la variazione di comportamento della sonda. Si osserva che per rendere più evidenti gli effetti dell'avaria l'*accuracy* del segnale è stata posta a zero.

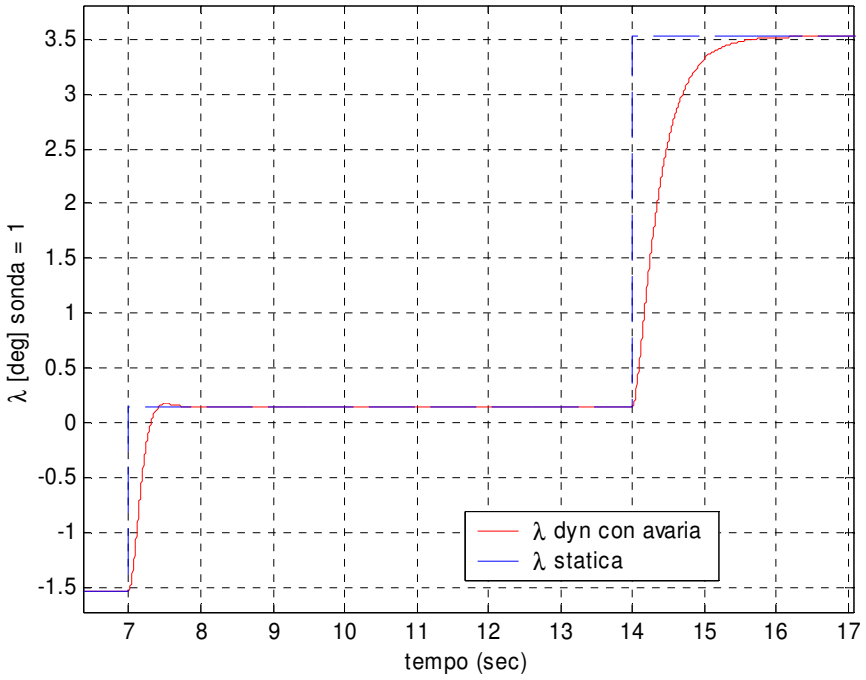

Fig. 6.16 : Failure Frequency

- *Time costant, failure* inserita al decimo secondo di simulazione. Come per la *failure frequency*, l'ingresso di α è un doppio gradino che varia da 0^o a 2^o e da 2^o a 4^o in modo da mettere in evidenza gli effetti di tale avaria. In questo caso l'avaria inserita modifica il valore della costante di tempo del sistema dinamico del secondo ordine che rappresenta il trasduttore di posizione angolare dal valore nominale di 0.125 sec ad un valore di 0.25 sec, a parità di frequenza. In fig. 6.16 viene riportato uno zoom che mostra il conseguente aumento del tempo di assestamento della dinamica del sensore. Si osserva che per rendere più evidenti gli effetti dell'avaria l'*accuracy* del segnale è stata posta a zero.

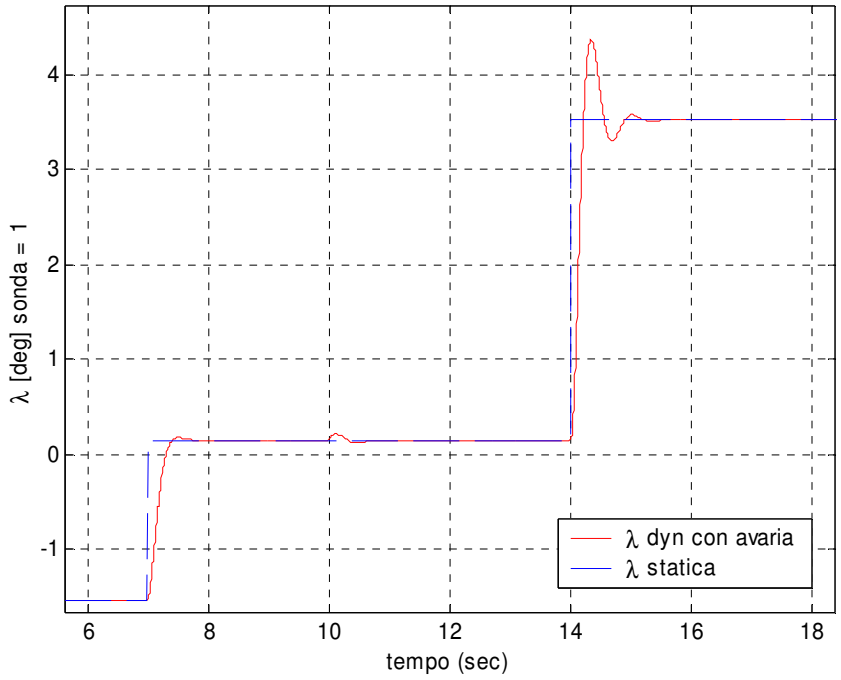

Fig. 6.17 : Failures Time costant

La *failure* di *validity* non introduce nessun effetto sulle uscita di posizione angolare, ma porta il segnale di validità associato alla posizione angolare da zero ad uno. Per la sua semplicità non viene riportata nessuna figura.

Si ricorda come già affermato nel Capitolo 4, che le avarie sul trasduttore d'angolo influenzano inevitabilmente le misure di pressione. Si riportano di seguito i grafici delle P_{front} e P_{slot2} a seguito di una avaria sul trasduttore di posizione angolare per un caso significativo.

- *Departure to Upper limit, failure* inserita al decimo secondo di simulazione. In fig. 6.18 si osserva come la failure attivata sul segnale della posizione angolare vada a disturbare le misure di pressione.

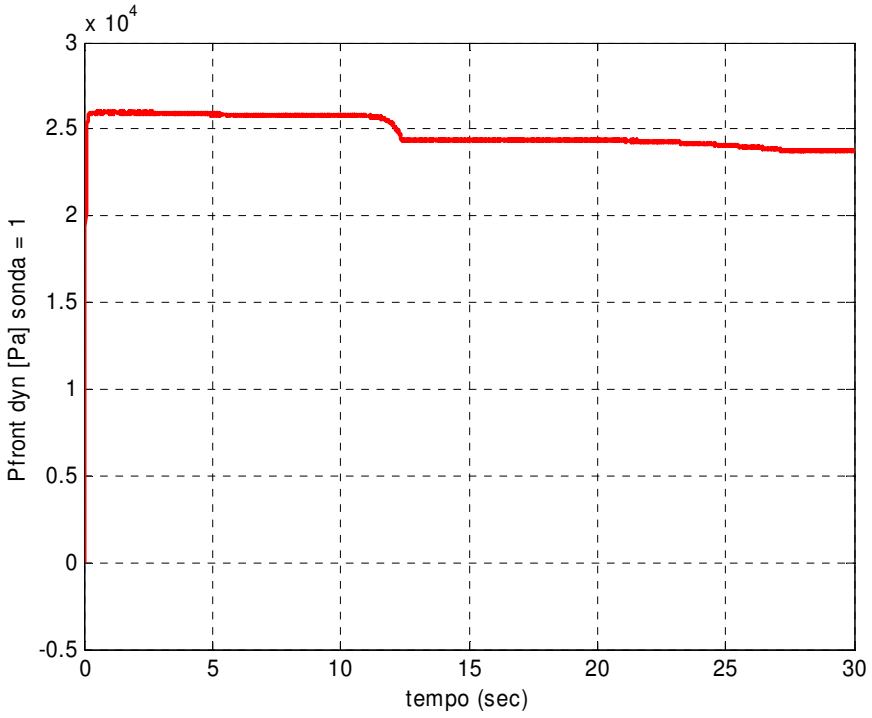

Fig. 6.18 : Pfront_1, Failure Departure to Upper limit su lambda_1

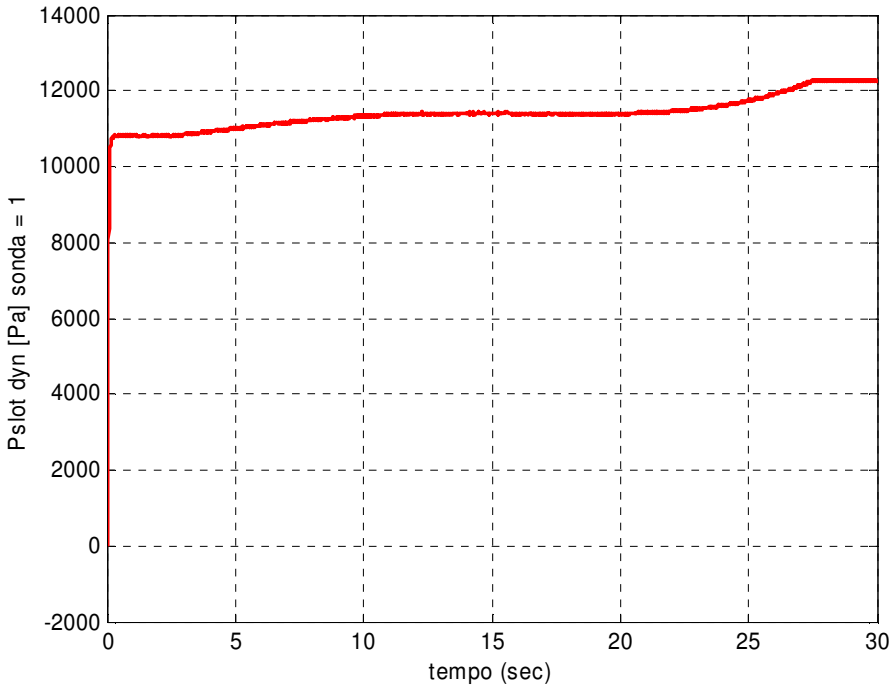

Fig. 6.19 : Pslot_1, Failure Departure to Upper limit su lambda_1

6.3.1 Failure relative alle misure di pressione

Si riportano di seguito le figure che documentano gli effetti delle avarie sul trasduttore di pressione. Si è scelto di riportare per brevità soltanto i grafici che mostrano la P_{front} poiché le tipologie di avaria sono le stesse sia per la P_{front} che per la P_{slot2}.

- *Accuracy, failure* inserita al decimo secondo di simulazione. La fig. 6.19 mette in evidenza l'effetto di una variazione dell' *Accuracy,* dal suo valore di 61 Pa a 400 Pa.

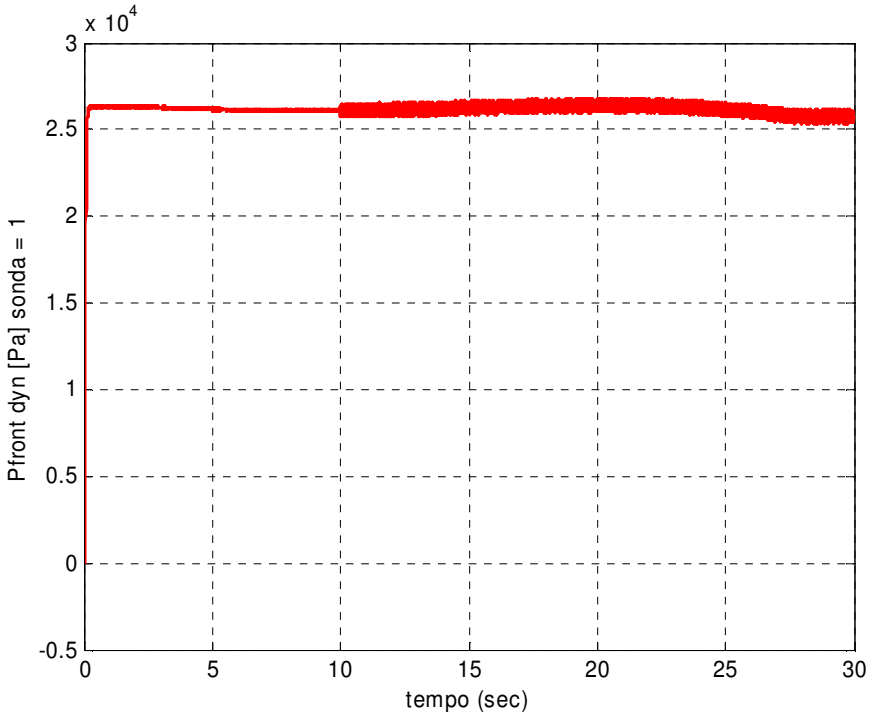

Fig. 6.20 : Pfront_1, Failure Accuracy

- *Offset, failure* inserita al decimo secondo di simulazione. La misura di pressione P_{front_1}, attraverso un transitorio lineare viene traslato di +1000 Pa. Tale avaria è stato realizzata sommando al segnale in ingresso una rampa che si satura quando raggiunge il valore dell'offset impostato.

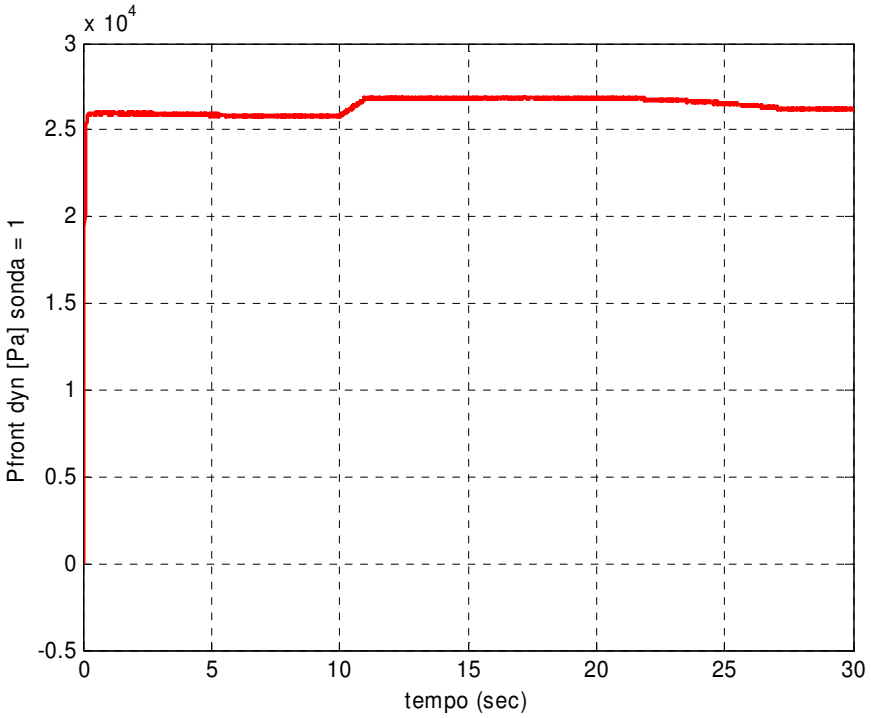

Fig. 6.21 : Pfront_1, Failure Offset

- *Hysteresis, failure* inserita al decimo secondo di simulazione. In questo caso l'ingresso di α è stato preso sinusoidale con un ampiezza di 10^o in modo da mettere in evidenza gli effetti di tale avaria. In fig. 6.21 vengono riportate le curve del segnale della P_{front_1} all'ingresso e all'uscita delle blocco *Sensors Dinamics* di fig. 5.8 che racchiude i modelli delle avarie. La *failure* di *Hysteresis* è stata realizzata andando a realizzare un codice che descrivesse il comportamento del blocchetto *Backlash* della libreria *Matlab-Simulink®* (per maggiori informazioni vedi l'*help* di *Matlab-Simulink®*) poiché questo non permetteva di variare il suo parametro *Deaband* durante la simulazione. La *Deaband* è stata variata da zero a 1000 Pa.

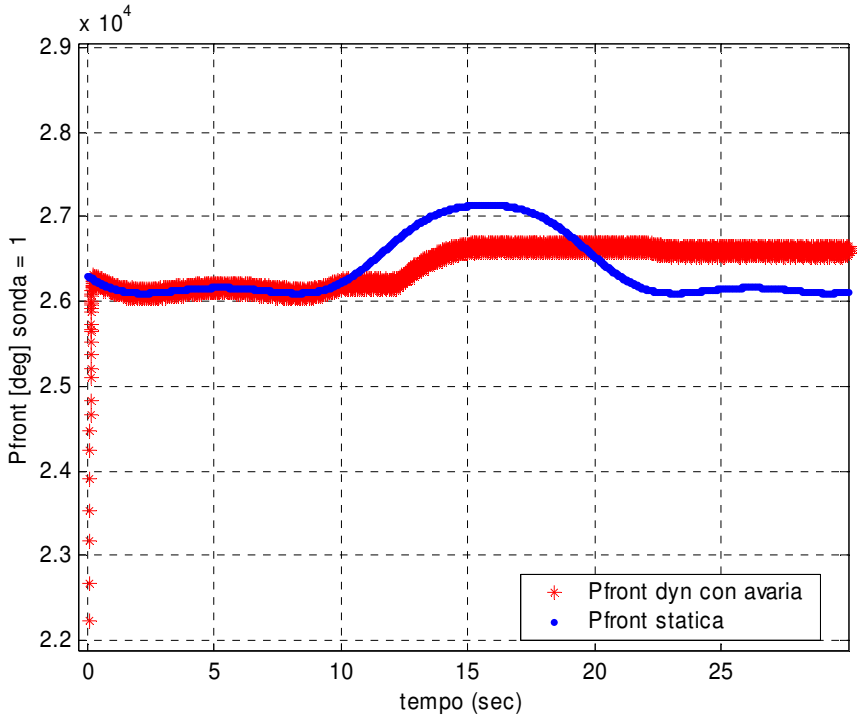

Fig. 6.22 : Pfront_1, Failure Hysteresis,

- *Departure to Upper limit, failure* inserita al decimo secondo di simulazione. La P_{front_1} raggiunge il limite superiore di pressione misurabile dal suo trasduttore (fissato a 304775 Pa), attraverso un transitorio con andamento esponenziale. Tale avaria è stata realizzata andando a sommare al segnale di ingresso un esponenziale che raggiunto il valore di fondo corsa si blocca.

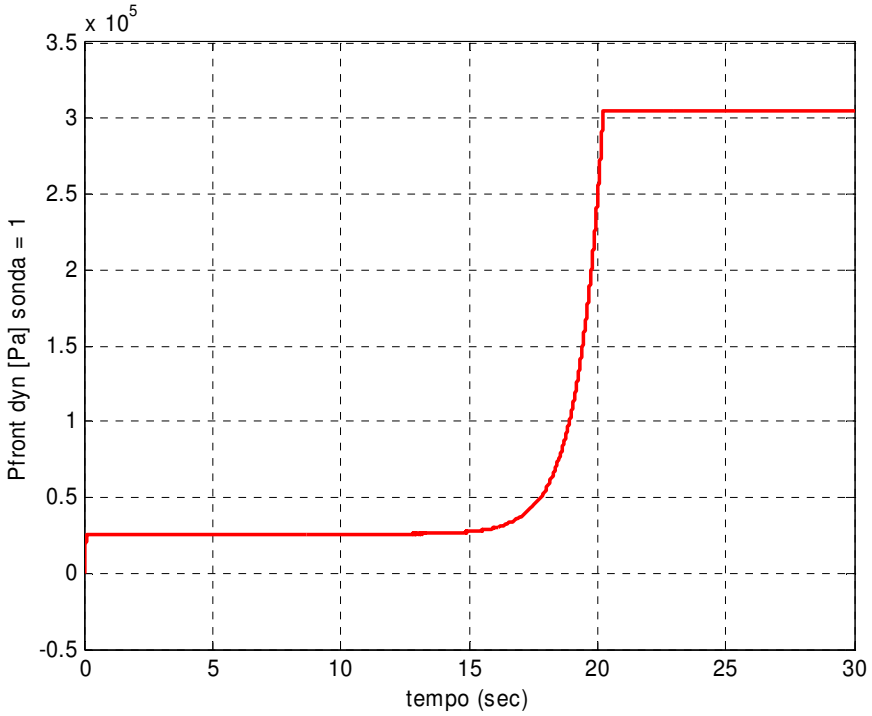

Fig. 6.23 : Pfront_1, Failure Departure to Upper limit

- *Departure to Lower limit, failure* inserita al decimo secondo di simulazione. Il comportamento è del tutto analogo al caso precedente con la differenza che la pendenza dell'esponenziale è negativa. La P_{front_1} raggiunge il limite inferiore di pressione (fissato a 5079 Pa) misurabile dal suo trasduttore.

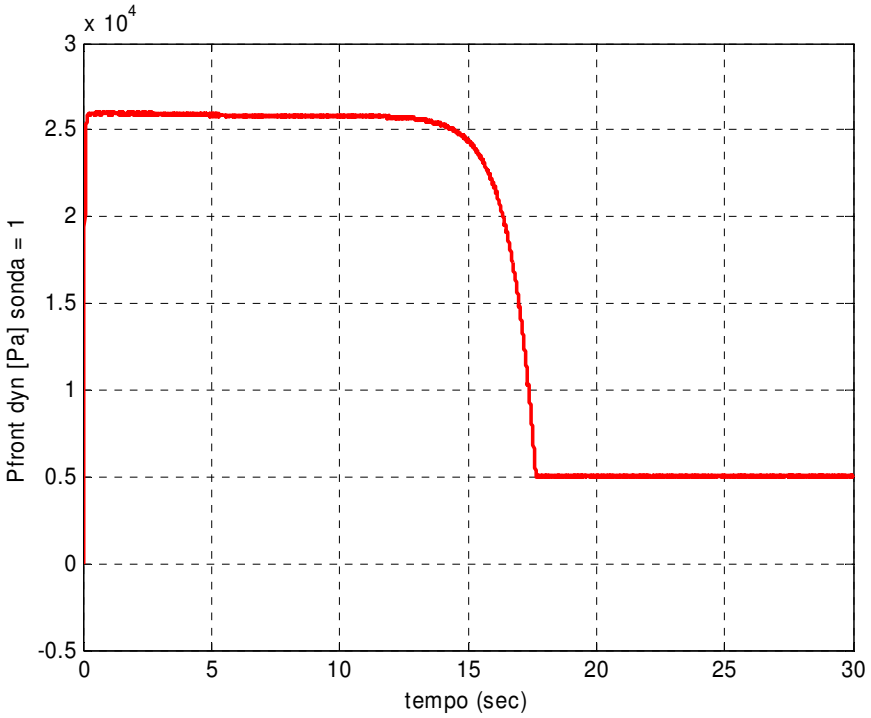

Fig. 6.24 : Pfront_1, Failure Departure to Lower limit

- *Lock, failure* inserita al decimo secondo di simulazione. L'avaria consiste nel congelamento della pressione nell'istante di attivazione della *failure.*

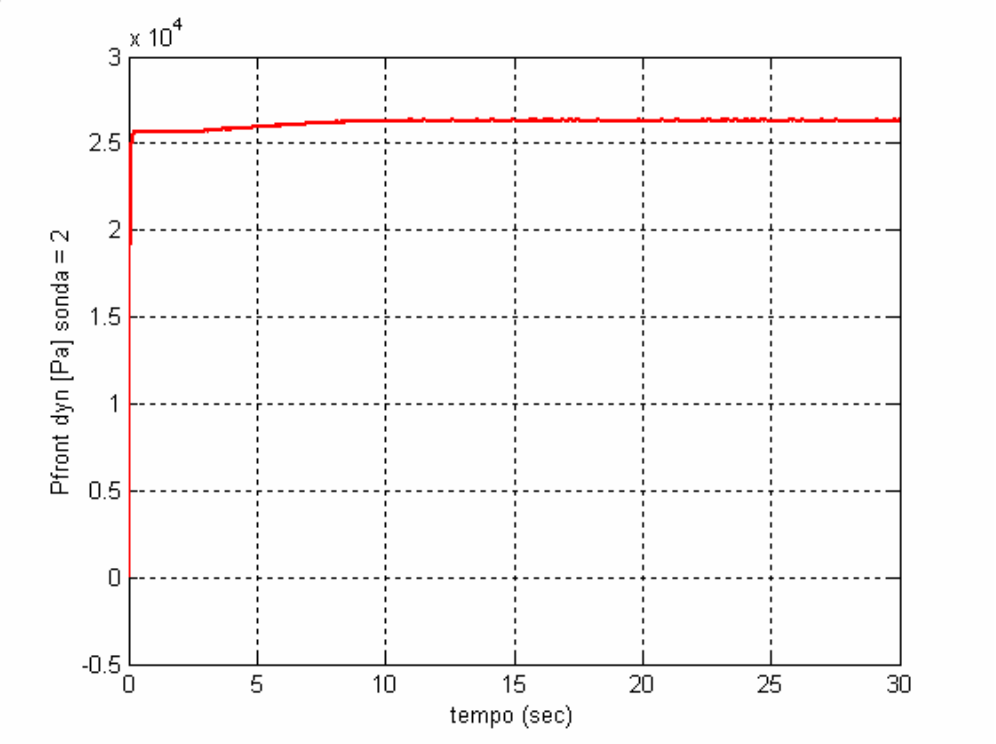

Fig. 6.25 : Failure Lock

- *Upper limit Endstroke e Departure to Upper limit.* E' un caso di più *failure* sulla stessa sonda. L'avaria che riguarda la variazione del limite di fondo corsa è stata attivata prima dell' avaria *Departure to Upper limit,* che è stata inserita dopo 10 sec di simulazione . Il limite di fondo corsa è stato variato da 304775 Pa a 150000 Pa.

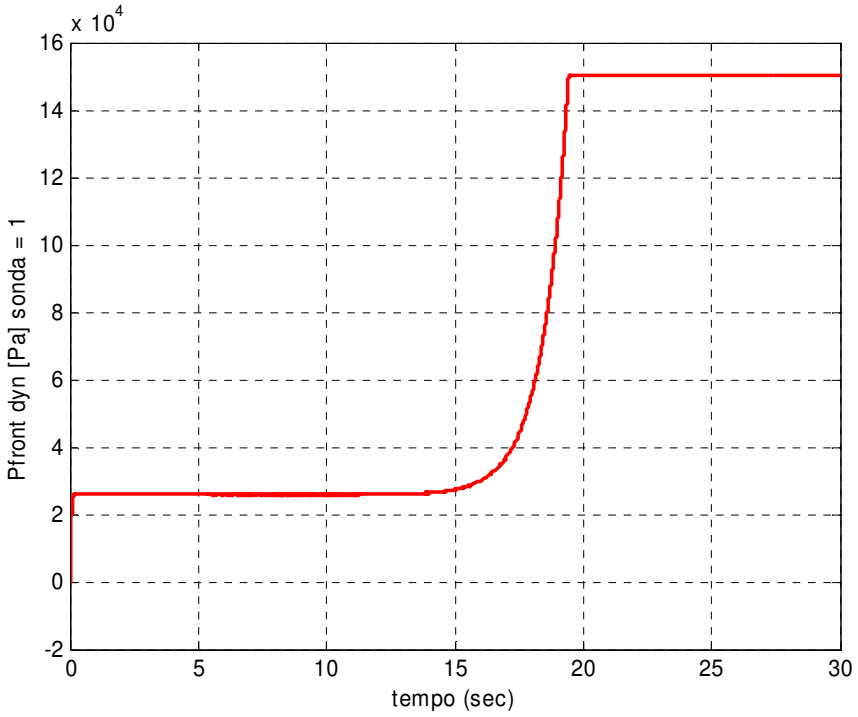

Fig. 6.26 : Pfront_1, Failure Upper limit Endstroke e Departure to Upper limit

- *Lower limit Endstroke e Departure to Lower limit,* Caso sostanzialmente analogo al precedente. Il limite di fondo corsa è stato variato da 5079 Pa a 0 Pa.

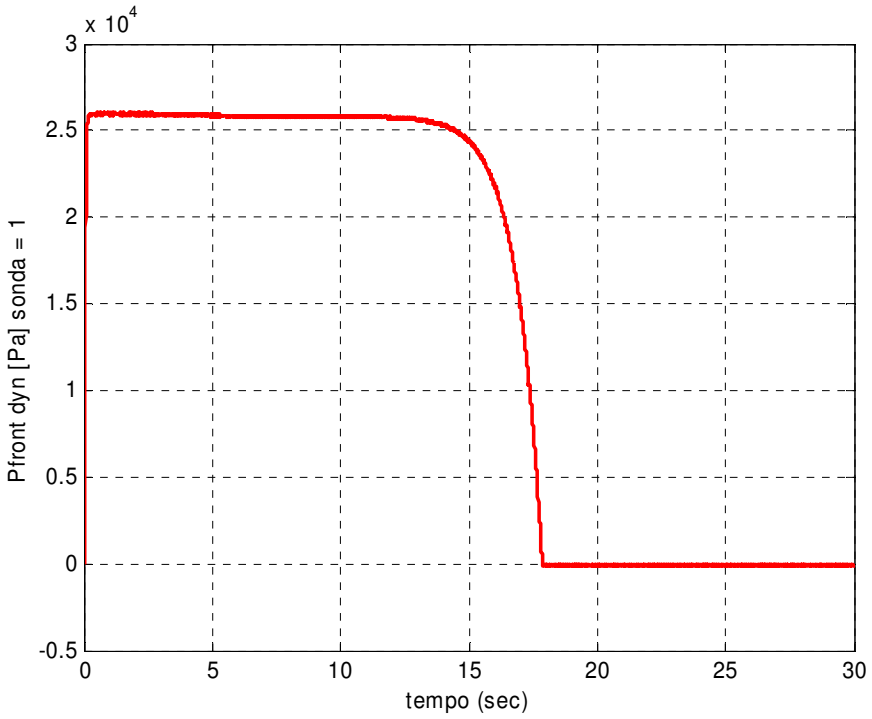

Fig. 6.27 : Pfront_1, Failure Lower limit Endstroke e Departure to Lower limit

- *Delay, failure* inserita al decimo secondo di simulazione. L'ingresso di α è un doppio scalino che varia da 0^o a 2^o e da 2^o a 4^o in modo da mettere in evidenza gli effetti di tale avaria. In questo caso l'avaria inserita varia il valore del ritardo del sistema dinamico del primo ordine con il quale si è approssimato il trasduttore di pressione, dal valore nominale di 0.025 sec ad un valore di 1 sec. In fig. 6.26 viene riportato uno zoom che mostra il conseguente aumento del ritardo della dinamica del sensore. Si osserva che per rendere più leggibile l'avaria, l'*accuracy* del segnale è stata posta a zero. In questo caso si è scelto di riportare P_{slot2} piuttosto che P_{front} poiché la prima metteva più in evidenza gli effetti della *failure*.

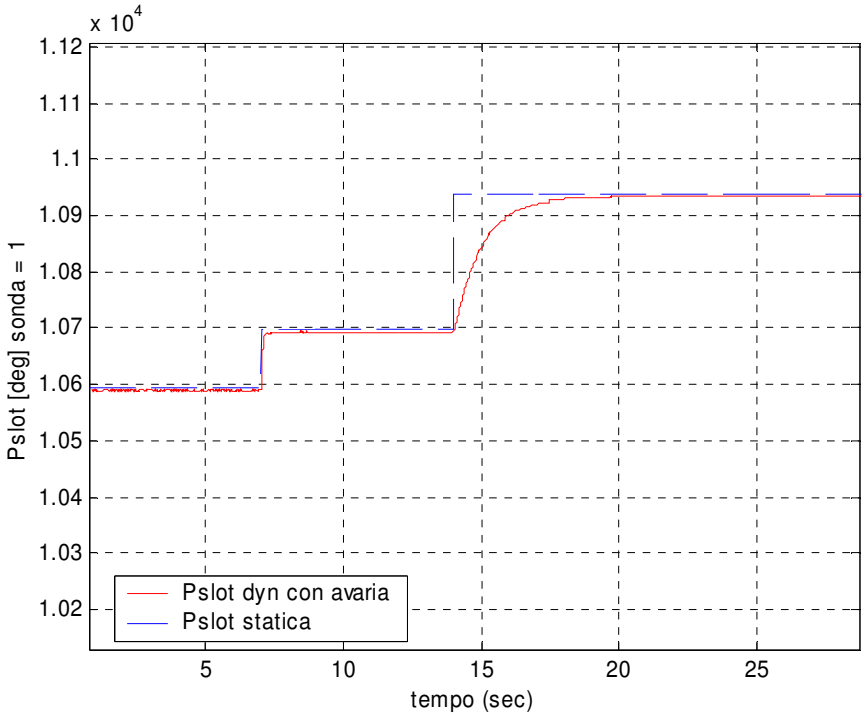

Fig. 6.28 : Pslot_1 Failures Delay

6.3.3 Failure relative al Noseboom

Le *failure* previste per le misure di posizione angolare (angolo di incidenza e angolo di derapata) del *Noseboom* sono analoghe a quelle del trasduttore di pressione per cui per non appesantire troppo la presente relazione sono state omesse.

6.4 Test con Manovre

Sono stati fatti test caricando gli ingressi di simulazione da dati salvati durante msimulazioni reali del velivolo in considerazione al fine di documentare e verificare le misure delle sonde e del *Noseboom*. Sono stati forniti i dati per tre tipi di manovre, identificate dalle seguenti sigle:

MA $\rightarrow$ Mil-A, manovra di Pull-up (manovra di richiamata);

RD $\rightarrow$ Rudder, manovra di puro timone verticale a partire da una condizione di aereo livellato, ad 1"g".

RP $\rightarrow$ Rolling Pull-out, manovra di rollio a partire da una condizione di bank diverso da 0 e sotto fattore di carico.

Vengono riportati di seguito i grafici relativi alle uscite di simulazione della sonda 1 per un caso appartenente ad ogni classe di manovra specificata sopra.
Manovra:

- MA h=1000 ft, Mach=0.46. In fig.6.29 si riportatono i grafici relativi agli ingressi di simulazione per la manovra specificata. Dalle figg. 6.30 e 6.33 si osserva che l'andamento di λ segue abbastanza fedelmente quello di α come è logico aspettarsi da una manovra di richiamata. In fig. 6.29 si osserva anche la presenza dell'*accuracy*.

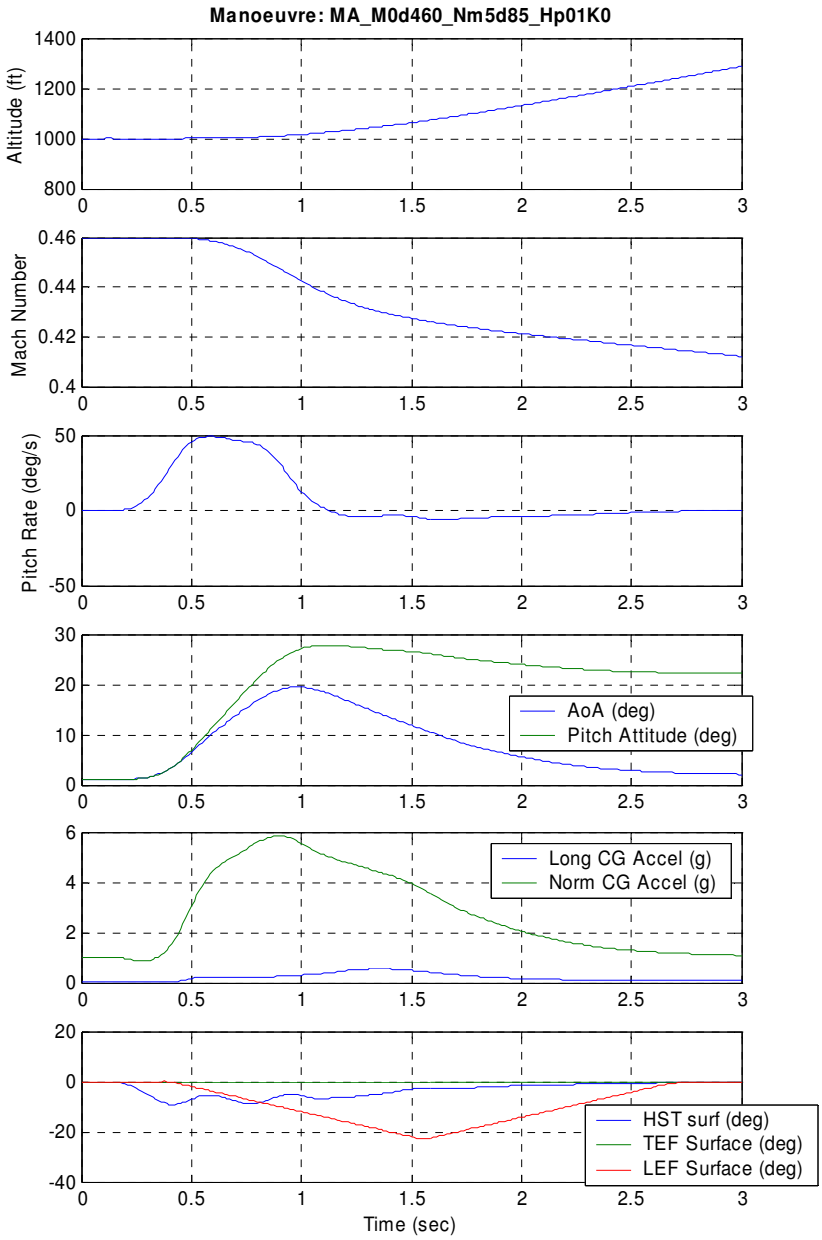

Fig. 6.29 : Ingressi di simulazione

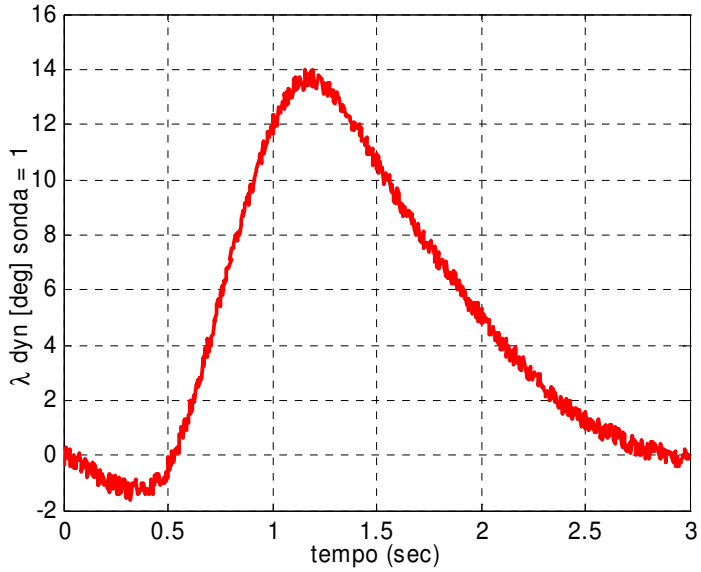

Fig. 6.30 : Lambda_1

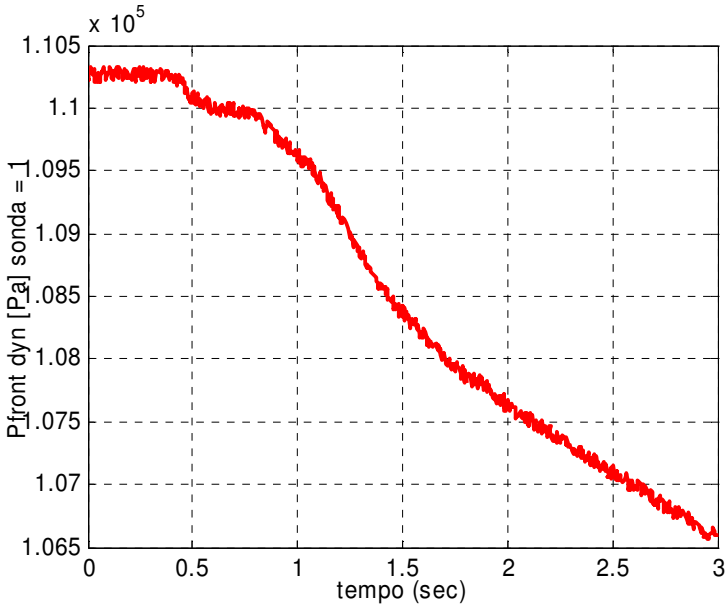

Fig. 6.31 : Pfront_1

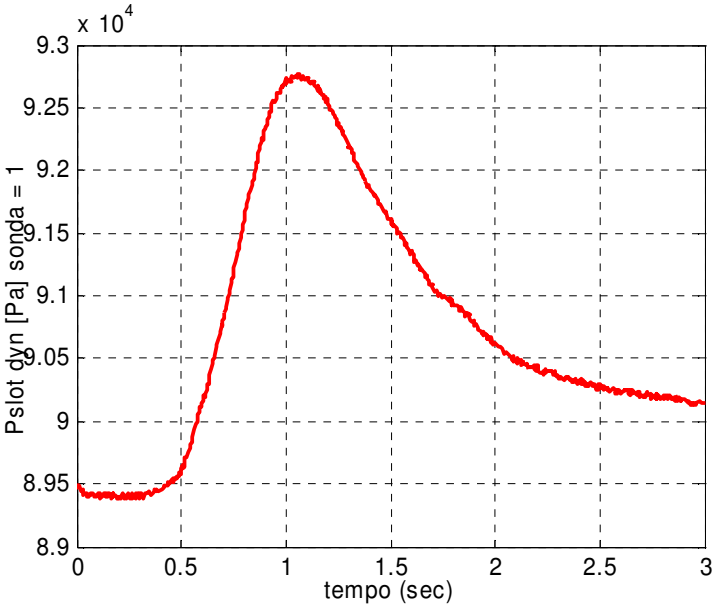

Fig. 6.32 : Pslot_1

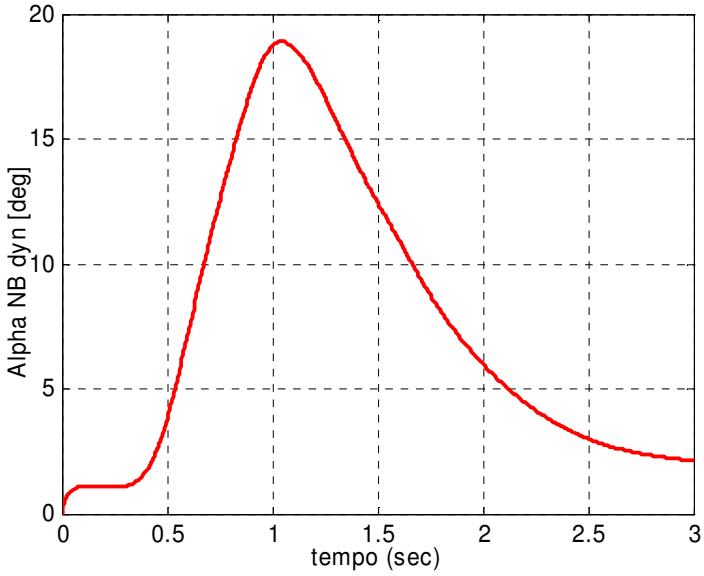

Fig. 6.33 : Alpha Noseboom

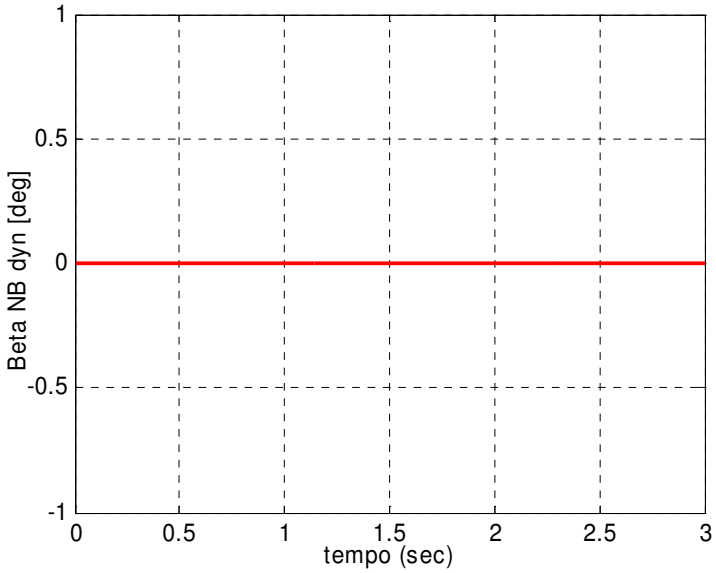

Fig. 6.34 : Beta Noseboom

- RD h=15000 ft, Mach=0.6. In fig.6.35 si riportatono i grafici relativi agli ingressi di simulazione per la manovra specificata. Le figg. 6.36, 6.37, 6.38, 6.39 e 6.40 mostrano le uscite della sonda 1 e del *Noseboom* durante una manovra di timone verticale. Al contrario della manovra precedente l'andamento di λ non segue quello di α poiché in questo caso l'angolo di derapata (β) è diverso da zero.

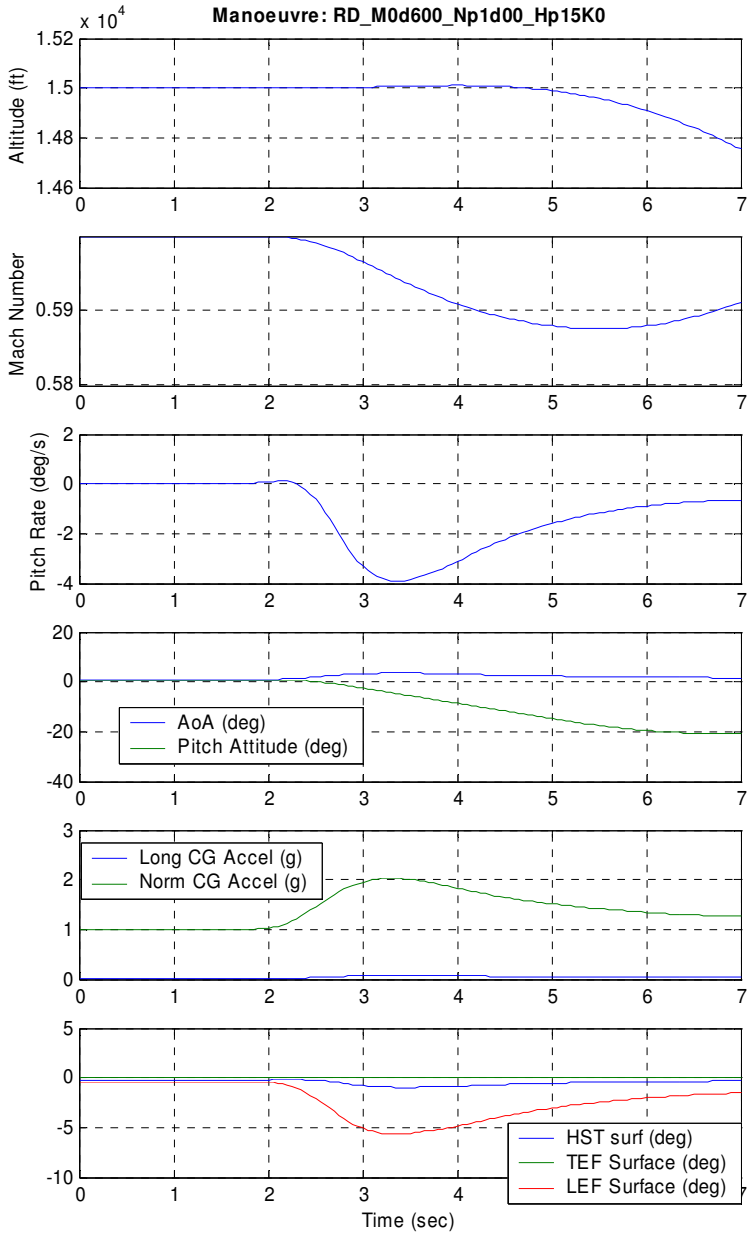

Fig. 6.35 : Ingressi di simulazione

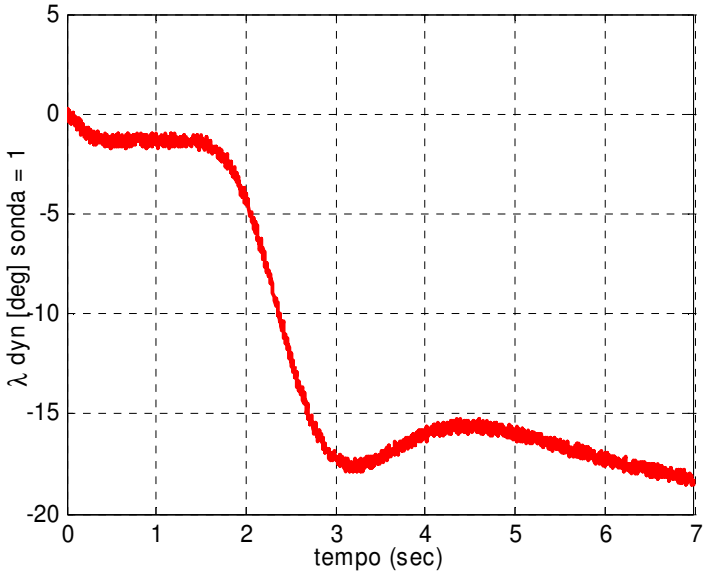

Fig. 6.36 : Lambda_1

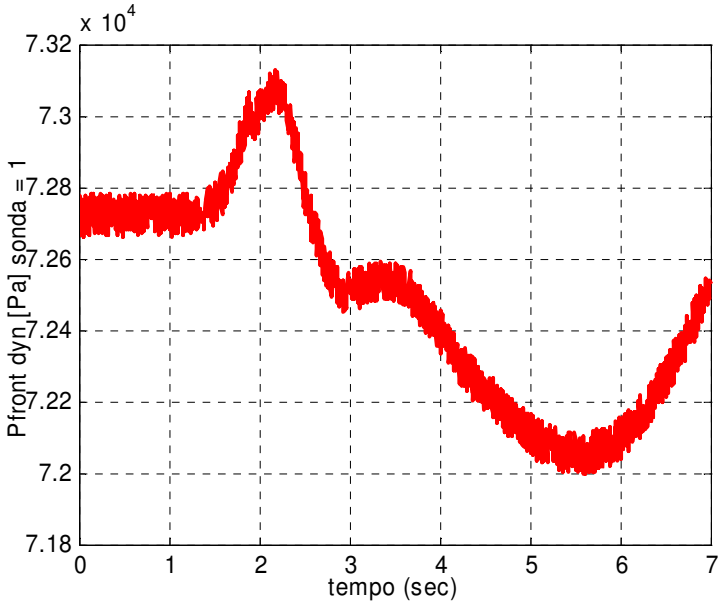

Fig. 6.37 : Pfront_1

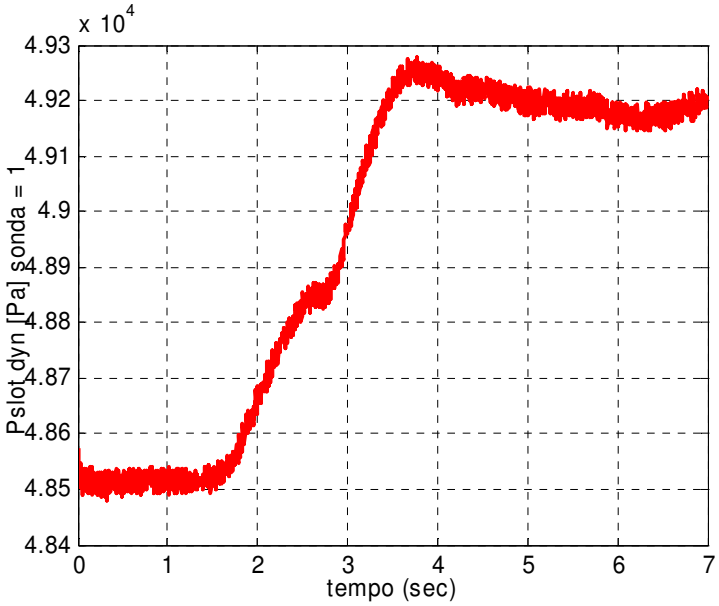

Fig. 6.38 : Pslot_1

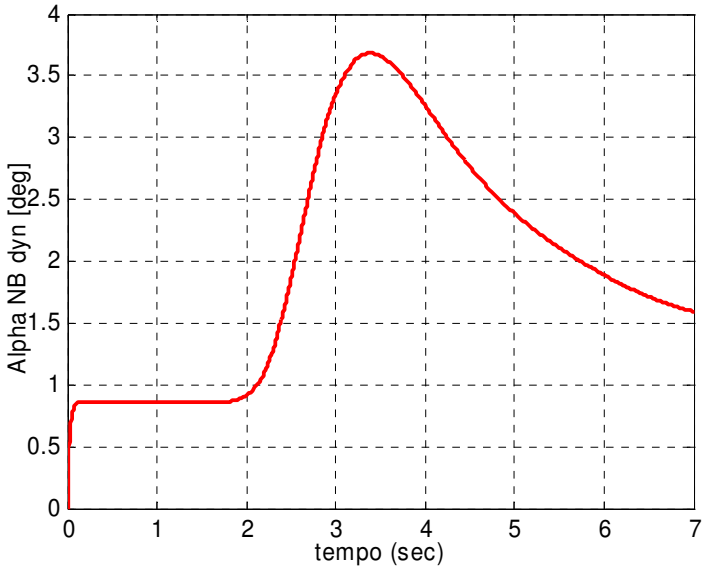

Fig. 6.39 : Alpha Noseboom

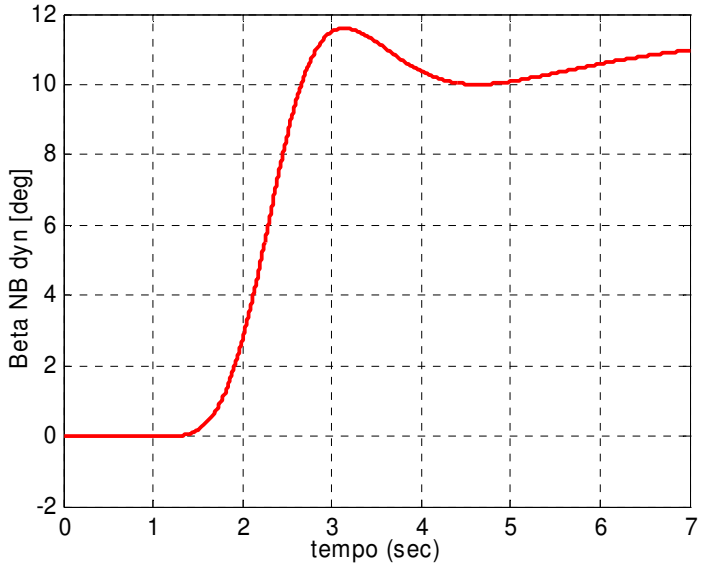

Fig. 6.40 : Beta Noseboom

- RP h=30000 ft, Mach=0.9. In fig.6.41 si riportatono i grafici relativi agli ingressi di simulazione per la manovra specificata.. Le figg. 6.42, 6.43, 6.44, 6.45 e 6.46 mostrano le uscite della sonda 1 e del *Noseboom* durante una manovra di rollio.

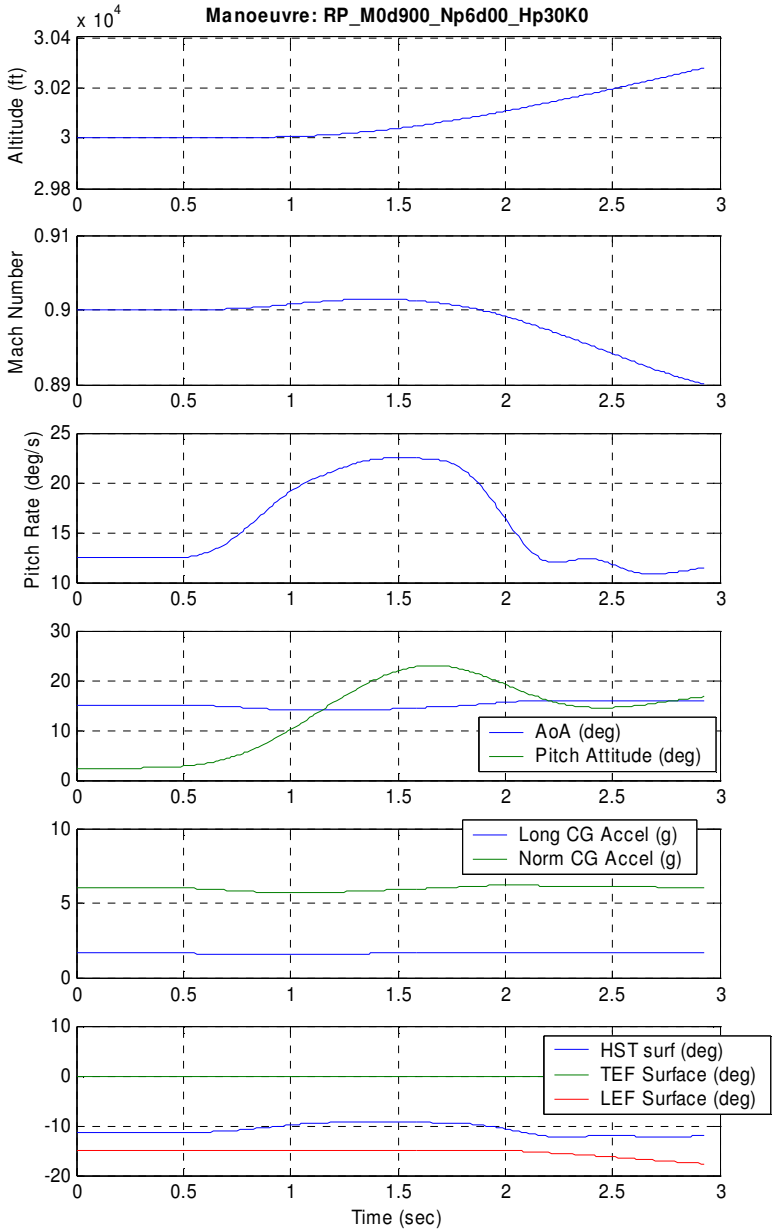

Fig. 6.41 : Ingressi di simulazione

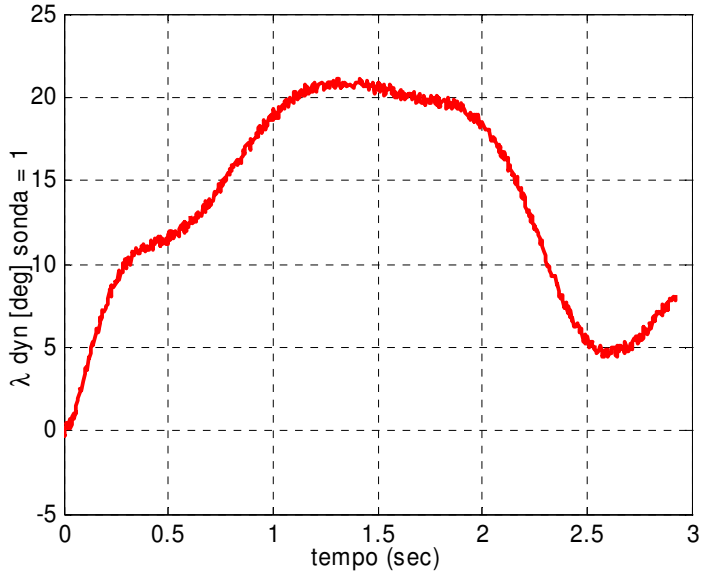

Fig. 6.42 : Lambda_1

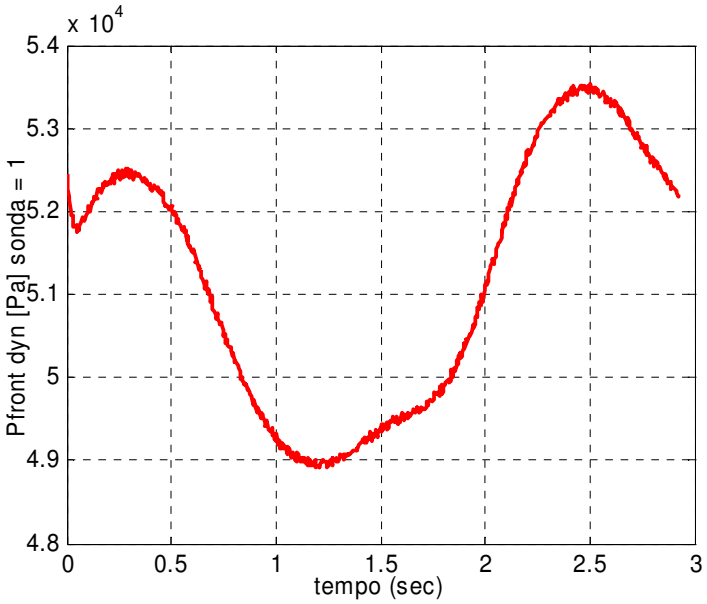

Fig. 6.43 : Pfront_1

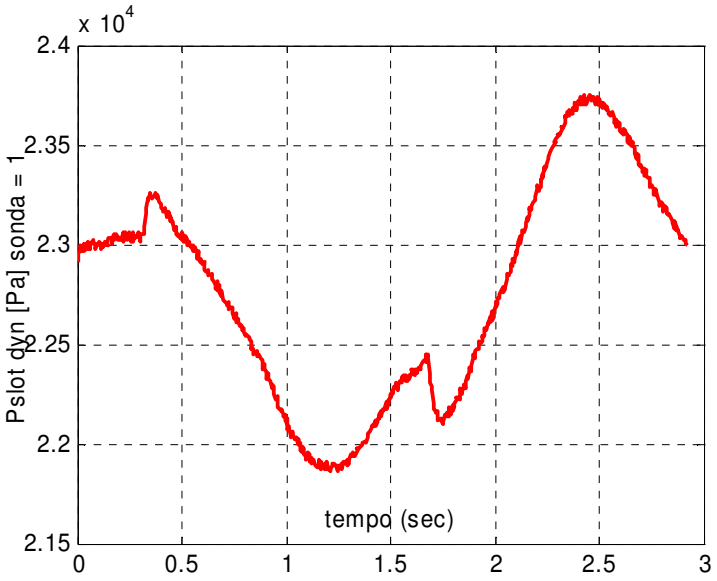

Fig. 6.44 : Pslot_1

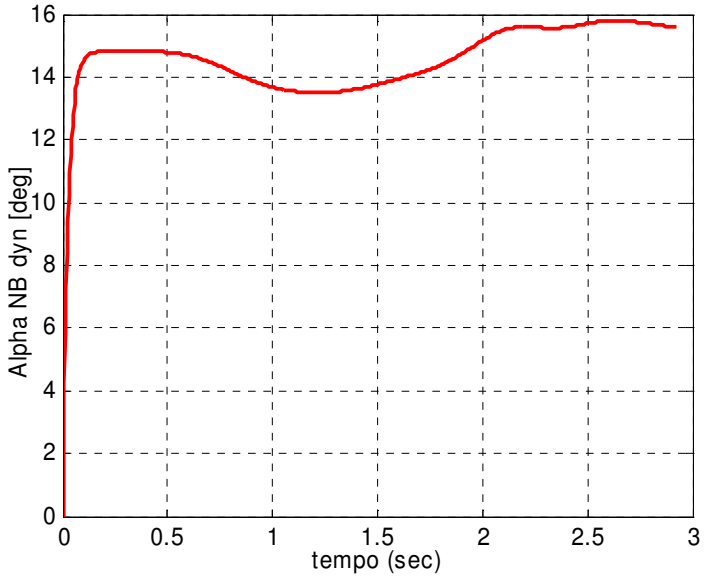

Fig. 6.45 : Alpha Noseboom

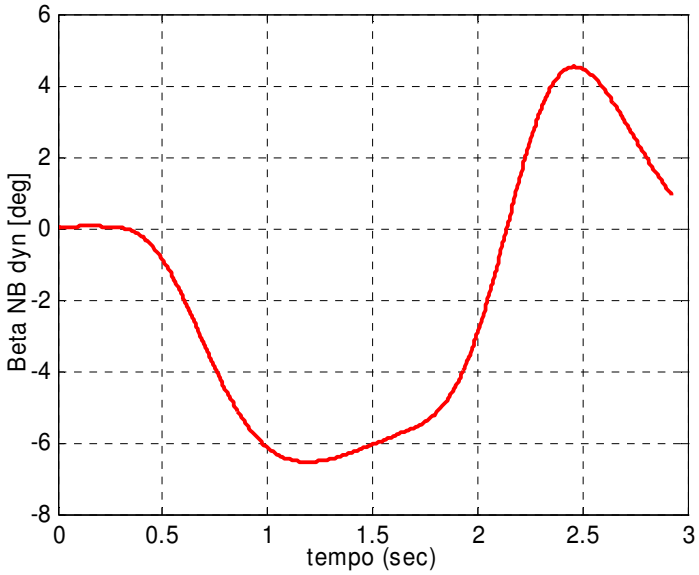

Fig. 6.46 : Beta Noseboom

6.5 Test in Real-Time

6.5.1 Problematiche relative al Real-Time

Il *software* realizzato nella presente tesi per la procedura di simulazione ha come finalità ultima quella di essere utilizzato su *Iron-Bird* per testare tutto l'insieme dell'*hardware* e del *software* dell'intero *Flight Control System* (FCS). Per questo motivo il *software* di simulazione deve poter funzionare correttamente in tempo reale (*Real-Time*).

Lo strumento utilizzato per le simulazioni in tempo reale è un *toolbox* di *Matlab®,* denominato *xPC Target ®*, tramite il quale è stato possibile gestire la compilazione e l'esecuzione del modello di simulazione in tempo reale su un PC, detto *Target*, collegato in rete con il PC , detto *Server*, che gestisce il *software Matlab-Simulink®*.

La simulazione del software realizzato è stata effettuata su un singolo *Target*. La procedura standard fornita da *xPC Target ®* per la simulazione in tempo reale segue i seguenti passi:

- Scelta dell'ambiente di simulazione. Dal menù *Simulation* selezionare *Simulation Parameters* e scegliere la configurazione (pulsante Browse) con *xPC Target.*

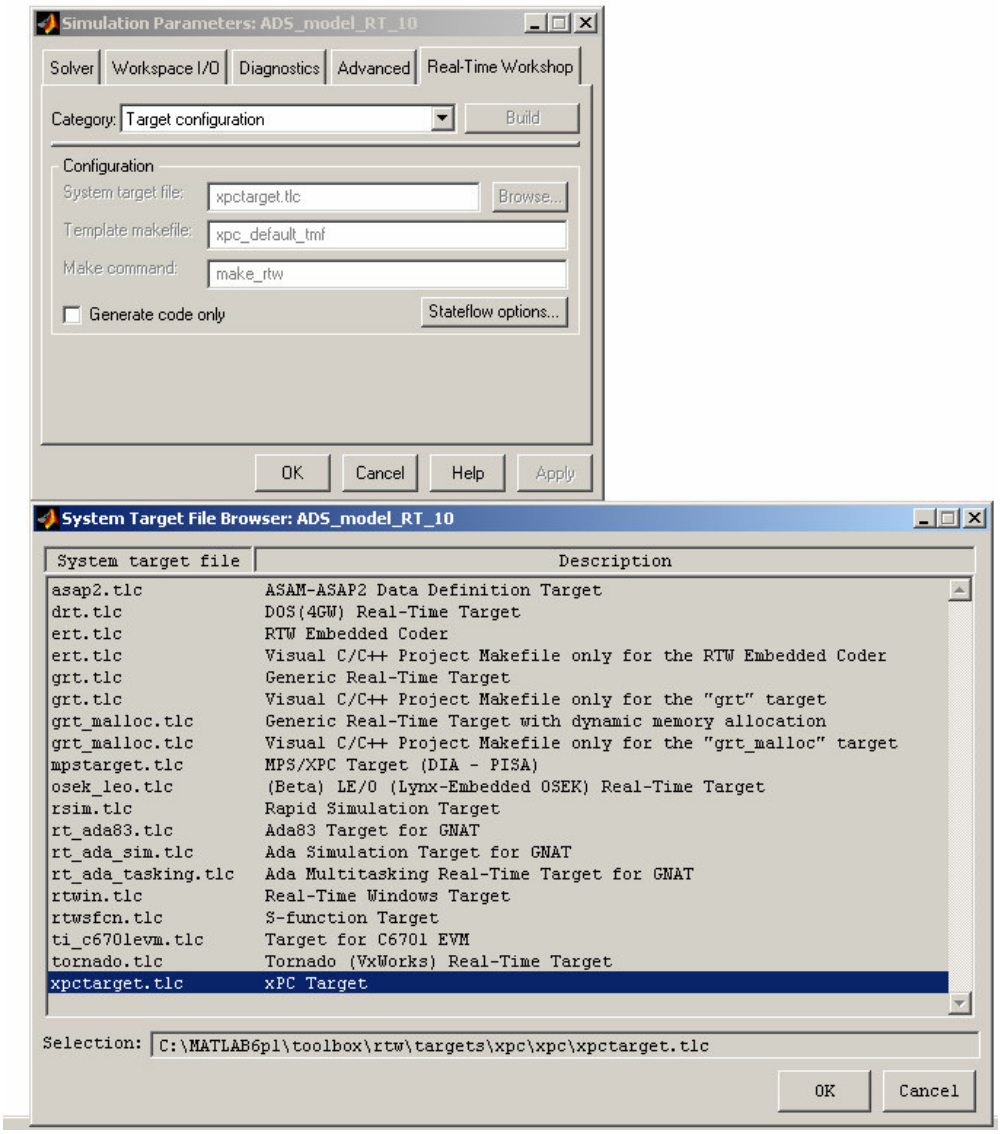

Fig. 6.47 : Passo 1 : scelta dell'ambiente di simulazione

- Generazione del codice eseguibile. Il comando *Build* genera il *file* eseguibile e l'oggetto denominato "tg", che serve da riferimento per *Matlab®*.

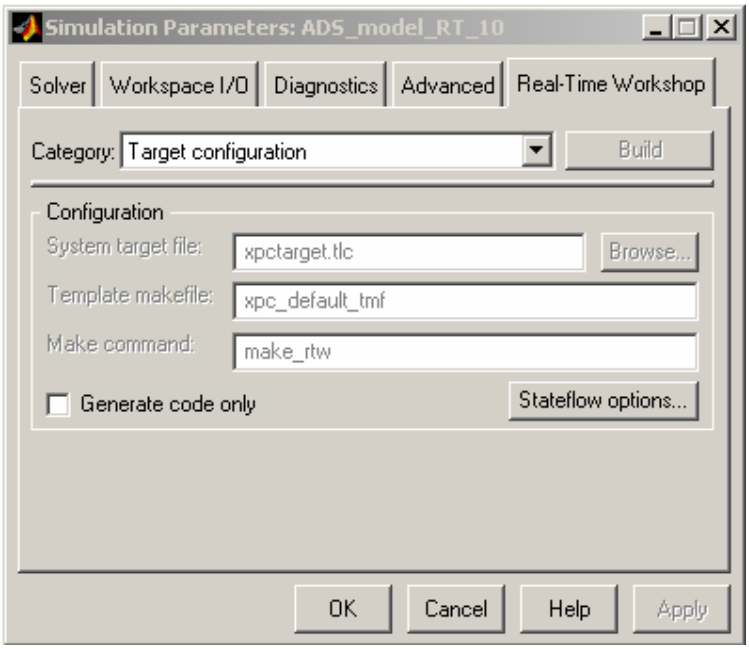

Fig. 6.48 : Passo 2 : generazione del codice eseguibile

A questo punto in modello viene compilato sul *Server* e da qui caricato, tramite collegamento *Ethernet*, su un *Target.*

La scelta del particolare *Target* che si vuole utilizzare, oltre che alcune opzioni riguardanti il compilatore, si può impostare con il comando "*xpcsetup*" (fig. 6.49) dalla *Command Window* di *Matlab®*. Con il comandi "+tg" e "-tg" dalla *Command Window di Matlab®* si fa partire e si ferma la simulazione. Per nozioni più approfondite sui comandi eseguibili in *Matlab®* per la gestione della simulazione si rimanda il lettore al manuale di *xPC Target ®*.

La compilazione del modello di simulazione ha presentato in partenza alcuni problemi a causa delle elevate dimensioni del modello. Il problema è stato risolto settando il parametro relativo ai limiti di memoria del compilatore (/Zm, Specify Memory Allocation Limit) dal volore di *default* di 100 ad un valore pari a 200. Il parametro in questione è stato variato all'interno del *file xpc_vc.tmf* alla riga 122.

Il *file xpc_vc.tmf* si trova nella cartella *xpc* che può essere individuata dal seguente percorso *C:\matlab\toolbox\rtw\targets\xpc\xpc.*

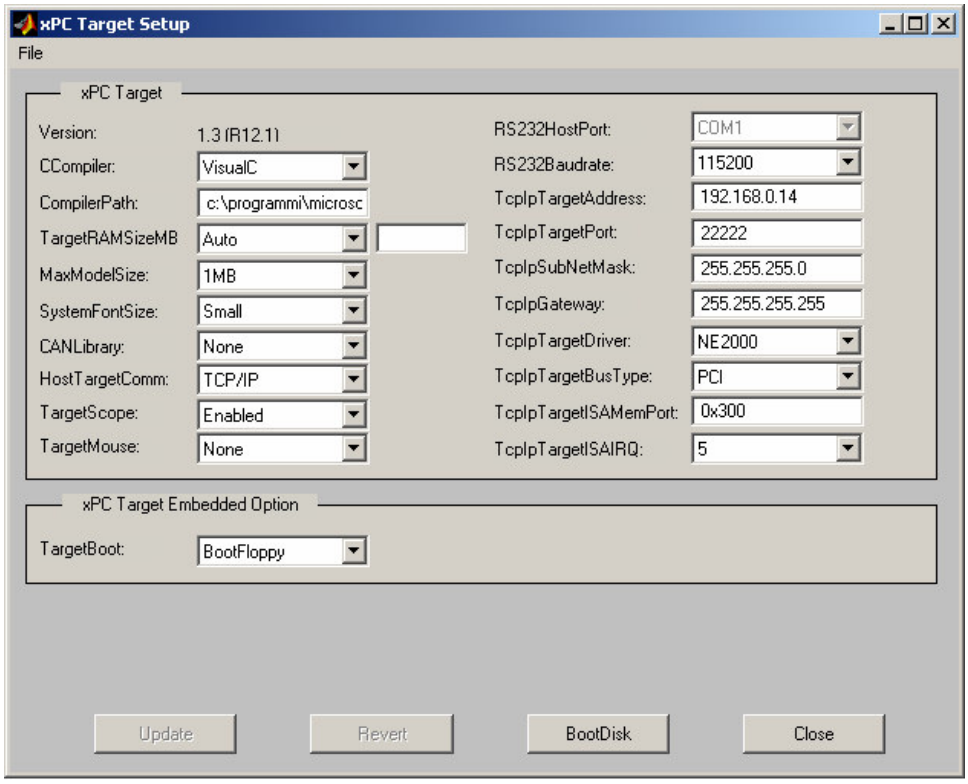

Fig. 6.49 : xpcsetup

Sempre a causa delle elevate dimensioni del modello si è presentato anche un altro tipo di problema legato a due fattori. Il primo riguarda il tempo di compilazione del modello che è stimabile in 30/40 minuti avendo utlizzato un un *Target* di prestazioni:

- Processore Intel Pentium 4 a 2.1 GHz.

e su un *Server* di prestazioni:

- Processore Intel Pentium 4 a 1.7 GHz con 650 MB di RAM.

.Il secondo è legato al fatto che una volta chiuso e riaperto il software *Matlab®,* un modello che è già stato compilato sul *Target* ha bisogno di una nuova compilazione per poter girare in tempo reale. Questo tipo di difficoltà è stata risolta quando sono stati scoperti i comandi da scrivere sulla *Command Window* di *Matlab®* che permettono di caricare un *file* già compilato sul *Target* evitando così una nuova compilazione. I comandi in questione da digitare in sequenza sono, "*tg = xpc*" (invio) e *"load(tg,'ADS_model_RT')"* (invio) dove *ADS_model_RT* è il nome del modello da caricare.

6.5.2 Test

Ricapitolando, per effettuare un test in tempo reale occorre settare correttamente le voci del *xpcsetup* (fig. 6.49), compilare il modello nell'ambiente di simulazione *xPC Target* predisposto attraverso il menù *Simulation*/*Simulation Parameters,* quindi con il comando "+tg" e "-tg" dalla *Command Window di Matlab®* si fa partire e si ferma la simulazione. Altrimenti se il modello è già stato compilato occorre caricare il modello attraverso la digitazione nella *Command Window di Matlab®* dei comandi "*tg = xpc*" (invio) e *"load(tg,'ADS_model_RT')"* (invio) dove *ADS_model_RT* è il nome del modello da caricare. Dopodiché con i comandi "+tg" e "-tg" si fa partire e si ferma la simulazione.

Le prime prove in tempo reale sono state effettuate per testare la corretta funzionalità del pannello di gestione delle avarie realizzato con il software LabVIEW®. Facendo girare il modello sul Target ed il pannello di controllo sul

Server insieme al software Matlab® aperto, si è osservato che per un corretto funzionamento del pannello occorre settare nel modello di simulazione la voce Sampletime del blocchetto UDP Receive Binary e del blocchetto UDP Send Binary mostrato in figg. 5.7 e 5.9 a un valore non superiore a 1/30. In caso contrario il pannello di gestione delle avarie termina di funzionare correttamente appena si cerca di modificare un parametro di una failure (vedi fig. 5.20). Un Sampletime minore o uguale di 1/30 introduce però un ritardo nel tempo di visualizzazione in LabVIEW® rispetto al tempo di simulazione del modello sul Target. Questo comportamento potrebbe essere determinato dal fatto che il pannello di controllo non riesce ad aggiornarsi a causa dell'elevato numero di segnali che deve gestire.

La seconda tipologia di prove ha riguardato il salvataggio delle uscite di simulazione in tempo reale della sonda 1 per effettuare un confronto con gli stessi risultati di una simulazione non un tempo reale. Queste prove hanno riguardato il caso significativo di α variabile da 0^o a 25^o dopo un ritardo 2.5 secondi, β=0, M=0.4, h=13200 m per una simulazione della durata di 30 secondi. In figg. 6.50, 6.51 e 6.52 vengono riportate rispettivamente le misure di λ, P_{front} e P_{slot2} per i due tipi di simulazione sopra citati. Dalle figg. 6.50, 6.51 e 6.52 si osserva che non ci sono differenze tra le due tipologie di prova.

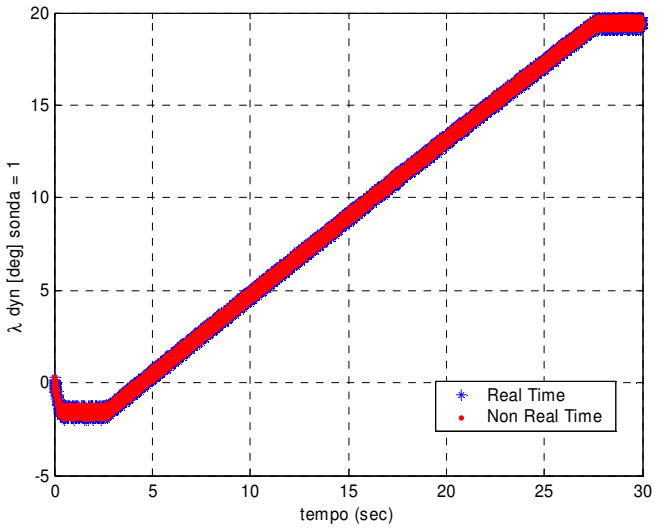

Fig. 6.50 : Lambda_1

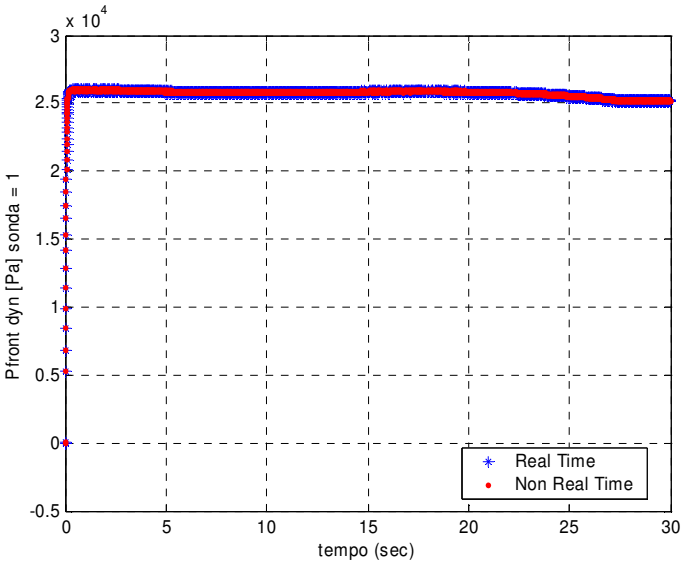

Fig. 6.51 : Pfront_1

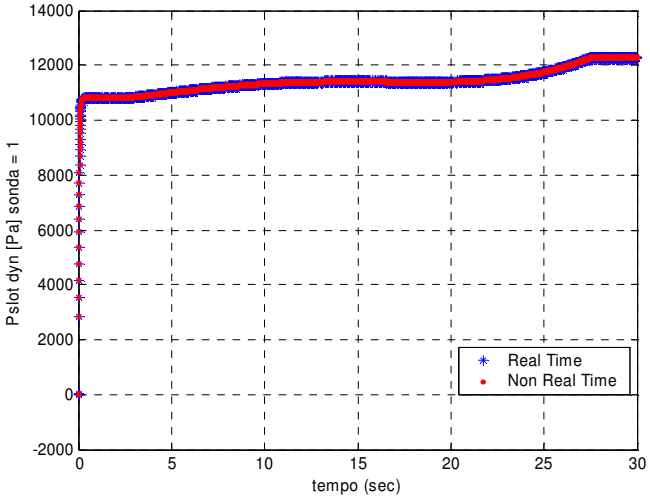

Fig. 6.52 : Pslot_1

7. Conclusioni e sviluppi futuri

Il modello di simulazione realizzato consente di simulare le uscite dei sensori Dati Aria in tutte le possibili condizioni di volo dell'inviluppo di volo di interesse, cosa che precedentemente a questa tesi non era possibile.

Lo studio del funzionamento delle sonde multifunzione ha permesso poi di individuare una serie di avarie che possono interessare il sistema. Questa attività si è concretizzata nella realizzazione di un pannello di controllo in ambiente *LabVIEW®* per la gestione delle *failure* (attivazione e impostazione dei parametri delle avarie). Questo pannello, intergrato con il *software* di simulazione del sistema Dati Aria consente di attivare le avarie, impostando i relativi parametri, in qualunque momento del volo simulato, verificando gli effetti delle avarie stesse sui segnali generati dalle sonde.

Infine sono state effettuate prove di simulazione in tempo reale in ambiente *Matlab-Simulink-xPC Target ®* che hanno consentito di verificare il corretto funzionamento del *software* di simulazione e di validare la procedura completa del sistema Dati Aria per i succesivi test su *Iron-Bird* dell'intero *Flight Control System (FCS).*

BIBLIOGRAFIA

[1] **F.Cervia, E. Denti, R. Galatolo, F. Schettini.** "*Sensori dei Dati – Aria: Ricostruzione degli angoli* α *e* β *in condizioni stazionarie dalle curve di calibrazione in galleria del vento*"-Documento del Dipartimento di Ing. Aerospaziale DDIA 2001-10, Università di Pisa, Giugno 2001.

[2] **F.Cervia, E. Denti, R. Galatolo, F. Schettini.** "*Ricostruzione della pressione statica ambiente e del Mach di volo in condizioni stazionarie a partire dai segnali di pressione forniti dalle ADP* "-Documento del Dipartimento di Ing. Aerospaziale DDIA 2002-16, Università di Pisa, Settembre 2002.

[3] **F.Cervia, E. Denti, R. Galatolo, F. Schettini.** "*Procedure e modelli per la simulazione dei dati – aria*"-Documento del Dipartimento di Ing. Aerospaziale DDIA 2002-15, Università di Pisa, Luglio 2002.

[4] **A.Calia, R. Galatolo, E. Denti, F.Cervia, F. Schettini**. "*Sviluppo di modelli e procedure per la simulazione ed elaborazione dei Dati Aria di un moderno velivolo Fly-by-Wire*"- Tesi di laurea del Dipartimento di Ing. Aerospaziale DDIA 2002/2003, Università di Pisa, Luglio 2003.

[5] **F.Cervia, E. Denti, R. Galatolo, F. Schettini.** "*Modellizzazione dei segnali generati da sonde coniche per la misura dei dati - aria*"-Documento del Dipartimento di Ing. Aerospaziale DDIA 2001-24, Università di Pisa, Ottobre 2001.

[6] **B.Cinque, D.Marchetti**. "*Memo ITV/ARAT/029/03, Air Data Sensor characteristics (enhancement 1)*"- Documento di AerMacchi 2003-24, Luglio 2003.

APPENDICE A: file *Matlab®*

Si riportano di seguito i file *Dati_Galleria_Array.m, LUT_V_generator.m e LUT_MinQuad.m,* descritti rispettivamente in § 5.1.1, 5.1.2 e 5.1.3, che hanno permesso di ampliare l'inviluppo di volo come spiegato nel Capitolo 3.

Dati_Galleria_Array.m

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%                                                                                 %
% NOME FILE     : Dati_Galleria_Array.m                                           %
%                                                                                 %
% PROGETTO      : Fly-By-Wire (Proff. Galatolo / Denti )                          %
%                                                                                 %
% AUTORE        : Rosignoli Nicola                                                %
%                 Cervia Fabio                                                    %
%                 Schettini Francesco                                             %
%                                                                                 %
% RIFERIMENTI   : "Sviluppo di modelli e metodi per la                            %
%                  simulazione delle avarie del sistema Dati                      %
%                  Aria di un moderno velivolo Fly-by Wire"                       %
%                                                                                 %
%                                                                                 %
% ULTIMA REVISIONE  : 7/02/2004                                                   %
%                                                                                 %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

   File relativo all'ampliamento dell'invilupo di volo in Alpha e Beta

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%


clc,clear all,close all

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% CARICAMENTO dei DATI di galleria sotto forma di ARRAY STRUTTURATO %
% per l'approssimazione ai Minimi Quadrati nella cinfigurazione     %
% di volo di Crociera.                                              %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

%   CRUISE   %

% (1) Caricamento dati di galleria per Alpha variabile e Beta costante

config='Cruise';
sonda=['1'; '2'; '3'; '4'];
```

```
   vett_Mach_av=['020'; '040'; '060'; '070'; '080'; '085';'090';'095'];
   vett_beta=[-10;-5;0;5;10];

   % Valori di beta per i quali si hanno i valori di galleria
   string(1).beta='10n';
   string(2).beta='5n';
   string(3).beta='0';
   string(4).beta='5';
   string(5).beta='10';

   % Carica dalla cartella:
   cd Dati_simulazione

   for  k=1:4                          % al variare del num di sonda
     for i=1:length(vett_Mach_av)      % al variare del Mach
       for j=1:5                       % al variare di beta

           % --- Per BETA=0 carica anche le Lut avNeg
           if j==3

           stringmat=['LUT_Cruise_av_b'   string(j).beta   '_M'   vett_Mach_av(i,:)
'.mat'];
           %      stringmatNeg=['LUT_Cruise_avNeg_b'      string(j).beta      '_M'
vett_Mach_av(i,:) '.mat'];

            % -- Solo per Mach>0.2 si hanno le Lut avNeg
            if exist(stringmat)==2 % & exist(stringmatNeg)==2

   %           % - Carica le Lut avNeg
   %                     eval(['load  '  'LUT_Cruise_avNeg_b'  string(j).beta  '_M'
vett_Mach_av(i,:)]);
   %           Lut_sonda_b111=eval(strcat('Lut_sonda',num2str(k)));
   %
   %           % Limitazione sulle Lut avNeg : -15<alpha<-6
   %           ii=1;jj=1;
   %           while (Lut_sonda_b111(jj,3) > -6)
   %           jj=jj+1;
   %           end
   %           while (Lut_sonda_b111(ii,3) > -15)
   %           ii=ii+1;
   %           end
   %           Lut_sonda_b11=Lut_sonda_b111(jj:ii,:);
   %
   %           % Inverte le righe delle colonne della matrice
   %           Lut_sonda_b1=flipud(Lut_sonda_b11);

             % - Carica le Lut av
             eval(['load     '      'LUT_Cruise_av_b'      string(j).beta      '_M'
vett_Mach_av(i,:)]);
             Lut_sonda_b2=eval(strcat('Lut_sonda',num2str(k)));
```

```
            % - Concatena le due matrici
            %Lut_sonda_b=vertcat(Lut_sonda_b1,Lut_sonda_b2);
            Lut_sonda_b=Lut_sonda_b2;

            % Limitazione di alpha tra -10 : 25
            Ind_size=size(Lut_sonda_b);
            ii=1;jj=1;
            while (Lut_sonda_b(ii,3) < 25) | (ii==Ind_size(1))
            ii=ii+1;
            end
            while (Lut_sonda_b(jj,3) < -10) | (ii==Ind_size(1))
            jj=jj+1;
            end
            Lut_sonda_b=Lut_sonda_b(jj:ii,:);

   % Array strutturati che danno, per ogni Mach e per ogni beta fissato, i valori
di alpha,beta,lambda,CpL,ML

   eval(['probe' sonda(k,:) '.alpha.M' vett_Mach_av(i,:) '(' num2str(j) ')' '.av'
'=Lut_sonda_b(:,3);'] );
   eval(['probe' sonda(k,:) '.beta.M' vett_Mach_av(i,:) '(' num2str(j) ')' '.av'
'=Lut_sonda_b(:,4);'] );
   eval(['probe' sonda(k,:) '.lambda.M' vett_Mach_av(i,:) '(' num2str(j) ')' '.av'
'=Lut_sonda_b(:,5);'] );
   eval(['probe' sonda(k,:) '.CpL.M' vett_Mach_av(i,:) '(' num2str(j) ')' '.av'
'=Lut_sonda_b(:,10);'] );
   eval(['probe' sonda(k,:) '.ML.M' vett_Mach_av(i,:)   '(' num2str(j) ')' '.av'
'=Lut_sonda_b(:,12);'] );


           % -- Per Mach=0.2 non si hanno le Lut avNeg
           elseif exist(stringmat)==2

            eval(['load      '      'LUT_Cruise_av_b'      string(j).beta      '_M'
vett_Mach_av(i,:)]);
            Lut_sonda_b=eval(strcat('Lut_sonda',num2str(k)));

            % Limitazione di alpha tra -10 : 25
            Ind_size=size(Lut_sonda_b);
            ii=1;jj=1;
            while (Lut_sonda_b(ii,3) < 25) | (ii==Ind_size(1))
            ii=ii+1;
            end
            while (Lut_sonda_b(jj,3) < -10) | (ii==Ind_size(1))
            jj=jj+1;
            end
            Lut_sonda_b=Lut_sonda_b(jj:ii,:);
```

```
   % Array strutturati che danno, per ogni Mach e per ogni beta fissato, i valori
di alpha,beta,lambda,CpL,ML

   eval(['probe' sonda(k,:) '.alpha.M' vett_Mach_av(i,:) '(' num2str(j) ')' '.av'
'=Lut_sonda_b(:,3);'] );
   eval(['probe' sonda(k,:) '.beta.M' vett_Mach_av(i,:) '(' num2str(j) ')' '.av'
'=Lut_sonda_b(:,4);'] );
   eval(['probe' sonda(k,:) '.lambda.M' vett_Mach_av(i,:) '(' num2str(j) ')' '.av'
'=Lut_sonda_b(:,5);'] );
   eval(['probe' sonda(k,:) '.CpL.M' vett_Mach_av(i,:) '(' num2str(j) ')' '.av'
'=Lut_sonda_b(:,10);'] );
   eval(['probe' sonda(k,:) '.ML.M' vett_Mach_av(i,:)   '(' num2str(j) ')' '.av'
'=Lut_sonda_b(:,12);'] );

            else

   eval(['probe' sonda(k,:) '.alpha.M' vett_Mach_av(i,:) '(' num2str(j) ')' '.av'
'=[];'] );
   eval(['probe' sonda(k,:) '.beta.M' vett_Mach_av(i,:) '(' num2str(j) ')' '.av'
'=[];'] );
   eval(['probe' sonda(k,:) '.lambda.M' vett_Mach_av(i,:) '(' num2str(j) ')' '.av'
'=[];'] );
   eval(['probe' sonda(k,:) '.CpL.M' vett_Mach_av(i,:) '(' num2str(j) ')' '.av'
'=[];'] );
   eval(['probe' sonda(k,:) '.ML.M' vett_Mach_av(i,:)   '(' num2str(j) ')' '.av'
'=[];'] );

            end

          % --- Per BETA diverso da zero non si hanno le Lut avNeg
          else

          stringmat=['LUT_Cruise_av_b'   string(j).beta   '_M'   vett_Mach_av(i,:)
'.mat'];

             if exist(stringmat)==2
             eval(['load     '     'LUT_Cruise_av_b'     string(j).beta     '_M'
vett_Mach_av(i,:)]);

             % Limitazione di alpha tra -10 : 25
             Lut_sonda_b=eval(strcat('Lut_sonda',num2str(k)));
             Ind_size=size(Lut_sonda_b);
             ii=1;jj=1;
             while (Lut_sonda_b(ii,3) < 25) | (ii==Ind_size(1))
             ii=ii+1;
             end
             while (Lut_sonda_b(jj,3) < -10) | (ii==Ind_size(1))
             jj=jj+1;
             end
             Lut_sonda_b=Lut_sonda_b(jj:ii,:);
```

```
   % Array strutturati che danno, per ogni Mach e per ogni beta fissato, i valori
di alpha,beta,lambda,CpL,ML

   eval(['probe' sonda(k,:) '.alpha.M' vett_Mach_av(i,:) '(' num2str(j) ')' '.av'
'=Lut_sonda_b(:,3);'] );
   eval(['probe' sonda(k,:) '.beta.M' vett_Mach_av(i,:) '(' num2str(j) ')' '.av'
'=Lut_sonda_b(:,4);'] );
   eval(['probe' sonda(k,:) '.lambda.M' vett_Mach_av(i,:) '(' num2str(j) ')' '.av'
'=Lut_sonda_b(:,5);'] );
   eval(['probe' sonda(k,:) '.CpL.M' vett_Mach_av(i,:) '(' num2str(j) ')' '.av'
'=Lut_sonda_b(:,10);'] );
   eval(['probe' sonda(k,:) '.ML.M' vett_Mach_av(i,:)   '(' num2str(j) ')' '.av'
'=Lut_sonda_b(:,12);'] );

             else
   eval(['probe' sonda(k,:) '.alpha.M' vett_Mach_av(i,:) '(' num2str(j) ')' '.av'
'=[];'] );
   eval(['probe' sonda(k,:) '.beta.M' vett_Mach_av(i,:) '(' num2str(j) ')' '.av'
'=[];'] );
   eval(['probe' sonda(k,:) '.lambda.M' vett_Mach_av(i,:) '(' num2str(j) ')' '.av'
'=[];'] );
   eval(['probe' sonda(k,:) '.CpL.M' vett_Mach_av(i,:) '(' num2str(j) ')' '.av'
'=[];'] );
   eval(['probe' sonda(k,:) '.ML.M' vett_Mach_av(i,:)   '(' num2str(j) ')' '.av'
'=[];'] );

             end
          end
       end
     end
   end

   % (2) Caricamento dati di galleria per Beta variabile e Alpha costante

   vett_Mach_bv=['020'; '040'];

   string(1).alpha='0';
   string(2).alpha='10';
   string(3).alpha='15'; % contributi relativi al Nose-drop (in manovra)
   string(4).alpha='20'; % contributi relativi al Nose-drop (in manovra)
   string(5).alpha='25'; % contributi relativi al Nose-drop (in manovra)

   % Per le sonde 1,2,3,4.
   for  k=1:4       % al variare del num di sonda
     for i=1:2      % al variare del Mach
       for j=1:5    % al variare di Alpha

           stringmat=['LUT_Cruise_bv_a'   string(j).alpha   '_M'   vett_Mach_bv(i,:)
'.mat'];
```

```
          if exist(stringmat)==2
             eval(['load     '     'LUT_Cruise_bv_a'     string(j).alpha     '_M'
vett_Mach_bv(i,:)]);

             % AAA: Per la sonda 1 si prendono i valori della sonda 4
             if k==1

             Lut_sonda_4=eval(strcat('Lut_sonda',num2str(4)));

             % Limitazione di beta tra -20 : 20 per Alpha=10,15,20,25
             if j>=2
             iii=1;jjj=1;
             while (Lut_sonda_4(iii,4) > 20)
             iii=iii+1;
             end
             while (Lut_sonda_4(jjj,4) > -20)
             jjj=jjj+1;
             end
             Lut_sonda_4=Lut_sonda_4(iii:jjj,:);

             else
             Lut_sonda_4=Lut_sonda_4;
             end

             % Assegnazione
             Lut_sonda_4new=Lut_sonda_4(:,5:12);

             % Inverte quest'ultima matrice
             Lut_sonda_1new=flipud(Lut_sonda_4new);

             % Concatena
             Lut_sonda_a=horzcat(Lut_sonda_4(:,1),Lut_sonda_4(:,2),...
Lut_sonda_4(:,3),Lut_sonda_4(:,4),Lut_sonda_1new);

             else % per le sonde 2,3,4

             Lut_sonda_a=eval(strcat('Lut_sonda',num2str(k)));
             Ind_size=size(Lut_sonda_a);

             % Limitazione di beta tra -20 : 20 per Alpha=10,15,20,25
             if j>=2
             ii=1;jj=1;
             while (Lut_sonda_a(ii,4) > 20)
             ii=ii+1;
             end
             while (Lut_sonda_a(jj,4) > -20)
             jj=jj+1;
             end
```

```
              Lut_sonda_a=Lut_sonda_a(ii:jj,:);

              else
              Lut_sonda_a=Lut_sonda_a;
              end

              end

   eval(['probe' sonda(k,:) '.alpha.M' vett_Mach_bv(i,:)  '(' num2str(j) ')' '.bv'
'=Lut_sonda_a(:,3);'] );
   eval(['probe' sonda(k,:) '.beta.M' vett_Mach_bv(i,:)  '(' num2str(j) ')' '.bv'
'=Lut_sonda_a(:,4);'] );
   eval(['probe' sonda(k,:) '.lambda.M' vett_Mach_bv(i,:) '(' num2str(j) ')' '.bv'
'=Lut_sonda_a(:,5);'] );
   eval(['probe' sonda(k,:) '.CpL.M' vett_Mach_bv(i,:) '(' num2str(j) ')' '.bv'
'=Lut_sonda_a(:,10);'] );
   eval(['probe' sonda(k,:) '.ML.M' vett_Mach_bv(i,:)  '(' num2str(j) ')' '.bv'
'=Lut_sonda_a(:,12);'] );

              else

   eval(['probe' sonda(k,:) '.alpha.M' vett_Mach_bv(i,:) '(' num2str(j) ')' '.bv'
'=[];'] );
   eval(['probe' sonda(k,:) '.beta.M' vett_Mach_bv(i,:) '(' num2str(j) ')' '.bv'
'=[];'] );
   eval(['probe' sonda(k,:) '.lambda.M' vett_Mach_bv(i,:) '(' num2str(j) ')' '.bv'
'=[];'] );
   eval(['probe' sonda(k,:) '.CpL.M' vett_Mach_bv(i,:) '(' num2str(j) ')' '.bv'
'=[];'] );
   eval(['probe' sonda(k,:) '.ML.M' vett_Mach_bv(i,:)  '(' num2str(j) ')' '.bv'
'=[];'] );

            end  % if exist
         end     % for Alpha
      end        % for Mach
   end           % for sonda
   cd ..

   % SALVATAGGIO LUT
   disp('  ');
   res=input(' Save Array Strutt. ? (y/n) ','s');
    if res=='y'

   s_str1=num2str(1);s_str2=num2str(2);
   s_str3=num2str(3);s_str4=num2str(4);

   name_file=strcat('Array_Strutt_DATI_GALL_Cruise');

   name_var1=strcat('probe',s_str1,'');
   name_var2=strcat('probe',s_str2,'');
```

```
   name_var3=strcat('probe',s_str3,'');
   name_var4=strcat('probe',s_str4,'');

   cd Dati_Simulazione  % Salva nella cartella "LUT_Adm_input"

   save(name_file,name_var1,name_var2,name_var3,name_var4)

   cd ..

   disp('    ');
   disp(' Gli Array Strutt. sono salvati nel file "Array_Strutt_DATI_GALL_Cruise"
');
   disp('  nella cartella Dati_Simulazione.');
   disp('     ');

    else
    end  % fine if decisionale sul salvatagio LUT

   disp('  ');
   disp(' Fine');
```

LUT_V_MinQuad.m

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%                                                                              %
% NOME FILE     : LUT_V_MinQuad.m                                              %
%                                                                              %
% PROGETTO      : Fly-By-Wire (Proff. Galatolo / Denti )                       %
%                                                                              %
% AUTORE        : Rosignoli Nicola                                             %
%                 Cervia Fabio                                                 %
%                 Schettini Francesco                                          %
%                                                                              %
% RIFERIMENTI   : "Sviluppo di modelli e metodi per la                         %
%                  simulazione delle avarie del sistema Dati                   %
%                  Aria di un moderno velivolo Fly-by Wire"                    %
%                                                                              %
%                                                                              %
% ULTIMA REVISIONE  : 7/02/2004                                                %
%                                                                              %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

    File relativo all'ampliamento dell'invilupo di volo in Alpha e Beta

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
clc,clear all,close all

% Caricamento dati di Galleria sotto forma dell''ARRAY STRUTTURATO
% ----------------------------------------------------------------
cd Dati_Simulazione
```

```
string0mat=['Array_Strutt_DATI_GALL_Cruise.mat'];

  if exist(string0mat)==2

  eval(['load ' 'Array_Strutt_DATI_GALL_Cruise']);

  else
  end

cd ..

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Procedura per la scalatura dei dati di Galleria a BETA variabile aMach>0.4,%
% con consenguente generazione delle LUT_V_Cruise_bv                          %
% per la successiva Approssimazione ai MINIMI QUADRATI.                       %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

disp('  ')
disp('  Procedura per la scalatura dei dati di Galleria a BETA variabile a
Mach>0.4,')
disp('             con consenguente generazione delle LUT_V_Cruise_bv ')
disp('            per la successiva Approssimazione ai MINIMI QUADRATI.')
disp('  ')
disp(' - Inizio procedura iterativa - ')
n_it_max=input(' Scegliere il numero di iterazioni: ');
disp('  ')
disp(' Attendere.....')
disp('  ')

Mach=[0.2;0.4;0.6;0.7;0.8;0.85;0.9;0.95];

% Assegnazione utile per ridurre il num.di righe del file
str1='alpha';str2='beta';str3='lambda';str4='CpL';str5='ML';

% Assegnazione iniziale per il ciclo iterativo
for s=1:4
 for u=3:length(Mach) % Mach
eval(['ErrM_lambda_s' num2str(s) '_M0' num2str(Mach(u)*100) '_it0=[0 0 0 0
0];'] );
eval(['ErrM_CpL_s' num2str(s) '_M0' num2str(Mach(u)*100) '_it0=[0 0 0 0 0];']
);
eval(['ErrM_ML_s' num2str(s) '_M0' num2str(Mach(u)*100) '_it0=[0 0 0 0 0];'] );
 end
end

% Iterazione per la traslazione delle strisciate a beta variabile
% --------------------------------------------------------------
n_it=0;
while n_it<=n_it_max
disp('    ')
disp(sprintf(' Iterazione: %8.3f',n_it));
disp(' ..........');
```

```
   % Per ogni Sonda
   % --------------
     for s=1:4
   disp('   ');
   disp(sprintf(' Sonda: %8.3f',s));
   disp(' ------');

   % Solo per i valori di Mach>=0.6
   % ------------------------------
       for u=3:length(Mach)
   disp('  ');
   disp(sprintf(' Mach: %8.3f',Mach(u)));

   % Per Mach>=0.6

               for st=1:5   % Per lambda,CpL,ML
                   if st==1;
                       str=str1; % Alpha
                   elseif st==2
                       str=str2; % Beta
                   elseif st==3
                       str=str3; % lambda
                   elseif st==4
                       str=str4; % CpL
                   elseif st==5
                       str=str5; % ML
                   end

                   if st<=2 % Per Alpha e Beta

   % Genera il vettore per es. alpha_av.sx.M0xx
   eval(['',str,'_av.s' num2str(s) '.M0' num2str(Mach(u)*100) '' ...
                  '=[probe'   num2str(s)   '.',str,'.M0'   num2str(Mach(u)*100)   '('
num2str(1) ')' '.av;'...
                    'probe'   num2str(s)   '.',str,'.M0'   num2str(Mach(u)*100)   '('
num2str(2) ')' '.av;'...
                    'probe'   num2str(s)   '.',str,'.M0'   num2str(Mach(u)*100)   '('
num2str(3) ')' '.av;'...
                    'probe'   num2str(s)   '.',str,'.M0'   num2str(Mach(u)*100)   '('
num2str(4) ')' '.av;'...
                    'probe'   num2str(s)   '.',str,'.M0'   num2str(Mach(u)*100)   '('
num2str(5) ')' '.av];']);

   % Assegnazioni utili per generare le LUT_V_
   eval(['',str,'_bv_a0.s' num2str(s) '.M0' num2str(Mach(u)*100) '' ...
   '=[probe'   num2str(s)   '.',str,'.M0'   num2str(0.4*100)   '('   num2str(1)   ')'
'.bv];']);

   eval(['',str,'_bv_a10.s' num2str(s) '.M0' num2str(Mach(u)*100) '' ...
```

```
   '=[probe'   num2str(s)   '.',str,'.M0'   num2str(0.4*100)   '('   num2str(2)   ')'
'.bv];']);

   eval(['',str,'_bv_a15.s' num2str(s) '.M0' num2str(Mach(u)*100) '' ...
   '=[probe'   num2str(s)   '.',str,'.M0'   num2str(0.4*100)   '('   num2str(3)   ')'
'.bv];']);

   eval(['',str,'_bv_a20.s' num2str(s) '.M0' num2str(Mach(u)*100) '' ...
   '=[probe'   num2str(s)   '.',str,'.M0'   num2str(0.4*100)   '('   num2str(4)   ')'
'.bv];']);

   eval(['',str,'_bv_a25.s' num2str(s) '.M0' num2str(Mach(u)*100) '' ...
   '=[probe'   num2str(s)   '.',str,'.M0'   num2str(0.4*100)   '('   num2str(5)   ')'
'.bv];']);

   % Genera il vettore per es. alpha_bv.sx.M0xx
   eval(['',str,'_bv.s' num2str(s) '.M0' num2str(Mach(u)*100) '' ...
   '=[',str,'_bv_a0.s' num2str(s) '.M0' num2str(Mach(u)*100) ';' ...
     '',str,'_bv_a10.s' num2str(s) '.M0' num2str(Mach(u)*100) ';' ...
     '',str,'_bv_a15.s' num2str(s) '.M0' num2str(Mach(u)*100) ';' ...
     '',str,'_bv_a20.s' num2str(s) '.M0' num2str(Mach(u)*100) ';' ...
     '',str,'_bv_a25.s' num2str(s) '.M0' num2str(Mach(u)*100) '];']);

   % Concatena i vettori av e bv
   % Assegnazione utile per il successivo comando "Griddata" per la
   % stima dell'errore.
   eval(['',str,'.s' num2str(s) '.M0' num2str(Mach(u)*100) ''...
   '=vertcat(',str,'_av.s' num2str(s) '.M0' num2str(Mach(u)*100)...
           ',',str,'_bv.s' num2str(s) '.M0' num2str(Mach(u)*100) ');'] );

                else % Per lambda,CpL,ML

   % VALORI SCALATI per lambda,CpL,ML a bv e Mach>0.4:
   % ------------------------------------------------

   % Genera il vettore per es. lambda_av.sx.M0xx
   eval(['',str,'_av.s' num2str(s) '.M0' num2str(Mach(u)*100) '' ...
                '=[probe'   num2str(s)   '.',str,'.M0'   num2str(Mach(u)*100)   '('
num2str(1) ')' '.av;'...
                   'probe'   num2str(s)   '.',str,'.M0'   num2str(Mach(u)*100)   '('
num2str(2) ')' '.av;'...
                   'probe'   num2str(s)   '.',str,'.M0'   num2str(Mach(u)*100)   '('
num2str(3) ')' '.av;'...
                   'probe'   num2str(s)   '.',str,'.M0'   num2str(Mach(u)*100)   '('
num2str(4) ')' '.av;'...
                   'probe'   num2str(s)   '.',str,'.M0'   num2str(Mach(u)*100)   '('
num2str(5) ')' '.av];']);

   % Assegnazioni utili per generare le LUT_V
   % Le strisciate, per es. probex.lambda.M040.bv ,vengono scalate per
```

```
   % realizzare quelle a Mach>0.4.
   eval(['',str,'_bv_a0.s' num2str(s) '.M0' num2str(Mach(u)*100) '' ...
   '=[ErrM_',str,'_s'  num2str(s)  '_M0'  num2str(Mach(u)*100)  '_it'  num2str(n_it)
'(1)'...
     '+probe'  num2str(s)  '.',str,'.M0'  num2str(0.4*100)  '('  num2str(1)  ')'
'.bv];']);

   eval(['',str,'_bv_a10.s' num2str(s) '.M0' num2str(Mach(u)*100) '' ...
   '=[ErrM_',str,'_s'  num2str(s)  '_M0'  num2str(Mach(u)*100)  '_it'  num2str(n_it)
'(2)'...
     '+probe'  num2str(s)  '.',str,'.M0'  num2str(0.4*100)  '('  num2str(2)  ')'
'.bv];']);

   eval(['',str,'_bv_a15.s' num2str(s) '.M0' num2str(Mach(u)*100) '' ...
   '=[ErrM_',str,'_s'  num2str(s)  '_M0'  num2str(Mach(u)*100)  '_it'  num2str(n_it)
'(3)'...
     '+probe'  num2str(s)  '.',str,'.M0'  num2str(0.4*100)  '('  num2str(3)  ')'
'.bv];']);

   eval(['',str,'_bv_a20.s' num2str(s) '.M0' num2str(Mach(u)*100) '' ...
   '=[ErrM_',str,'_s'  num2str(s)  '_M0'  num2str(Mach(u)*100)  '_it'  num2str(n_it)
'(4)'...
     '+probe'  num2str(s)  '.',str,'.M0'  num2str(0.4*100)  '('  num2str(4)  ')'
'.bv];']);

   eval(['',str,'_bv_a25.s' num2str(s) '.M0' num2str(Mach(u)*100) '' ...
   '=[ErrM_',str,'_s'  num2str(s)  '_M0'  num2str(Mach(u)*100)  '_it'  num2str(n_it)
'(5)'...
     '+probe'  num2str(s)  '.',str,'.M0'  num2str(0.4*100)  '('  num2str(5)  ')'
'.bv];']);

   % Genera il vettore per es. lambda_bv.sx.M0xx
   eval(['',str,'_bv.s' num2str(s) '.M0' num2str(Mach(u)*100) '' ...
   '=[',str,'_bv_a0.s' num2str(s) '.M0' num2str(Mach(u)*100) ';' ...
     '',str,'_bv_a10.s' num2str(s) '.M0' num2str(Mach(u)*100) ';' ...
     '',str,'_bv_a15.s' num2str(s) '.M0' num2str(Mach(u)*100) ';' ...
     '',str,'_bv_a20.s' num2str(s) '.M0' num2str(Mach(u)*100) ';' ...
     '',str,'_bv_a25.s' num2str(s) '.M0' num2str(Mach(u)*100) '];']);

    % Concatena i vettori av e bv
    % Assegnazione utile per il successivo comando "Griddata" per la
    % stima dell'errore.
    eval(['',str,'.s' num2str(s) '.M0' num2str(Mach(u)*100) ''...
   '=vertcat(',str,'_av.s' num2str(s) '.M0' num2str(Mach(u)*100)...
           ',',str,'_bv.s' num2str(s) '.M0' num2str(Mach(u)*100) ');'] );

              end % if str
            end    % for str

   % Calcolo delle quantita medie per traslare le strisciate a bv per Mach>0.4 %
```

```
   % ------------------------------------------------------------------------ %

   % Dominio
   alpha_int=[0.15 10 15 20 24.95];
   beta_int=[-10 -4.98 0 4.95 10];
   LUT_alpha_Cruise=alpha_int;
   LUT_beta_Cruise=beta_int;

   [alphai,betai]=meshgrid(LUT_alpha_Cruise,LUT_beta_Cruise);

              for st=3:5   % Per lambda,CpL,ML
                  if st==3
                      str=str3; % lambda
                  elseif st==4
                      str=str4; % CpL
                  elseif st==5
                      str=str5; % ML
                  end

   %%%%%
   % (1) griddata: av
   % Griddata
   % Genera matrici dove le righe variano con beta
   %                   e le colonne variano con alpha.

   eval(['',str,'Iav_s' num2str(s) '_M0' num2str(Mach(u)*100) ''...
         '=griddata(alpha_av.s'      num2str(s)      '.M0'      num2str(Mach(u)*100)
',beta_av.'...
                     's' num2str(s) '.M0' num2str(Mach(u)*100) ','...
         '',str,'_av.s'  num2str(s)  '.M0'  num2str(Mach(u)*100)  ',alphai,betai);']
);

   %%%%%
   % (2) griddata: bv

   eval(['',str,'Ibv_s' num2str(s) '_M0' num2str(Mach(u)*100) ''...
         '=griddata(alpha_bv.s'      num2str(s)      '.M0'      num2str(Mach(u)*100)
',beta_bv'...
                     '.s' num2str(s) '.M0' num2str(Mach(u)*100) ','...
         '',str,'_bv.s'  num2str(s)  '.M0'  num2str(Mach(u)*100)  ',alphai,betai);']
);

              end

   % Calcolo delle differenze tra le strisciate dei valori di galleria
   % a bv e av nei loro punti di incontro per ogni Mach.
   % Le strisciate a bv per Mach>0.4 vengono prese uguali a quelle a Mach=0.4
   % al passo zero dell'iterazione.

   vett_alpha=[00;10;15;20;25];
```

```
   vett_beta=[10;05;00;05;10];

   for k1=1:length(vett_alpha)
       for k2=1:length(vett_beta)
           if k2<=2 % Per beta neg

               for st=3:5   % Per lambda,CpL,ML
                  if st==3
                      str=str3; % lambda
                  elseif st==4
                      str=str4; % CpL
                  elseif st==5
                      str=str5; % ML
                  end

   % Errore tra i griddata ad av e bv

   eval(['E_',str,'_s'     num2str(s)      '_M0'     num2str(Mach(u)*100)      '_a'
num2str(vett_alpha(k1,:)) ''...
         '_b' num2str(vett_beta(k2,:)) 'n'...
   '=',str,'Iav_s' num2str(s) '_M0' num2str(Mach(u)*100) '(k2,k1)'...
   '-',str,'Ibv_s' num2str(s) '_M0' num2str(Mach(u)*100) '(k2,k1);'] );

               end

          else % per beta pos

              for st=3:5   % Per lambda,CpL,ML
                  if st==3
                      str=str3; % lambda
                  elseif st==4
                      str=str4; % CpL
                  elseif st==5
                      str=str5; % ML
                  end

   eval(['E_',str,'_s'     num2str(s)      '_M0'     num2str(Mach(u)*100)      '_a'
num2str(vett_alpha(k1,:)) ''...
         '_b' num2str(vett_beta(k2,:)) ''...
   '=',str,'Iav_s' num2str(s) '_M0' num2str(Mach(u)*100) '(k2,k1)'...
   '-',str,'Ibv_s' num2str(s) '_M0' num2str(Mach(u)*100) '(k2,k1);'] );

            end % for str
          end    % if beta
       end       % for beta
   end           % for alpha

   % Differenze (o errore) MEDIE al variare di beta per ogni alpha e Mach cost.

   for k1=1:length(vett_alpha)
```

```
       % Mach>0.4

            for st=3:5   % Per lambda,CpL,ML
                  if st==3
                      str=str3; % lambda
                  elseif st==4
                      str=str4; % CpL
                  elseif st==5
                      str=str5; % ML
                  end

   % Diff ad alpha fissato e beta variabile
   eval(['EM',str,'_s'      num2str(s)      '_M0'      num2str(Mach(u)*100)      '_a'
num2str(vett_alpha(k1,:)) ''...
         '=[E_',str,'_s'     num2str(s)     '_M0'     num2str(Mach(u)*100)      '_a'
num2str(vett_alpha(k1,:)) '_b10n '...
         'E_',str,'_s'      num2str(s)      '_M0'      num2str(Mach(u)*100)      '_a'
num2str(vett_alpha(k1,:)) '_b5n '...
         'E_',str,'_s'      num2str(s)      '_M0'      num2str(Mach(u)*100)      '_a'
num2str(vett_alpha(k1,:)) '_b0 '...
         'E_',str,'_s'      num2str(s)      '_M0'      num2str(Mach(u)*100)      '_a'
num2str(vett_alpha(k1,:)) '_b5 '...
         'E_',str,'_s'      num2str(s)      '_M0'      num2str(Mach(u)*100)      '_a'
num2str(vett_alpha(k1,:)) '_b10];'] );

   % Diff media per ogni alpha
   eval(['EM_',str,'_s'     num2str(s)      '_M0'     num2str(Mach(u)*100)      '_a'
num2str(vett_alpha(k1,:)) ''...
         '=mean(EM',str,'_s'    num2str(s)    '_M0'    num2str(Mach(u)*100)    '_a'
num2str(vett_alpha(k1,:)) ');'] );

           end
       end

     end % for MACH
   end % for SONDA

   % Differenza Media su beta al variare del Mach a alpha cost.

    for s=1:4
      for k1=1:length(vett_alpha)
          for st=3:5   % Per lambda,CpL,ML
                  if st==3
                      str=str3; % lambda
                  elseif st==4
                      str=str4; % CpL
                  elseif st==5
                      str=str5; % ML
                  end
```

```
   eval(['ErrM_',str,'_s' num2str(s) '_a' num2str(vett_alpha(k1,:)) ''...
          '=[EM_',str,'_s' num2str(s) '_M060_a' num2str(vett_alpha(k1,:)) ''...
           ' EM_',str,'_s' num2str(s) '_M070_a' num2str(vett_alpha(k1,:)) ''...
           ' EM_',str,'_s' num2str(s) '_M080_a' num2str(vett_alpha(k1,:)) ''...
           ' EM_',str,'_s' num2str(s) '_M085_a' num2str(vett_alpha(k1,:)) ''...
           ' EM_',str,'_s' num2str(s) '_M090_a' num2str(vett_alpha(k1,:)) ''...
           ' EM_',str,'_s' num2str(s) '_M095_a' num2str(vett_alpha(k1,:)) '];'] );
         end
     end
   end % SONDA

   % Errore Medio (sulle le strisciate di valori a beta costante)
   % al variare di alpha, a Mach cost.

    for s=1:4
      for u=3:length(Mach) % Mach
          for st=3:5   % Per lambda,CpL,ML
                  if st==3
                      str=str3; % lambda
                  elseif st==4
                      str=str4; % CpL
                  elseif st==5
                      str=str5; % ML
                  end

   eval(['ErrM_',str,'_s'    num2str(s)    '_M0'    num2str(Mach(u)*100)    '_it'
num2str(n_it+1) ''...
          '=[ErrM_',str,'_s' num2str(s) '_a0(u-2)'...
           ' ErrM_',str,'_s' num2str(s) '_a10(u-2)'...
           ' ErrM_',str,'_s' num2str(s) '_a15(u-2)'...
           ' ErrM_',str,'_s' num2str(s) '_a20(u-2)'...
           ' ErrM_',str,'_s' num2str(s) '_a25(u-2)];'] );


   % Assegnazione finale del ciclo iterativo

   eval(['ErrM_',str,'_s'    num2str(s)    '_M0'    num2str(Mach(u)*100)    '_it'
num2str(n_it+1) ''...
        '=ErrM_',str,'_s'    num2str(s)    '_M0'    num2str(Mach(u)*100)    '_it'
num2str(n_it) '+'...
         'ErrM_',str,'_s'    num2str(s)    '_M0'    num2str(Mach(u)*100)    '_it'
num2str(n_it+1) ';'] );
        end
    end
   end

   n_it=n_it+1;
   end % while
```

```
   % Salvataggio di tutte le variabili del file
   disp('  ');
   disp(' Salvataggio di tutte le variabili del file');
   disp(' ----------------------------------------');

   save LUT_V_Generetor

   %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
   % Generazione e salvatagggio LUT_V_Cruise_bv %
   %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

   res=input(' Save LUT_V ? (y/n) ','s');
   if res=='y'

      for u=3:length(Mach)
          Mach_str=num2str(Mach(u)*100);

          for k1=1:length(vett_alpha)
              vett_alpha_str=num2str(vett_alpha(k1,:));

              for s=1:4

   eval(['Lut_sonda' num2str(s) ''...
   '=[  Mach(u)*ones(size(alpha_bv_a'  num2str(vett_alpha(k1,:))  '.s'  num2str(s)
'.M0' num2str(Mach(u)*100) '))'...
       'alpha_bv_a'     num2str(vett_alpha(k1,:))     '.s'     num2str(s)     '.M0'
num2str(Mach(u)*100) ' '...
       'beta_bv_a'     num2str(vett_alpha(k1,:))     '.s'     num2str(s)     '.M0'
num2str(Mach(u)*100) ' '...
       'lambda_bv_a'     num2str(vett_alpha(k1,:))     '.s'     num2str(s)     '.M0'
num2str(Mach(u)*100) ' '...
       'CpL_bv_a'     num2str(vett_alpha(k1,:))     '.s'     num2str(s)     '.M0'
num2str(Mach(u)*100) ' '...
       'ML_bv_a'     num2str(vett_alpha(k1,:))     '.s'     num2str(s)     '.M0'
num2str(Mach(u)*100) ' ];'] );

          end

   name_file=strcat('LUT_V_Cruise_bv_a',vett_alpha_str,'_M0',Mach_str);
   name_var1=strcat('Lut_sonda1');
   name_var2=strcat('Lut_sonda2');
   name_var3=strcat('Lut_sonda3');
   name_var4=strcat('Lut_sonda4');

   cd Dati_Simulazione  % Salva nella cartella "Dati_Simulazione"

   save(name_file,name_var1,name_var2,name_var3,name_var4)

   cd ..
```

```
    end
end

disp('    ');
disp(' Le LUT sono salvati nel file "LUT_V_Cruise_bv_axx_M0xx.mat" ');
disp('  nella cartella Dati_Simulazione .');
disp('     ');

else
end

disp('    ');
disp(' Fine. ');
disp('    ');
```

LUT_MinQuad.m

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%                                                                               %
% NOME FILE      : LUT_MinQuad.m                                              % 
%                                                                               %
% PROGETTO       : Fly-By-Wire (Proff. Galatolo / Denti )                       %
%                                                                               %
% AUTORE         : Rosignoli Nicola                                             %
%                  Cervia Fabio                                                 %
%                  Schettini Francesco                                          %
%                                                                               %
% RIFERIMENTI    : "Sviluppo di modelli e metodi per la                         %
%                   simulazione delle avarie del sistema Dati                   %
%                   Aria di un moderno velivolo Fly-by Wire"                    %
%                                                                               %
%                                                                               %
% ULTIMA REVISIONE  : 7/02/2004                                                 %
%                                                                               %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
```

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

    File relativo all'ampliamento dell'invilupo di volo in Alpha e Beta

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
clc,clear all,close all

% Caricamento dati di Galleria sotto forma dell''ARRAY STRUTTURATO
% ----------------------------------------------------------------
cd Dati_Simulazione

string0mat=['Array_Strutt_DATI_GALL_Cruise.mat'];
```

```
     if exist(string0mat)==2

     eval(['load ' 'Array_Strutt_DATI_GALL_Cruise']);

     else
     end

   %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
   % Caricamento dei dati delle LUT_V sotto forma di ARRAY STRUTTURATO %
   %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

   vett_Mach_bv=[ '060'; '070'; '080'; '085';'090';'095'];

   string(1).alpha='0';
   string(2).alpha='10';
   string(3).alpha='15'; % contributi relativi al Nose-drop (in manovra)
   string(4).alpha='20'; % contributi relativi al Nose-drop (in manovra)
   string(5).alpha='25'; % contributi relativi al Nose-drop (in manovra)

   % Per le sonde 1,2,3,4.
   for  k=1:4                          % al variare del num di sonda
     for i=1:length(vett_Mach_bv)      % al variare del Mach
       for j=1:5                       % al variare di Alpha

           stringmat=['LUT_V_Cruise_bv_a'  string(j).alpha  '_M'  vett_Mach_bv(i,:)
'.mat'];

           if exist(stringmat)==2
              eval(['load     '     'LUT_V_Cruise_bv_a'     string(j).alpha     '_M'
vett_Mach_bv(i,:)]);

              Lut_sonda_a=eval(strcat('Lut_sonda',num2str(k)));
              Ind_size=size(Lut_sonda_a);

   eval(['probe' num2str(k) '.alpha.M' vett_Mach_bv(i,:)  '(' num2str(j) ')' '.bv'
'=Lut_sonda_a(:,2);'] );
   eval(['probe' num2str(k) '.beta.M' vett_Mach_bv(i,:)  '(' num2str(j) ')' '.bv'
'=Lut_sonda_a(:,3);'] );
   eval(['probe' num2str(k) '.lambda.M' vett_Mach_bv(i,:) '(' num2str(j) ')' '.bv'
'=Lut_sonda_a(:,4);'] );
   eval(['probe' num2str(k) '.CpL.M' vett_Mach_bv(i,:) '(' num2str(j) ')' '.bv'
'=Lut_sonda_a(:,5);'] );
   eval(['probe' num2str(k) '.ML.M' vett_Mach_bv(i,:)  '(' num2str(j) ')' '.bv'
'=Lut_sonda_a(:,6);'] );

           else

   eval(['probe' num2str(k) '.alpha.M' vett_Mach_bv(i,:) '(' num2str(j) ')' '.bv'
'=[];'] );
```

```
   eval(['probe' num2str(k) '.beta.M' vett_Mach_bv(i,:) '(' num2str(j) ')' '.bv'
'=[];'] );
   eval(['probe' num2str(k) '.lambda.M' vett_Mach_bv(i,:) '(' num2str(j) ')' '.bv'
'=[];'] );
   eval(['probe' num2str(k) '.CpL.M' vett_Mach_bv(i,:) '(' num2str(j) ')' '.bv'
'=[];'] );
   eval(['probe' num2str(k) '.ML.M' vett_Mach_bv(i,:)   '(' num2str(j) ')' '.bv'
'=[];'] );

          end   % if exist
       end      % for Alpha
     end        % for Mach
   end          % for sonda
   cd ..

   %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
   % Approssimazione ai minimi quadrati e generazione LUT per l''Adm_input  %
   %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

   disp(' ----------------------------------')
   disp(' Approssimazione ai minimi quadrati')
   disp(' e generazione LUT per l''Adm_input');
   disp(' ----------------------------------')
   disp('   ');
   disp(' Grado max del polinomio approssimante scelto = 4.  ')
   pmax=4;
   disp('  ')
   disp('  ')
   disp(' Attendere.....')

   Mach=[0.2;0.4;0.6;0.7;0.8;0.85;0.9;0.95];

   % Assegnazione utile per ridurre il num.di righe del file
   str1='alpha';str2='beta';str3='lambda';str4='CpL';str5='ML';

   % Per ogni Sonda
   % --------------
     for s=1:4
   disp('   ');
   disp(sprintf(' Sonda: %8.3f',s));
   disp(' ------');

   % Per ogni valore di Mach
   % -----------------------
       for u=1:length(Mach)
   disp('  ');
   disp(sprintf(' Mach: %8.3f',Mach(u)));

   % Assegnazioni utili dei valori di galleria opportunamente scalati    %
   % per l'approssimazione ai Minimi Quadrati.                           %
```

```
   % ------------------------------------------------------------------ %

   % Metodo meno ingombrante (in num.di righe) per l'assegnazione
   % str1='alpha';str2='beta';str3='lambda';str4='CpL';str5='ML';

           if Mach(u)<=0.4

              for st=1:5
                  if st==1;
                      str=str1; % Alpha
                  elseif st==2
                      str=str2; % Beta
                  elseif st==3
                      str=str3; % lambda
                  elseif st==4
                      str=str4; % CpL
                  elseif st==5
                      str=str5; % ML
                  end

   % Assegnamento valori di alpha, beta, lambda, CpL e ML  per beta=-5,0,5 e
alpha=0,10,15,20,25
   % av
   eval(['',str,'_av.s' num2str(s) '.M0' num2str(Mach(u)*100) '' ...
               '=[probe'   num2str(s)   '.',str,'.M0'   num2str(Mach(u)*100)   '('
num2str(2) ')' '.av;'...
                 'probe'   num2str(s)   '.',str,'.M0'   num2str(Mach(u)*100)   '('
num2str(3) ')' '.av;'...
                 'probe'   num2str(s)   '.',str,'.M0'   num2str(Mach(u)*100)   '('
num2str(4) ')' '.av];']);

   % bv
   eval(['',str,'_bv.s' num2str(s) '.M0' num2str(Mach(u)*100) '' ...
               '=[probe'   num2str(s)   '.',str,'.M0'   num2str(Mach(u)*100)   '('
num2str(1) ')' '.bv;'...
                 'probe'   num2str(s)   '.',str,'.M0'   num2str(Mach(u)*100)   '('
num2str(2) ')' '.bv;'...
                 'probe'   num2str(s)   '.',str,'.M0'   num2str(Mach(u)*100)   '('
num2str(3) ')' '.bv;'...
                 'probe'   num2str(s)   '.',str,'.M0'   num2str(Mach(u)*100)   '('
num2str(4) ')' '.bv;'...
                 'probe'   num2str(s)   '.',str,'.M0'   num2str(Mach(u)*100)   '('
num2str(5) ')' '.bv];']);

   % Concatena i precedenti vettori av e bv
   eval(['',str,'.s' num2str(s) '.M0' num2str(Mach(u)*100) ''...
   '=vertcat(',str,'_av.s' num2str(s) '.M0' num2str(Mach(u)*100)...
           ',',str,'_bv.s' num2str(s) '.M0' num2str(Mach(u)*100) ');'] );

              end
```

```
          else % Mach>0.4

              for st=1:5
                  if st==1;
                      str=str1; % Alpha
                  elseif st==2
                      str=str2; % Beta
                  elseif st==3
                      str=str3; % lambda
                  elseif st==4
                      str=str4; % CpL
                  elseif st==5
                      str=str5; % ML
                  end

   % Assegnamento valori di alpha, beta, lambda, CpL e ML  per beta=-5,0,5 e
alpha=0,10,15,20,25
   % av
   eval(['',str,'_av.s' num2str(s) '.M0' num2str(Mach(u)*100) '' ...
                  '=[probe'   num2str(s)   '.',str,'.M0'   num2str(Mach(u)*100)   '('
num2str(1) ')' '.av;'...
                    'probe'   num2str(s)   '.',str,'.M0'   num2str(Mach(u)*100)   '('
num2str(2) ')' '.av;'...
                    'probe'   num2str(s)   '.',str,'.M0'   num2str(Mach(u)*100)   '('
num2str(3) ')' '.av;'...
                    'probe'   num2str(s)   '.',str,'.M0'   num2str(Mach(u)*100)   '('
num2str(4) ')' '.av;'...
                    'probe'   num2str(s)   '.',str,'.M0'   num2str(Mach(u)*100)   '('
num2str(5) ')' '.av];']);

   % bv
   eval(['',str,'_bv.s' num2str(s) '.M0' num2str(Mach(u)*100) '' ...
                  '=[probe'   num2str(s)   '.',str,'.M0'   num2str(Mach(u)*100)   '('
num2str(1) ')' '.bv;'...
                    'probe'   num2str(s)   '.',str,'.M0'   num2str(Mach(u)*100)   '('
num2str(2) ')' '.bv;'...
                    'probe'   num2str(s)   '.',str,'.M0'   num2str(Mach(u)*100)   '('
num2str(3) ')' '.bv;'...
                    'probe'   num2str(s)   '.',str,'.M0'   num2str(Mach(u)*100)   '('
num2str(4) ')' '.bv;'...
                    'probe'   num2str(s)   '.',str,'.M0'   num2str(Mach(u)*100)   '('
num2str(5) ')' '.bv];']);

   % Concatena i precedenti vettori av e bv
   eval(['',str,'.s' num2str(s) '.M0' num2str(Mach(u)*100) ''...
   '=vertcat(',str,'_av.s' num2str(s) '.M0' num2str(Mach(u)*100)...
           ',',str,'_bv.s' num2str(s) '.M0' num2str(Mach(u)*100) ');'] );

              end
```

```
        end % if Mach

   % Inizio Approx ai Minimi Quadrati
   % --------------------------------

   % Valori di galleria dell'inviluppo in Alpha e Beta (x,y)

   eval(['X' '=alpha.s' num2str(s) '.M0' num2str(Mach(u)*100) ';'] );
   eval(['Y' '=beta.s' num2str(s) '.M0' num2str(Mach(u)*100) ';'] );

   % Valori di galleria (f=f(x,y)) nei punti dell'inviluppo (x,y)
   % ES: lambda.sx.M0xx è un vettore che contiene, incolonnate, tutte le
   % strisciate di lambda "av" e "bv".

   eval(['f1' '=lambda.s' num2str(s) '.M0' num2str(Mach(u)*100) ';'] );
   eval(['f2' '=CpL.s' num2str(s) '.M0' num2str(Mach(u)*100) ';'] );
   eval(['f3' '=ML.s' num2str(s) '.M0' num2str(Mach(u)*100) ';'] );

   % imax = numero dei punti dove è nota la funzione da approssimare
   eval(['imax' '=length(alpha.s' num2str(s) '.M0' num2str(Mach(u)*100) ');'] );

      for r=1:imax
          x=X(r);
          y=Y(r);

         if pmax==1
             eval(['M_s' num2str(s) '_M0' num2str(Mach(u)*100) '(r,:)' ...
                  '= [x y 1];'] )
         elseif pmax==2
            eval(['M_s' num2str(s) '_M0' num2str(Mach(u)*100) '(r,:)' ...
                  '= [x^2 y^2 x*y x y 1];'] )
         elseif pmax==3
            eval(['M_s' num2str(s) '_M0' num2str(Mach(u)*100) '(r,:)' ...
                  '= [x^3 y^3 x^2*y x*y^2 x^2 y^2 x*y x y 1];'] )
         elseif pmax==4
            eval(['M_s' num2str(s) '_M0' num2str(Mach(u)*100) '(r,:)' ...
                  '= [x^4 y^4 x^3*y x^2*y^2 x*y^3 x^3 y^3 x^2*y x*y^2 x^2 y^2 x*y
x y 1];'] )
         elseif pmax==5
            eval(['M_s' num2str(s) '_M0' num2str(Mach(u)*100) '(r,:)' ...
   '= [x^5 y^5 x^4*y x^3*y^2 x^2*y^3 x*y^4 x^4 y^4 x^3*y x^2*y^2 x*y^3 x^3 y^3
x^2*y x*y^2 x^2 y^2 x*y x y 1];'] )
         elseif pmax==6
            eval(['M_s' num2str(s) '_M0' num2str(Mach(u)*100) '(r,:)' ...
   '= [x^6 y^6 x^5*y x^4*y^2 x^3*y^3 x^2*y^4 x*y^5 x^5 y^5 x^4*y x^3*y^2 x^2*y^3
x*y^4 x^4 y^4 x^3*y x^2*y^2 '...
      'x*y^3 x^3 y^3 x^2*y x*y^2 x^2 y^2 x*y x y 1];'] )
         elseif pmax==7
            eval(['M_s' num2str(s) '_M0' num2str(Mach(u)*100) '(r,:)' ...
```

```
   '= [x^7 y^7 x^6*y x^5*y^2 x^4*y^3 x^3*y^4 x^2*y^5 x*y^6 x^6 y^6 x^5*y x^4*y^2
x^3*y^3 x^2*y^4 x*y^5 x^5 y^5 '...
      'x^4*y x^3*y^2 x^2*y^3 x*y^4 x^4 y^4 x^3*y x^2*y^2 x*y^3 x^3 y^3 x^2*y x*y^2
x^2 y^2 x*y x y 1];'] )
         else
            disp(['Il grado massimo del polinomio deve essere compreso tra 1 e
7']);
         break
         end
      end

   % Si costruisce la matrice A dei coeffienti del sistema lineare
   % che risolve il problema di minimo associato all'approssimazione
   % ai minimi quadrati

   eval(['A_s' num2str(s) '_M0' num2str(Mach(u)*100) ''...
         '=M_s' num2str(s) '_M0' num2str(Mach(u)*100) '''*M_s' num2str(s) '_M0'
num2str(Mach(u)*100) ';'] )

   % Si costruisce la matrice b dei termini noti del sistema lineare
   % che risolve il problema di minimo associato all'approssimazione
   % ai minimi quadrati

   eval(['b1_s' num2str(s) '_M0' num2str(Mach(u)*100) '' '=M_s' num2str(s) '_M0'
num2str(Mach(u)*100) '''*f1' ';'] )
   eval(['b2_s' num2str(s) '_M0' num2str(Mach(u)*100) '' '=M_s' num2str(s) '_M0'
num2str(Mach(u)*100) '''*f2' ';'] )
   eval(['b3_s' num2str(s) '_M0' num2str(Mach(u)*100) '' '=M_s' num2str(s) '_M0'
num2str(Mach(u)*100) '''*f3' ';'] )

   % Vettore soluzione i cui elementi sono i coefficienti della
   % combinazione lineare del polinomio approssimante

   eval(['c1_s' num2str(s) '_M0' num2str(Mach(u)*100) ''...
           '=pinv(A_s' num2str(s) '_M0' num2str(Mach(u)*100) ')' '*b1_s'
num2str(s) '_M0' num2str(Mach(u)*100) ';'] )
   eval(['c2_s' num2str(s) '_M0' num2str(Mach(u)*100) ''...
           '=pinv(A_s' num2str(s) '_M0' num2str(Mach(u)*100) ')' '*b2_s'
num2str(s) '_M0' num2str(Mach(u)*100) ';'] )
   eval(['c3_s' num2str(s) '_M0' num2str(Mach(u)*100) ''...
           '=pinv(A_s' num2str(s) '_M0' num2str(Mach(u)*100) ')' '*b3_s'
num2str(s) '_M0' num2str(Mach(u)*100) ';'] )

   % Vettore che contiene i valori delle funzioni approssimanti
   % nelle stesso ordine del vettore dei valori di galleria (es:lambda.sx.M0xx
richiamato al rigo 193 )

   eval(['lambda_LSQ_s' num2str(s) '_M0' num2str(Mach(u)*100) ''...
           '=M_s' num2str(s) '_M0' num2str(Mach(u)*100) '*c1_s' num2str(s) '_M0'
num2str(Mach(u)*100) ';'] )
```

```
   eval(['CpL_LSQ_s' num2str(s) '_M0' num2str(Mach(u)*100) ''...
           '=M_s' num2str(s) '_M0' num2str(Mach(u)*100) '*c2_s' num2str(s) '_M0'
num2str(Mach(u)*100) ';'] )
   eval(['ML_LSQ_s' num2str(s) '_M0' num2str(Mach(u)*100) ''...
           '=M_s' num2str(s) '_M0' num2str(Mach(u)*100) '*c3_s' num2str(s) '_M0'
num2str(Mach(u)*100) ';'] )

      end  % fine del for sul Mach
   end % fine for sulla sonda

   % Salvataggio di tutte le variabili del file
   disp('  ');
   disp(' Salvataggio di tutte le variabili del file');
   disp(' ----------------------------------------');

   save LUT_MinQuad

   %%%%%%%%%%%%%%%%%%
   % GENERAZIONE LUT %
   %%%%%%%%%%%%%%%%%%

   disp(' ---------------');
   disp(' GENERAZIONE LUT');
   disp(' ---------------');
   disp('  ')
   disp('  ');

   res=input(' Make LUT per l''ADS_input ? (y/n) ','s');
   if res=='y'

   % Per ogni Sonda
   % --------------
     for s=1:4

   disp('   ');
   disp(sprintf(' Sonda: %8.3f',s));
   disp(' ----');

   % Per ogni valore di Mach
   % -----------------------
       for u=1:length(Mach)
   disp('  ');
   disp(sprintf(' Mach: %8.3f',Mach(u)));

   % Passo di campionamento dell'intervallo
   % --------------------------------------
   passo_a=1;
   passo_b=1;
   disp('  ');
   disp(sprintf(' Passo di campionamento del polinomio in alpha: %8.3f',passo_a));
```

```
   disp(sprintf(' Passo di campionamento del polinomio in beta: %8.3f',passo_b));

   xplot=-10:passo_b:25;  % Intervallo max per alpha
   yplot=-20:passo_b:20;  % Intervallo max per beta

   [x,y]=meshgrid(xplot,yplot);

   % Calcolo delle LUT solo per il 4 grado del polinomio approssimante
   % ai minimi quadrati

   % Genera matrici del tipo lambda(beta,alpha) per ogni combinazione di
(beta,alpha)
   % nell'intervallo specificato di alpha e beta.

   eval(['lambdaLSQ_s' num2str(s) '_M0' num2str(Mach(u)*100) '' ...
    '= [c1_s' num2str(s) '_M0' num2str(Mach(u)*100) '(1)*x.^4+'...
       'c1_s' num2str(s) '_M0' num2str(Mach(u)*100) '(2)*y.^4+'...
       'c1_s' num2str(s) '_M0' num2str(Mach(u)*100) '(3)*(x.^3).*y+'...
       'c1_s' num2str(s) '_M0' num2str(Mach(u)*100) '(4)*(x.^2).*y.^2+'...
       'c1_s' num2str(s) '_M0' num2str(Mach(u)*100) '(5)*x.*y.^3+'...
       'c1_s' num2str(s) '_M0' num2str(Mach(u)*100) '(6)*x.^3+'...
       'c1_s' num2str(s) '_M0' num2str(Mach(u)*100) '(7)*y.^3+'...
       'c1_s' num2str(s) '_M0' num2str(Mach(u)*100) '(8)*(x.^2).*y+'...
       'c1_s' num2str(s) '_M0' num2str(Mach(u)*100) '(9)*x.*y.^2+'...
       'c1_s' num2str(s) '_M0' num2str(Mach(u)*100) '(10)*x.^2+'...
       'c1_s' num2str(s) '_M0' num2str(Mach(u)*100) '(11)*y.^2+'...
       'c1_s' num2str(s) '_M0' num2str(Mach(u)*100) '(12)*x.*y+'...
       'c1_s' num2str(s) '_M0' num2str(Mach(u)*100) '(13)*x+'...
       'c1_s' num2str(s) '_M0' num2str(Mach(u)*100) '(14)*y+'...
       'c1_s' num2str(s) '_M0' num2str(Mach(u)*100) '(15)*ones(size(x))];'] )

   eval(['CpLLSQ_s' num2str(s) '_M0' num2str(Mach(u)*100) '' ...
    '= [c2_s' num2str(s) '_M0' num2str(Mach(u)*100) '(1)*x.^4+'...
       'c2_s' num2str(s) '_M0' num2str(Mach(u)*100) '(2)*y.^4+'...
       'c2_s' num2str(s) '_M0' num2str(Mach(u)*100) '(3)*(x.^3).*y+'...
       'c2_s' num2str(s) '_M0' num2str(Mach(u)*100) '(4)*(x.^2).*y.^2+'...
       'c2_s' num2str(s) '_M0' num2str(Mach(u)*100) '(5)*x.*y.^3+'...
       'c2_s' num2str(s) '_M0' num2str(Mach(u)*100) '(6)*x.^3+'...
       'c2_s' num2str(s) '_M0' num2str(Mach(u)*100) '(7)*y.^3+'...
       'c2_s' num2str(s) '_M0' num2str(Mach(u)*100) '(8)*(x.^2).*y+'...
       'c2_s' num2str(s) '_M0' num2str(Mach(u)*100) '(9)*x.*y.^2+'...
       'c2_s' num2str(s) '_M0' num2str(Mach(u)*100) '(10)*x.^2+'...
       'c2_s' num2str(s) '_M0' num2str(Mach(u)*100) '(11)*y.^2+'...
       'c2_s' num2str(s) '_M0' num2str(Mach(u)*100) '(12)*x.*y+'...
       'c2_s' num2str(s) '_M0' num2str(Mach(u)*100) '(13)*x+'...
       'c2_s' num2str(s) '_M0' num2str(Mach(u)*100) '(14)*y+'...
       'c2_s' num2str(s) '_M0' num2str(Mach(u)*100) '(15)*ones(size(x))];'] )

   eval(['MLLSQ_s' num2str(s) '_M0' num2str(Mach(u)*100) '' ...
    '= [c3_s' num2str(s) '_M0' num2str(Mach(u)*100) '(1)*x.^4+'...
```

```
      'c3_s' num2str(s) '_M0' num2str(Mach(u)*100) '(2)*y.^4+'...
      'c3_s' num2str(s) '_M0' num2str(Mach(u)*100) '(3)*(x.^3).*y+'...
      'c3_s' num2str(s) '_M0' num2str(Mach(u)*100) '(4)*(x.^2).*y.^2+'...
      'c3_s' num2str(s) '_M0' num2str(Mach(u)*100) '(5)*x.*y.^3+'...
      'c3_s' num2str(s) '_M0' num2str(Mach(u)*100) '(6)*x.^3+'...
      'c3_s' num2str(s) '_M0' num2str(Mach(u)*100) '(7)*y.^3+'...
      'c3_s' num2str(s) '_M0' num2str(Mach(u)*100) '(8)*(x.^2).*y+'...
      'c3_s' num2str(s) '_M0' num2str(Mach(u)*100) '(9)*x.*y.^2+'...
      'c3_s' num2str(s) '_M0' num2str(Mach(u)*100) '(10)*x.^2+'...
      'c3_s' num2str(s) '_M0' num2str(Mach(u)*100) '(11)*y.^2+'...
      'c3_s' num2str(s) '_M0' num2str(Mach(u)*100) '(12)*x.*y+'...
      'c3_s' num2str(s) '_M0' num2str(Mach(u)*100) '(13)*x+'...
      'c3_s' num2str(s) '_M0' num2str(Mach(u)*100) '(14)*y+'...
      'c3_s' num2str(s) '_M0' num2str(Mach(u)*100) '(15)*ones(size(x))];'] )


   % Es:LUTi_lambda_Cruise_M è una matrice in 3D dove ogni piano rappresenta i
valori
   % della funzione per ogni alpha e beta dell'intervallo specificato.
   % L'ultimo indice indica a quale Mach si riferisce.

   eval(['LUT' num2str(s) '_lambda_Cruise_M(:,:,u)'...
           '=lambdaLSQ_s' num2str(s) '_M0' num2str(Mach(u)*100) ';'] )
   eval(['LUT' num2str(s) '_Cp_local_Cruise_M(:,:,u)'...
           '=CpLLSQ_s' num2str(s) '_M0' num2str(Mach(u)*100) ';'] )
   eval(['LUT' num2str(s) '_M_local_Cruise_M(:,:,u)'...
           '=MLLSQ_s' num2str(s) '_M0' num2str(Mach(u)*100) ';'] )


       end  % fine del for sul Mach
     end % fine for sulla sonda

   % SALVATAGGIO LUT
   disp('  ');
   res=input(' Save LUT ? (y/n) ','s');
    if res=='y'

      for s=1:4
   p_str=num2str(pmax);
   s_str=num2str(s);
   passo_str=num2str(passo_a);
   % Mach_str=num2str(Mach*100);

   name_file=strcat('LUT_MQ_pmax',p_str,'_s',s_str,'_passo',passo_str,'_Cruise');

   name_var1=strcat('LUT',s_str,'_lambda_Cruise_M');
   name_var2=strcat('LUT',s_str,'_Cp_local_Cruise_M');
   name_var3=strcat('LUT',s_str,'_M_local_Cruise_M');

   lut1=name_var1;
```

```
   lut2=name_var2;
   lut3=name_var3;

   cd LUT_ADS_input  % Salva nella cartella "LUT_Adm_input"

   save(name_file,lut1,lut2,lut3)

   cd ..
      end

   disp('    ');
   disp(' Le  LUT  sono  salvati  nel  file  "LUT_MQ_pmaxx_sx_passox_Cruise_M0xx.mat"
');
   disp('  nella cartella LUT_MinQuad .');
   disp('     ');
    else
    end  % fine if decisionale sul salvatagio LUT

   else
   end  % fine if decisionale sulla generazione LUT

   disp('    ');
   disp(' Fine. ');
   disp('    ');
```

APPENDICE B: tabelle test statici

Nelle tabelle seguenti sono riportati i risultati delle simulazioni descritte in § 6.1. In ognuna di queste si possono leggere i valori delle grandezze di simulazione per un assegnato valore di α e di β nell'inviluppo di volo previsto. Si è scelto di documentare le grandezze relative alla sonda 1, per una quota di 8000 m e per i valori di Mach pari a 0.2, 0.4, 0,6, 0.7, 0.8, 0.85, 0.9, 0.95.

- Simulazione: Sensor 1, h=8000 m, Mach=0.2;

α [deg]	β [deg]	Cp_{Li}	P_{Li} [Pa]	M_{Li}	λ_i [deg]	$P_{front\,i}$ [Pa]	$P_{slot2\,i}$ [Pa]	Cp_{front}	Cp_{slot}
-1.0e+001	0	8.6000615e-003	3.5608668e+004	2.2506480e-001	-3.8485873e+000	3.6639247e+004	3.4935032e+004	8.5521564e-001	-5.4478876e-001
-5.0e+000	0	-4.2713808e-002	3.5546204e+004	2.2264277e-001	-6.4351371e+000	3.6551836e+004	3.4888198e+004	7.8340795e-001	-5.8326289e-001
0.0e+000	0	-4.1342560e-002	3.5547873e+004	2.2176232e-001	-3.8544903e+000	3.6545212e+004	3.4895049e+004	7.7796607e-001	-5.7763445e-001
5.0e+000	0	-1.1251613e-002	3.5584503e+004	2.2019806e-001	1.0591006e+000	3.6568128e+004	3.4940226e+004	7.9679195e-001	-5.4052200e-001
1.0e+001	0	3.0493661e-002	3.5635319e+004	2.1685308e-001	6.4177563e+000	3.6589184e+004	3.5009644e+004	8.1408937e-001	-4.8349551e-001
1.5e+001	0	7.3727942e-002	3.5687948e+004	2.1175899e-001	1.1279971e+001	3.6596737e+004	3.5090541e+004	8.2029438e-001	-4.1703916e-001
2.0e+001	0	1.1518596e-001	3.5738414e+004	2.0607587e-001	1.5650613e+001	3.6598042e+004	3.5171947e+004	8.2136639e-001	-3.5016453e-001
2.5e+001	0	1.5850248e-001	3.5791143e+004	2.0209229e-001	2.0480923e+001	3.6617557e+004	3.5245630e+004	8.3739754e-001	-2.8963411e-001

- Simulazione: Sensor 1, h=8000 m, Mach=0.4;

α [deg]	β [deg]	Cp_{Li}	P_{Li} [Pa]	M_{Li}	λ_i [deg]	$P_{front\,i}$ [Pa]	$P_{slot2\,i}$ [Pa]	Cp_{front}	Cp_{slot}
-1.0e+001	0	-1.1171668e-001	3.5054232e+004	4.5364524e-001	-7.5966956e+000	3.9611233e+004	3.2334545e+004	8.2417211e-001	-6.7026918e-001
-5.0e+000	0	-7.0073119e-002	3.5257002e+004	4.4516978e-001	-6.7249467e+000	3.9654899e+004	3.2623976e+004	8.3313996e-001	-6.1082777e-001
0.0e+000	0	-5.3976941e-002	3.5335377e+004	4.4281986e-001	-2.4997225e+000	3.9692298e+004	3.2724592e+004	8.4082075e-001	-5.9016376e-001
5.0e+000	0	-3.6517899e-002	3.5420388e+004	4.4038415e-001	2.6509419e+000	3.9735418e+004	3.2832348e+004	8.4967642e-001	-5.6803348e-001
1.0e+001	0	-2.7455468e-003	3.5584831e+004	4.3434545e-001	7.3167195e+000	3.9791020e+004	3.3056350e+004	8.6109575e-001	-5.2202940e-001
1.5e+001	0	5.0330765e-002	3.5843268e+004	4.2388068e-001	1.1104992e+001	3.9860373e+004	3.3418905e+004	8.7533906e-001	-4.4757015e-001
2.0e+001	0	1.1374189e-001	3.6152028e+004	4.1086088e-001	1.4640847e+001	3.9937537e+004	3.3856109e+004	8.9118638e-001	-3.5777989e-001
2.5e+001	0	1.6655888e-001	3.6409203e+004	3.9985125e-001	1.9567084e+001	4.0003045e+004	3.4220328e+004	9.0464001e-001	-2.8297876e-001

- Simulazione: Sensor 1, h=8000 m, Mach=0.6;

α [deg]	β [deg]	Cp_{Li}	P_{Li} [Pa]	M_{Li}	λ_i [deg]	$P_{front\,i}$ [Pa]	$P_{slot2\,i}$ [Pa]	Cp_{front}	Cp_{slot}
-1.0e+001	0	-2.2776132e-001	3.3102930e+004	6.9483277e-001	-1.0018169e+001	4.4252418e+004	2.6971135e+004	7.8993338e-001	-7.8745485e-001
-5.0e+000	0	-7.2422588e-002	3.4804764e+004	6.7046480e-001	-6.2680505e+000	4.5612079e+004	2.8816816e+004	9.1403954e-001	-6.1898611e-001
0.0e+000	0	-3.2842471e-002	3.5238389e+004	6.6402815e-001	-2.0647873e+000	4.5943239e+004	2.9295332e+004	9.4426689e-001	-5.7530853e-001
5.0e+000	0	-2.8065891e-002	3.5290720e+004	6.6158268e-001	2.3426374e+000	4.5922080e+004	2.9383948e+004	9.4233557e-001	-5.6721986e-001
1.0e+001	0	-8.5399441e-003	3.5504639e+004	6.5454260e-001	6.7801167e+000	4.5944115e+004	2.9691630e+004	9.4434688e-001	-5.3913546e-001
1.5e+001	0	4.3886103e-002	3.6078999e+004	6.3967644e-001	1.1148420e+001	4.6149903e+004	3.0444641e+004	9.6313061e-001	-4.7040274e-001
2.0e+001	0	1.1596082e-001	3.6868623e+004	6.1910710e-001	1.5423194e+001	4.6428347e+004	3.1484491e+004	9.8854625e-001	-3.7548807e-001
2.5e+001	0	1.6303059e-001	3.7384302e+004	6.0031181e-001	1.9654962e+001	4.6428493e+004	3.2258814e+004	9.8855960e-001	-3.0480999e-001

- Simulazione: Sensor 1, h=8000 m, Mach=0.7;

α [deg]	β [deg]	Cp_{Li}	P_{Li} [Pa]	M_{Li}	λ_i [deg]	$P_{front\,i}$ [Pa]	$P_{slot2\,i}$ [Pa]	Cp_{front}	Cp_{slot}
-1.0e+001	0	-1.6533324e-001	3.3132778e+004	8.0358365e-001	-1.0460478e+001	4.8744275e+004	2.4796261e+004	8.8158696e-001	-7.2438717e-001
-5.0e+000	0	-6.7496478e-002	3.4591703e+004	7.8619955e-001	-6.2630688e+000	5.0079007e+004	2.6286878e+004	9.7109520e-001	-6.2442507e-001
0.0e+000	0	-3.3903809e-002	3.5092632e+004	7.7750588e-001	-1.9739050e+000	5.0402918e+004	2.6864787e+004	9.9281694e-001	-5.8567002e-001
5.0e+000	0	-2.3150103e-002	3.5252989e+004	7.7216945e-001	2.3671201e+000	5.0389178e+004	2.7107570e+004	9.9189558e-001	-5.6938881e-001
1.0e+001	0	-8.5743469e-003	3.5470340e+004	7.6592952e-001	6.7209820e+000	5.0415961e+004	2.7414393e+004	9.9369163e-001	-5.4881298e-001
1.5e+001	0	2.1740352e-002	3.5922388e+004	7.5559774e-001	1.1049525e+001	5.0590127e+004	2.7994471e+004	1.0053714e+000	-5.0991246e-001
2.0e+001	0	6.4966765e-002	3.6566973e+004	7.3905821e-001	1.5315464e+001	5.0754616e+004	2.8863702e+004	1.0164021e+000	-4.5162107e-001
2.5e+001	0	1.0353354e-001	3.7142074e+004	7.1526744e-001	1.9482380e+001	5.0509385e+004	2.9834826e+004	9.9995673e-001	-3.8649672e-001

- Simulazione: Sensor 1, h=8000 m, Mach=0.8;

α [deg]	β [deg]	Cp_{Li}	P_{Li} [Pa]	M_{Li}	λ_i [deg]	$P_{front\,i}$ [Pa]	$P_{slot2\,i}$ [Pa]	Cp_{front}	Cp_{slot}
-1.0e+001	0	-6.7390300e-002	5.9591629e+004	8.5686520e-001	-1.0367191e+001	9.2273781e+004	4.2325202e+004	1.0084078e+000	-6.3574912e-001
-5.0e+000	0	-5.2162925e-002	6.0054228e+004	8.5368726e-001	-6.1501196e+000	9.2698657e+004	4.2798741e+004	1.0223935e+000	-6.2016165e-001
0.0e+000	0	-2.6296017e-002	6.0840051e+004	8.4044271e-001	-1.8492360e+000	9.2702600e+004	4.3958073e+004	1.0225233e+000	-5.8199993e-001
5.0e+000	0	-3.9846407e-003	6.1517858e+004	8.2867836e-001	2.4806001e+000	9.2677558e+004	4.4969973e+004	1.0216990e+000	-5.4869123e-001
1.0e+001	0	8.9918234e-003	6.1912076e+004	8.2302327e-001	6.7961748e+000	9.2768767e+004	4.5505785e+004	1.0247013e+000	-5.3105393e-001
1.5e+001	0	1.5269678e-002	6.2102793e+004	8.2118872e-001	1.1065919e+001	9.2892199e+004	4.5725881e+004	1.0287643e+000	-5.2380900e-001
2.0e+001	0	2.5900908e-002	6.2425764e+004	8.1396825e-001	1.5269911e+001	9.2738865e+004	4.6276790e+004	1.0237170e+000	-5.0567476e-001
2.5e+001	0	6.0353183e-002	6.3472405e+004	7.8523762e-001	1.9399872e+001	9.1809176e+004	4.8273627e+004	9.9311444e-001	-4.3994488e-001

- Simulazione: Sensor 1, h=8000 m, Mach=0.85;

α [deg]	β [deg]	Cp_{Li}	P_{Li} [Pa]	M_{Li}	λ_i [deg]	$P_{front\,i}$ [Pa]	$P_{slot2\,i}$ [Pa]	Cp_{front}	Cp_{slot}
-1.0e+001	0	2.7040017e-002	3.6192738e+004	9.4279265e-001	-1.0468870e+001	6.1401523e+004	2.2876109e+004	1.1735536e+000	-5.7860976e-001
-5.0e+000	0	1.9807092e-002	3.6033705e+004	9.3634088e-001	-5.7448780e+000	6.0671569e+004	2.3042043e+004	1.1403547e+000	-5.7106295e-001
0.0e+000	0	-1.6540591e-003	3.5561831e+004	9.3136901e-001	-1.4917245e+000	5.9537183e+004	2.2932377e+004	1.0887620e+000	-5.7605062e-001
5.0e+000	0	-2.2432460e-003	3.5548876e+004	9.2294788e-001	2.6384824e+000	5.8956778e+004	2.3232477e+004	1.0623648e+000	-5.6240186e-001
1.0e+001	0	3.4280443e-002	3.6351935e+004	9.0691540e-001	6.8354575e+000	5.9248980e+004	2.4312776e+004	1.0756544e+000	-5.1326910e-001
1.5e+001	0	1.0529864e-001	3.7913437e+004	8.7987660e-001	1.1130739e+001	6.0078627e+004	2.6239872e+004	1.1133873e+000	-4.2562342e-001
2.0e+001	0	1.8933371e-001	3.9761144e+004	8.3920357e-001	1.5397688e+001	6.0511656e+004	2.8764160e+004	1.1330817e+000	-3.1081698e-001
2.5e+001	0	2.4604872e-001	4.1008157e+004	7.8303548e-001	1.9351488e+001	5.9196618e+004	3.1247192e+004	1.0732729e+000	-1.9788694e-001

- Simulazione: Sensor 1, h=8000 m, Mach=0.9;

α [deg]	β [deg]	Cp_{Li}	P_{Li} [Pa]	M_{Li}	λ_i [deg]	$P_{front\,i}$ [Pa]	$P_{slot2\,i}$ [Pa]	Cp_{front}	Cp_{slot}
-1.0e+001	0	2.6369373e-002	3.6248209e+004	9.4370513e-001	-1.0477764e+001	6.1562479e+004	2.2871777e+004	1.0533104e+000	-5.1628134e-001
-5.0e+000	0	2.0282716e-002	3.6098172e+004	9.3678466e-001	-5.7424576e+000	6.0811316e+004	2.3065443e+004	1.0228375e+000	-5.0842477e-001
0.0e+000	0	-1.7612278e-003	3.5554785e+004	9.3205268e-001	-1.4898568e+000	5.9571642e+004	2.2901906e+004	9.7254679e-001	-5.1505908e-001
5.0e+000	0	-2.8640907e-003	3.5527599e+004	9.2385306e-001	2.6368444e+000	5.8980599e+004	2.3186322e+004	9.4856955e-001	-5.0352100e-001
1.0e+001	0	3.4148834e-002	3.6439974e+004	9.0763861e-001	6.8324384e+000	5.9438311e+004	2.4347575e+004	9.6713784e-001	-4.5641165e-001
1.5e+001	0	1.0672859e-001	3.8229077e+004	8.7997104e-001	1.1129704e+001	6.0584632e+004	2.6455374e+004	1.0136414e+000	-3.7090315e-001
2.0e+001	0	1.9260258e-001	4.0345886e+004	8.3852098e-001	1.5399406e+001	6.1360949e+004	2.9207246e+004	1.0451348e+000	-2.5926609e-001
2.5e+001	0	2.4977452e-001	4.1755184e+004	7.8206798e-001	1.9350297e+001	6.0221766e+004	3.1842560e+004	9.9892078e-001	-1.5235755e-001

- Simulazione: Sensor 1, h=8000 m, Mach=0.95;

α [deg]	β [deg]	Cp_{Li}	P_{Li} [Pa]	M_{Li}	λ_i [deg]	$P_{front\,i}$ [Pa]	$P_{slot2\,i}$ [Pa]	Cp_{front}	Cp_{slot}
-1.0e+001	0	2.6369373e-002	3.6322438e+004	9.4370513e-001	-1.0477764e+001	6.1688547e+004	2.2918614e+004	9.4994349e-001	-4.6166075e-001
-5.0e+000	0	2.0282716e-002	3.6155267e+004	9.3678466e-001	-5.7424576e+000	6.0907500e+004	2.3101925e+004	9.2150575e-001	-4.5498645e-001
0.0e-000	0	-1.7612278e-003	3.5549827e+004	9.3205268e-001	-1.4898568e+000	5.9563336e+004	2.2898712e+004	8.7256503e-001	-4.6238537e-001
5.0e+000	0	-2.8640907e-003	3.5519537e+004	9.2385306e-001	2.6368444e+000	5.8967215e+004	2.3181060e+004	8.5086041e-001	-4.5210516e-001
1.0e+001	0	3.4148834e-002	3.6536103e+004	9.0763861e-001	6.8324384e+000	5.9595109e+004	2.4411804e+004	8.7372189e-001	-4.0729406e-001
1.5e+001	0	1.0672859e-001	3.8529517e+004	8.7997104e-001	1.1129704e+001	6.1060762e+004	2.6663285e+004	9.2708599e-001	-3.2531815e-001
2.0e+001	0	1.9260258e-001	4.0888060e+004	8.3852098e-001	1.5399406e+001	6.2185527e+004	2.9599738e+004	9.6803840e-001	-2.1840259e-001
2.5e+001	0	2.4977452e-001	4.2458296e+004	7.8206798e-001	1.9350297e+001	6.1235836e+004	3.2378754e+004	9.3346037e-001	-1.1721924e-001