POLITECNICO DI TORINO

DIPARTIMENTO DI INGEGNERIA MECCANICA E AEROSPAZIALE

Corso di Laurea in Ingegneria Meccanica

Tesi di Laurea

Analisi del ribaltamento di veicoli pesanti con tecniche di intelligenza artificiale

Relatori

Prof. Guido Perboli

Prof. Alessandro Vigliani

Candidato

Filippo Velardocchia

Anno Accademico 2020-2021

"Prima ero un balah,
ora sono un tamer"

Ringraziamenti

Ringrazio i miei relatori, che mi hanno dato la possibilità di svolgere un lavoro che mi è piaciuto e stimolato tantissimo.
Bene.
Ho deciso di iniziare facile, perché non nego di essere abbastanza emozionato nello scrivere quello che è il primo dei capitoli della mia Tesi, ma l'ultimo in ordine di stesura. È una sensazione strana trovarsi, dopo tutto questo tempo, a pensare di dover scrivere dei ringraziamenti... A realizzare che qualcosa è finito. Non è però mia intenzione annoiare fin dall'inizio di questa trattazione, per quello c'è tempo dopo. Voglio però davvero ripercorrere, insieme a una serie di persone, quello che è stato il mio viaggio al Poli.
Saluto primo tra tutti il mitico Afanasie, compagno (in tutti i sensi, vista la provenienza) di mille peripezie fin dal primo anno. Non sono ancora sicuro 42! sia la risposta ad ogni domanda dell'universo, ma nel mentre, ora che finalmente non ho più scuse, prometto di vedere The Martian (forse).
Proseguo con il grande Luca, che probabilmente sarebbe ancora in fila nell'aula di disegno se non l'avessi fatto passare. Epocale una sua premeditata uscita "animalesca" durante l'ora di Macchine, la porterò sempre nel cuore. Pensi a lui, e non puoi dimenticare l'intrepido Andrea, ancora convinto di saper giocare a calcio (certo, se il metro di paragone è la sua forza su Fifa, potrebbe anche avere ragione). Persona di grande cuore e di straordinaria sensibilità, non farebbe mai nulla per spoilerare la tua serie TV preferita o diffondere notizie che paiono arrivare direttamente da

Lercio. Amico vero.

Continuando in quello che ormai è un tour vorticoso di ricordi, è impossibile dimenticare il nobile Ardit, e il suo fiero trotto intorno ad una tavola rotonda per disquisire con il suo valvallone, Arco un tempo Marco, la persona forse più importante per me in questa ultima parte di percorso.

A proposito di persone importanti, non posso dimenticare Ilaria. Donna straordinaria, dagli innumerevoli contatti, pochi possono resistere al suo fascino. Grande appassionata di Eco, la prima volta che abbiamo seguito lezione insieme ha fatto fare un disegno "simpatico" ad Andrea sul mio quaderno. Insostituibile (sto ancora aspettando un invito a mangiare insieme, ieri mi hanno rifilato un pranzo terribile, tipo un certo bar).

Dato che ormai si fa tardi e devo letteralmente volare per evitare di consegnare la Tesi fuori scadenza, passo a salutare poi Antonino, amico straordinario, e Francesca, collegamento puramente casuale. Saluto anche Federico e "colei che vagabonda", augurandole di poter trovare presto la sua casa (e magari dello speck migliore), nonché il grande Giack, che a momenti non passa più dalle porte (omino Michelin) e Sara, che mi ha tenuto compagnia in quest'ultimo periodo aiutandomi a riscoprire aspetti come la musica e il disegno (probabilmente quando leggerò di nuovo questa dedica staremo ancora cercando di organizzare una passegiata).

A tutti voi, grazie (così come quelli che ho dimenticato, non offendetevi). Alla fine mi sono dilungato più del previsto, ma penso sapeste già che sarebbe andata a finire così. Non mi andava, però, di creare una semplice lista della spesa, se siete finiti qua dentro è perché credo che, riprendendo a distanza di anni una Tesi ormai impolverata, questa possa ancora regalarmi i sorrisi di un tempo. Termino salutando la mia famiglia e in particolare mio padre, uomo straordinario che mi ha sempre fornito il massimo supporto, e mia madre, donna altrettanto straordinaria. Ringrazio infine mio fratello Giovanni per premurarsi quasi più di me che non vada oltre la scadenza.

Non credo di aver scritto in maniera perfetta, ma ormai non c'è più tempo.

A presto,

Filippo

Indice

Elenco delle figure

Introduzione

Il lavoro presentato, come da titolo, verterà sull'analisi del ribaltamento di veicoli pesanti (nel caso specifico di un truck) e sulla possibilità di stimare e prevenire lo stesso tramite tecniche di intelligenza artificiale (IA). Lo studio di questa tematica risulta essere di particolare interesse per via del fatto che questa tipologia di veicoli, nella sola Europa e in Canada e Stati Uniti, è responsabile del trasporto di più dell'80% delle merci (Tianjun Zhu and Ma, 2020). Sempre riferendoci a Zhu ((Tianjun Zhu and Li, 2020)), viene poi evidenziato come quasi il 14% degli incidenti mortali sui mezzi in questione (caratterizzati da peso e dimensioni elevate-pullman, camion etc.) sia legato a ribaltamento degli stessi. Questa statistica assume proporzioni ancora maggiori se si considera che in Francia quasi la metà di eventi con conseguenze tragiche o molto serie sui veicoli pesanti siano da attribuire proprio a un verificarsi di condizioni di rollover (Yamine Sellami and Cadiou, 2018) e che, negli USA, queste portino ad incidenti mortali nel 33% dei casi ((Taewung Kim, 2017)). Risulta molto importante quindi, in quest'ottica, riuscire a sviluppare dei metodi che permettano di identificare quelle che possono essere le caratteristiche critiche di rollio che portano al ribaltamento dei veicoli sopracitati. Per fare ciò, è stata qui sviluppata una metodologia articolabile in due fasi principali. La prima di queste consiste nella generazione di dati tramite il programma Simscape, caratterizzato dall'avere a disposizione un modello realistico dei principali tipi di veicoli e, soprattutto, di permettere la definizione di manovre (su determinate superfici, personalizzabili anch'esse) e vedere come il mezzo scelto si comporta a

seguito delle stesse. Terminata questa prima parte, che si vedrà dettagliatamente nel primo capitolo della trattazione, si passa alla seconda fase, dove entra in gioco il vero e proprio algoritmo di IA. In particolare, questo risulterà essere composto da due reti neurali, una avente scopo di previsione/stima e una di predizione/anticipazione delle variabili oggetto di studio, rappresentate come detto dal rollio e, particolarmente rilevante, dai Load Transfer Ratio (LTR), ovvero dai parametri adimensionali dei trasferimenti di carico verticale sui singoli assali (tre nel caso in esame), veri indici di condizioni critiche di rollover (ribaltamento). Come sarà possibile osservare in seguito, l'efficacia dei due network viene poi testata su quattro manovre e superfici completamente nuove per gli stessi (ovvero non presenti in fase di addestramento), che permetteranno di valutare obiettivamente il potenziale di quanto realizzato. Si evidenzia, infine, che quanto creato è stato poi progressivamente e interamente implementato (tramite il livescript di Matlab) su un'interfaccia grafica presente su Simscape, nell'ottica di consentire, ad un generico utente, l'utilizzo completo di quanto realizzato in una maniera il più possibile facile e immediata.

Simscape: generazione ed elaborazione dati

Prefazione

In questo capitolo verrà presentata la prima parte dell'attività di Tesi. In particolare, dopo una breve premessa sull'ambiente di simulazione e sulla sua UI, sarà poi possibile seguire quanto fatto in ottica di generazione ed elaborazione dei dati, con presentazione di differerenti risultati ed evidenziazione dei codici sviluppati per ottenerli. Al termine viene poi proposta una versione modificata della UI del programma, con scopo di ottenere un sistema facilmente utilizzabile da un utente esterno.

Simscape

Presentazione

Nell'ottica di generare ed effettuare una prima elaborazione dei dati che saranno successivamente utilizzati per valutare l'efficacia dell'algoritmo di IA, ci si è avvalsi, come anticipato, del software Simscape in ambiente Matlab. Questo consente di costruire modelli di componenti fisici (quali ad esempio motori elettrici, sospensioni etc.) interagenti fra loro in maniera realistica, risultando quindi un programma

adatto per questa prima parte della trattazione. Nello specifico, si è poi utilizzato un particolare template del tool in questione, ovvero "Simscape Vehicle Templates", sviluppato da Steve Miller. Questo presenta la possibilità di effettuare diverse analisi preliminari sul veicolo che si sceglie di comporre (si vedrà meglio successivamente il significato di questa parola) e studiare. Si evidenzia che, a seconda che lo stesso sia a due o 3 assi, si hanno due generici modelli preliminari, come possibile osservare in Figura 1 e in Figura 2.

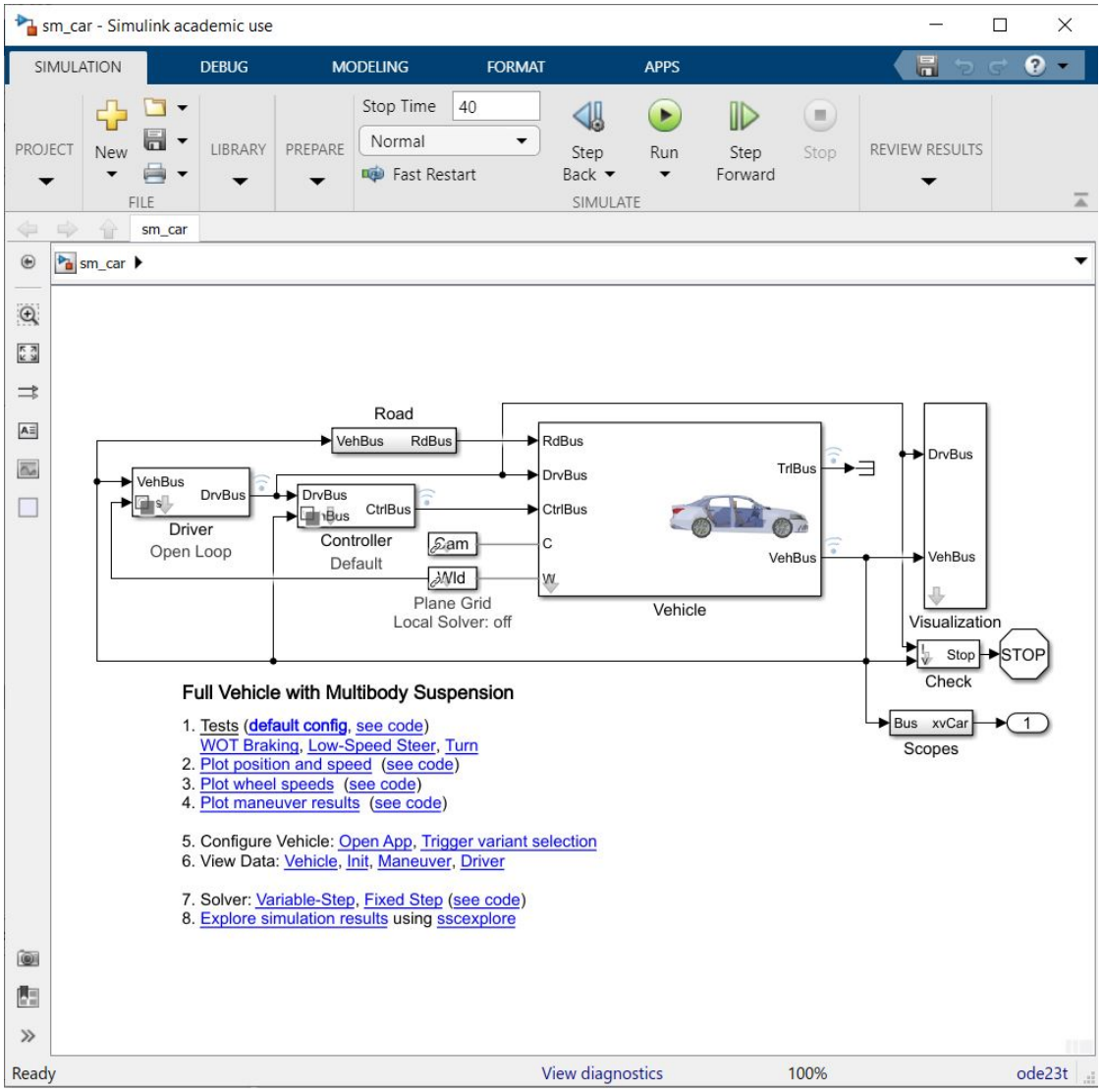


Figura 1: Modello Simulink di un generico veicolo a due assi

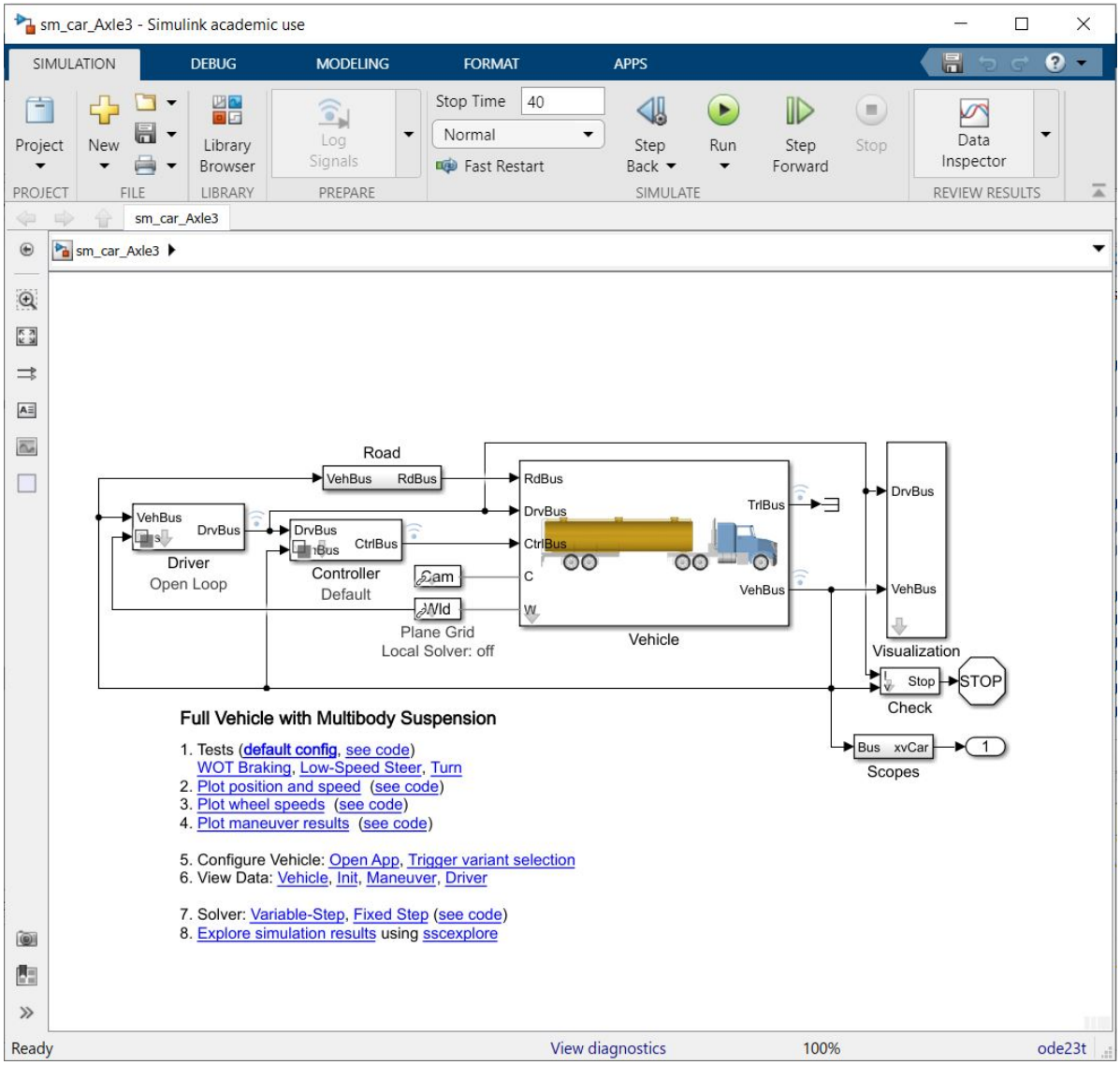


Figura 2: Modello Simulink di un generico veicolo a tre assi

I due modelli Simulink sopra presentati sono il primo punto di partenza per la generazione dei dati di interesse. Distinguiamo, in particolare, diversi blocchi caratterizzanti e comuni ai due sistemi, ovvero:

- *Driver*

- *Road*

- *Controller*
- *Cam*
- *World*
- *Vehicle*
- *Visualization*
- *Check*
- *Scopes*

Driver

Il blocco *Driver* risulta essere di fondamentale importanza in questa trattazione. Esso è, infatti, quello tramite cui vengono passate le informazioni di traiettoria al veicolo. Distinguiamo, in particolare, due tipologie di modelli utilizzabili all'interno di questo blocco, *Open Loop* (OL) e *Closed Loop* (CL), la prima delle quali immediatamente osservabile nella figura sottostante.

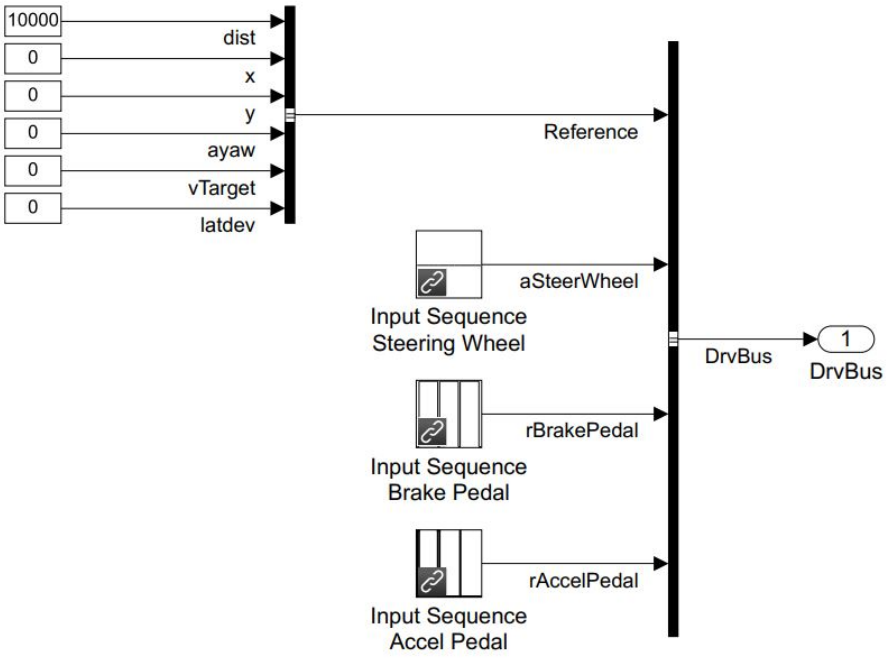


Figura 3: Driver Open Loop

Selezionando (solitamente tramite UI, come si vedrà più avanti) il modello OL, come intuibile da Figura 3, è sufficiente definire (tramite inserimento in libreria di un apposito foglio Excel, di cui discuteremo sempre in questa fase presentativa) una manovra in termini di angolo volante di sterzo, posizione pedale acceleratore e posizione pedale freno, senza l'intervento dei bus provenienti dai blocchi *Vehicle* e *World*, portatori rispettivamente delle informazioni della struttura veicolo e della superficie (*Scene*) su cui si sta muovendo. Da qui la definizione di modello *Open Loop*, il circuito tra il blocco *Vehicle* e quello *Driver* rimane "aperto". Avviene invece diversamente nel secondo caso che andremo a presentare, ovvero quello di *Closed Looop* (Figura 4).

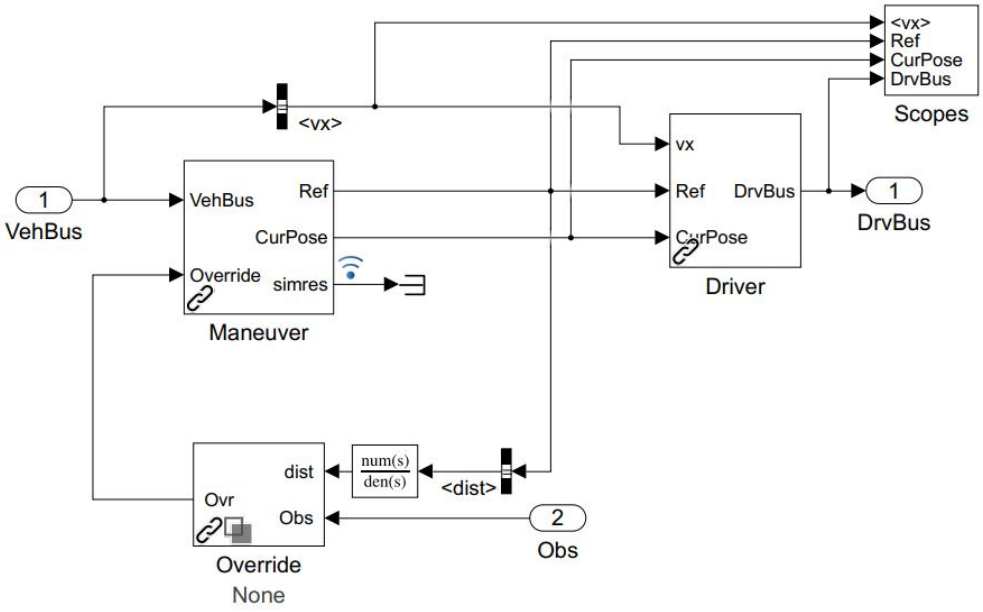


Figura 4: Driver Closed Loop

Come possibile commentare dalla figura immediatamente soprastante, il modello risulta essere effettivamente un *Closed Loop* (circuito "chiuso" tra blocco *Vehicle* e blocco *Driver*), dove si osserva l'intervento dei bus contenenti le informazioni di veicolo e superficie. Particolare molto rilevante è che, sebbene anche in questo caso le informazioni uscenti dal blocco (tramite il bus del *Driver*) siano sempre caratterizzate dall'essere in termini di posizione dei pedali di acceleratore e freno, oltre che di angolo di sterzo del volante, risulta necessario definire la manovra che si vorrà poi fare realizzare al mezzo in termini di parametri di posizione e velocità (blocco *Maneuver* in Figura 4), perché poi questi vengano convertiti negli elementi tipici in uscita dal *Driver* (omonimo blocco sempre in Figura 4, associato chiaramente al bus di uscita). Questa operazione risulta essere computazionalmente più onerosa di quella vista in caso di manovra in OL, in quanto si ha una continua verifica di quella che è una vera e propria traiettoria che si vuole fare seguire al

veicolo (in genere questo tipo di manovre sono quelle associate a specifici circuiti fisici, con percorsi da seguire precisamente).

Road

Legato direttamente al blocco *Vehicle* (di cui è possibile osservare il bus in ingresso in Figura 5), il blocco *Road* è caratterizzato dall'avere all'interno un modello che simula l'effetto di vari elementi, quali vento, altezza strada e attrito della stessa, attribuendo i medesimi alle componenti del veicolo generato (ad esempio, un determinato attrito per una determinata ruota) e producendo in uscita a sua volta, come logico, un *Road* bus. Questo blocco risulta particolarmente importante quando si va ad intervenire, a livello di codice o tramite UI, sulla tipologia di superficie su cui si sceglie di fare svolgere la manovra.

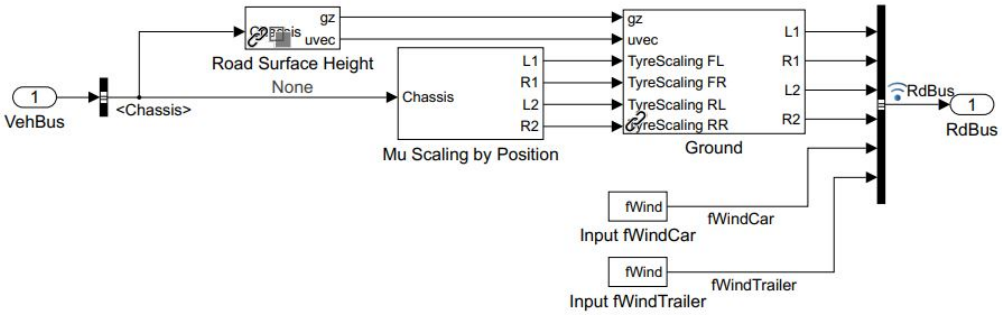


Figura 5: Modello blocco Road

Controller

Questo blocco, sequenziale a quello *Driver* e in CL con quello *Vehicle*, è caratterizzato dall'avere due modelli differenti, uno di default e uno per i motori Fuel cell. Avendo scelto di simulare il veicolo tramite motori ideali, si presenta solo il primo dei due.

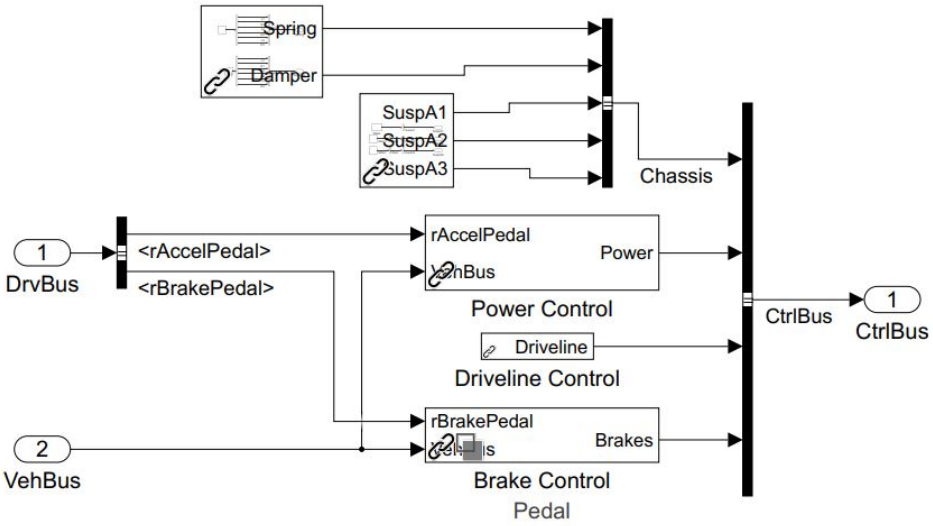


Figura 6: Modello default blocco Controller

Funzione di questo blocco è quello di monitorare la richiesta nei motori (sotto blocco *Power Control*), le valvole regolanti i freni e il moto di molle e sospensioni. Risulta essere stato coinvolto solo in maniera indiretta da quanto svolto successivamente nella trattazione.

Cam

Questo blocco risulta essere rilevante per poter scegliere il sistema di riferimento da cui osservare il moto del veicolo, non produce però alcun bus e si può considerare un elemento costante ed inalterato da eventuali modifiche sui componenti caratterizzanti il modello complessivo (Figure 1 e 2).

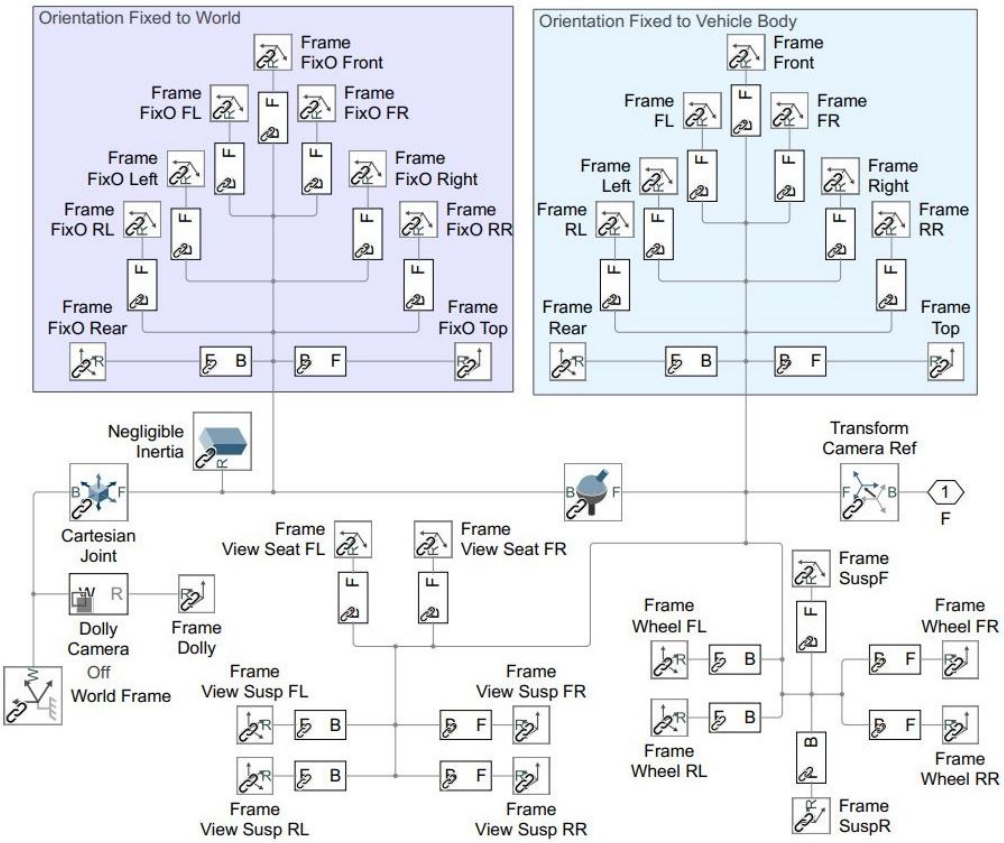


Figura 7: Sistemi di riferimento a disposizione per osservare il moto

World

Come osservabile in Figura 8, il blocco in questione ha la semplice funzione di riportare e selezionare i sotto blocchi corrispondenti alle varie superfici a disposizione (i cui parametri sono poi inseriti effettivamente nel modello *Road*.

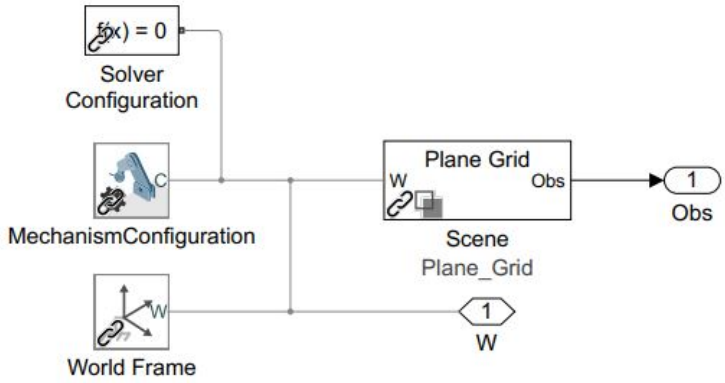


Figura 8: Superficie selezionata

Vehicle

Il blocco considerato è, con ogni probabilità, quello più importante per questa trattazione. All'interno dello stesso, infatti, vengono selezionati e scelti i modelli relativi ad ognuna delle componenti caratteristiche del mezzo, quali sospensioni, motori, pneumatici, freni (come osservabile in Figura 9) etc., permettendo la creazione del veicolo più adatto per gli studi di interesse. Inoltre, ciascuna di queste, oltre ad avere i sistemi precedentemente citati, richiama specifici file STL che compongono il veicolo anche in termini di fisionomia CAD. Si fa notare che, per quanto non effettuato in questo lavoro di Tesi (se non a livello molto superficiale), è possibile intervenire autonomamente sui file 3D, inserendone eventualmente di propri (Figura 10, non è stato però verificato come il programma possa reagire a questa eventualità, soprattutto in termini di comportamenti fisici).

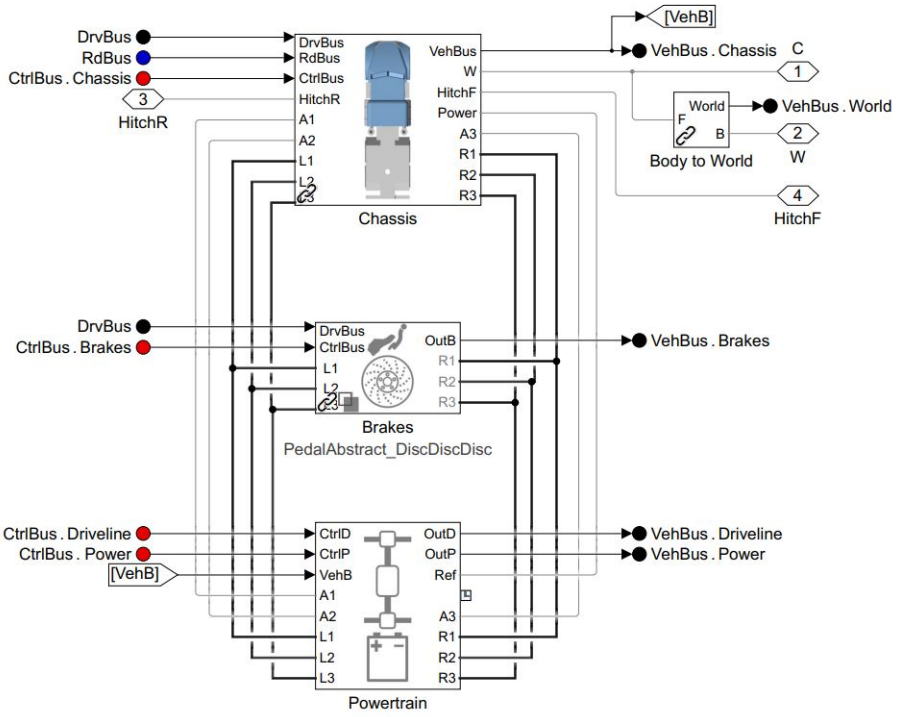


Figura 9: Struttura blocco *Vehicle* per veicolo a tre assi

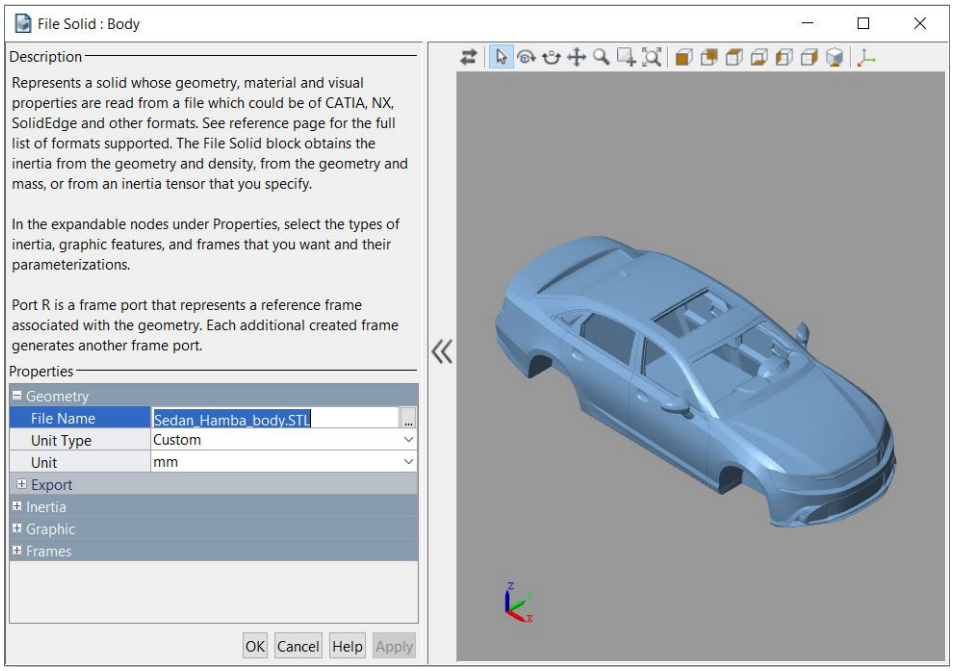


Figura 10: Particolare di file STL dello chassis di una berlina

Come anticipato, i modelli associati al sistema *Vehicle* sono quelli centrali nel progetto esaminato. Essi infatti ricevono in ingresso i bus provenienti dai blocchi *Driver*, *Controller* e *Road*, per poi restituire in uscita, tramite un bus (ambiente di modellazione) e un *logsout* (Workspace di Matlab) di Simulink, le informazioni caratterizzanti ciascuno dei componenti che costituiscono il veicolo, come ad esempio le azioni verticali per quanto concerne le ruote o le accelerazioni longitudinale, laterale e trasversale nei sistemi di riferimento *World* (osservatore solidale alla superficie su cui si muove il veicolo) o *baricentrico*. Proprio su queste informazioni sono stati poi basati i codici *Post-Processor* che hanno permesso l'effettiva analisi dei dati.
Si evidenzia, infine, il fatto che risulta essere possibile studiare non solo il sistema veicolo in sé e per sé, ma anche il movimento e le azioni caratterizzanti di un trailer (personalizzabile a sua volta) collegato allo stesso, sia per quanto concerne i mezzi a due assi, sia per i veicoli a tre, sulla falsariga di quanto già visto per il blocco *Vehicle*. Viene portato all'attenzione, però, che il bus e relativo *logsout* associati al rimorchio (*TrlBus* in Figura 2) non sono coinvolti nei collegamenti in CL con gli altri blocchi del modello generale (sempre Figura 2), per quanto sia comunque logicamente garantito un legame con il veicolo trainante tramite apposito blocco. In ogni caso, si fa notare che in questa trattazione si è presa la decisione di analizzare il mezzo prescelto senza la presenza di un trailer, in modo da verificare le prestazioni del solo veicolo considerato.

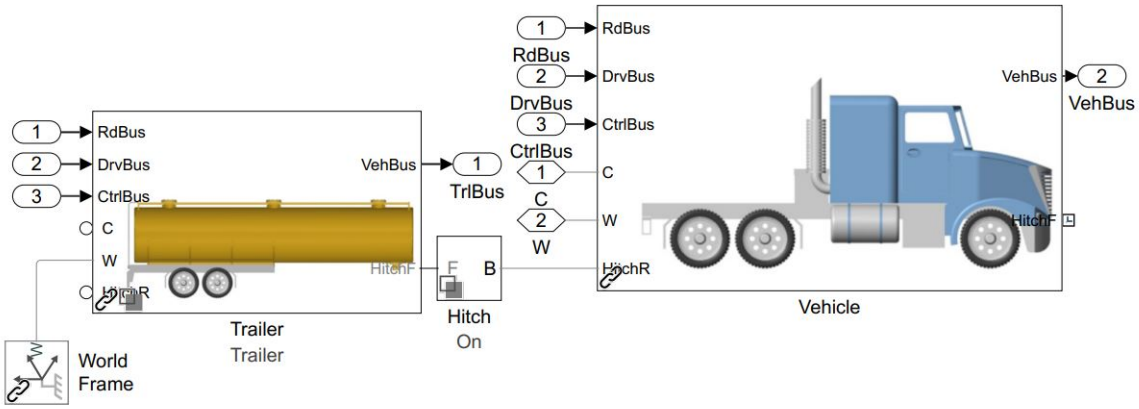


Figura 11: Modello di interazione tra rimorchio e veicolo

Visualization

Come osservabile in Figura 12, il blocco in questione è adibito alla visualizzazione del movimento del veicolo in fase di simulazione. Questo può essere fatto in contemporanea alla stessa sfruttando il sottosistema *XY Plot*, che restituisce percorso, posizione e velocità del veicolo su un piano bidimensionale. Avendo però già a disposizione la visualizzazione di default legata a *Mechanics Explorer* (tool di Simscape Multibody), la precedente non risulta di particolare interesse.

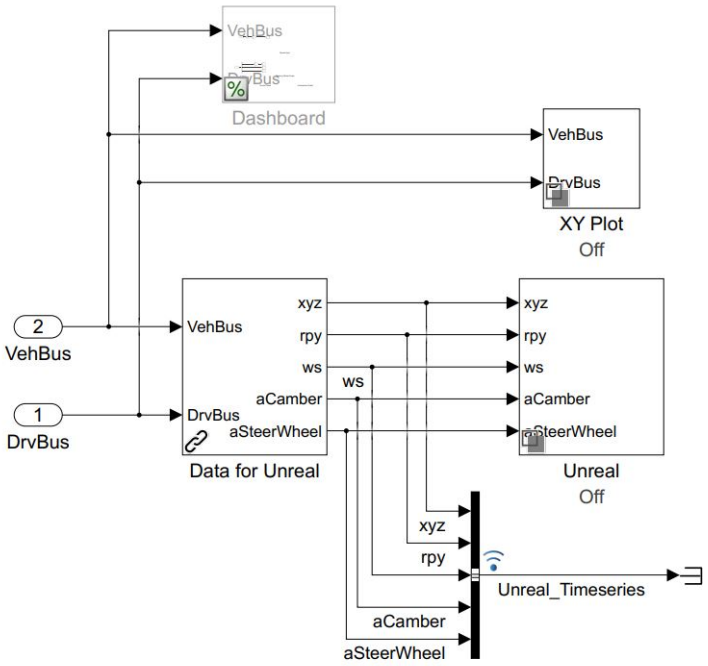


Figura 12: Interno blocco di visualizzazione della simulazione

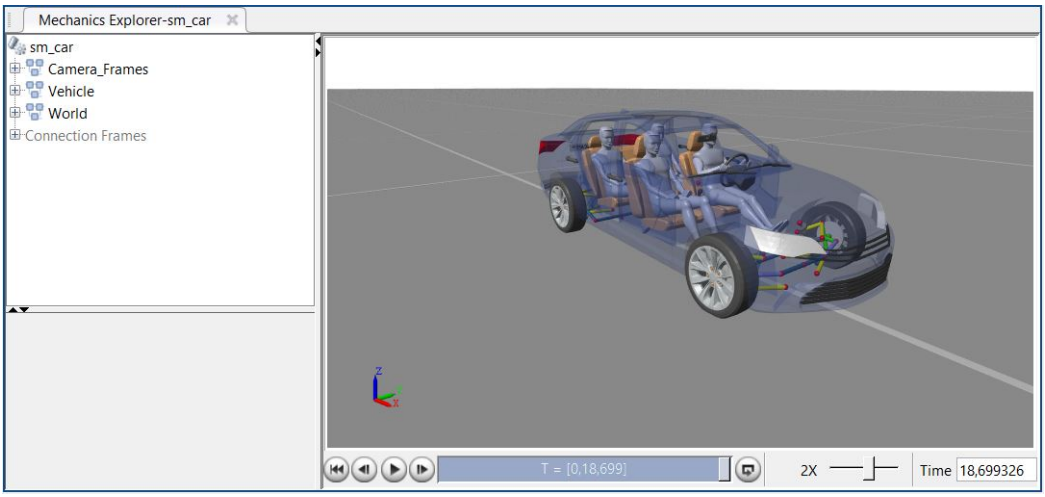


Figura 13: Visualizzazione standard in ambiente Simscape Multibody

Si fa notare, riferendosi a Figura 13, la possibilità di selezionare sistemi di riferimento e viste degli specifici componenti. Questo ha chiaramente valore nell'ottica di comprendere se, effettivamente, il veicolo stia seguendo la traiettoria che gli è

stata prefissata (ancora prima di agire con un eventuale *Post Processor*).
Risulta, tuttavia, di grande interesse sottolineare un'ulteriore modalità di visualizzazione della simulazione, sia contemporanea che successiva alla stessa. Questa procedura è caratterizzata dallo sfruttare l'*Unreal Engine*, motore grafico fornito gratuitamente da *Epic Games*, tramite cui è possibile simulare in questo specifico ambiente, largamente sfruttato nell'industria dei videogiochi, quanto precedentemente osservato nelle varie viste del Mechanics Explorer.

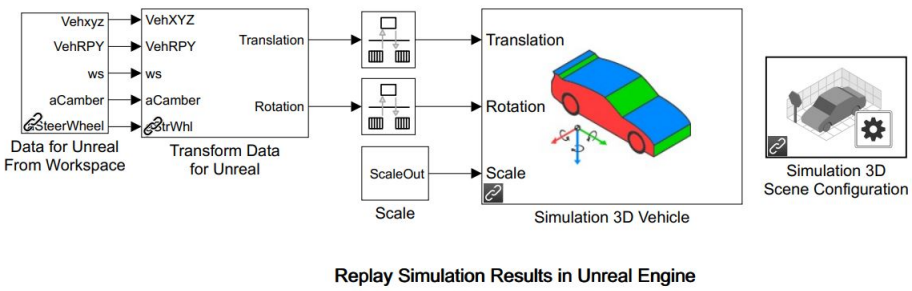


Figura 14: Modello Unreal Engine presente su Simscape Vehicles

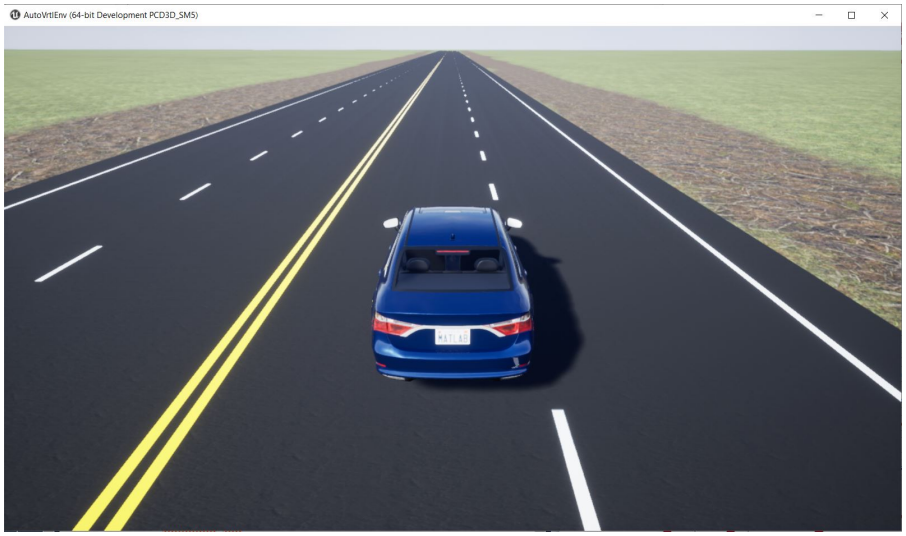


Figura 15: Visualizzazione di generica berlina in Unreal Engine

Dopo aver scaricato il motore di gioco dal sito del produttore, è stato possibile

provare effettivamente la sua efficacia. Come anticipato, questo è utilizzabile sia per mostrare la simulazione in tempo reale che in post-simulazione. Mentre il primo aspetto risulta essere interessante, ma principalmente da un punto di vista di risoluzione grafica, il secondo apre invece numerose opportunità. Infatti, per quanto al momento sia una semplice replica dei risultati ottenuti in precedenza (personalizzandone eventualmente la veste grafica, ma rimane una semplice animazione), si segnala la possibilità di intervenire in ambiente Unreal per modificare ad esempio la conformazione del manto stradale o le condizioni dello stesso, potendo valutare così la reazione del veicolo precedentemente creato alle nuove condizioni da un punto di vista, ad esempio, di rollio, imbardata e beccheggio. Si evidenzia che quanto sopra descritto è stato visto in maniera molto superficiale in questa trattazione, ma potrebbe essere un punto di partenza interessante (per le maggiori possibilità e sviluppi dell'ambiente in questione) nel caso in cui si volesse ampliare il lavoro.

Check

Questo sistema risulta essere dedicato alla verifica della soddisfazione di eventuali condizioni limite (ad esempio su velocità massima o massima deviazione laterale accettabile), personalizzabili. Nel caso in cui queste dovessero essere soddisfatte, la simulazione verrebbe automaticamente interrotta.

Scopes

Il blocco in questione risulta essere di utilizzo estremamente semplice e adatto, ad esempio, alla valutazione immediata dell'andamento di alcune grandezze. In questa trattazione, considerato anche il *Post Processor* generato appositamente anche per osservare variabili non direttamente restituite dal programma, i blocchi presenti in Figura 16 non sono praticamente mai stati utilizzati.

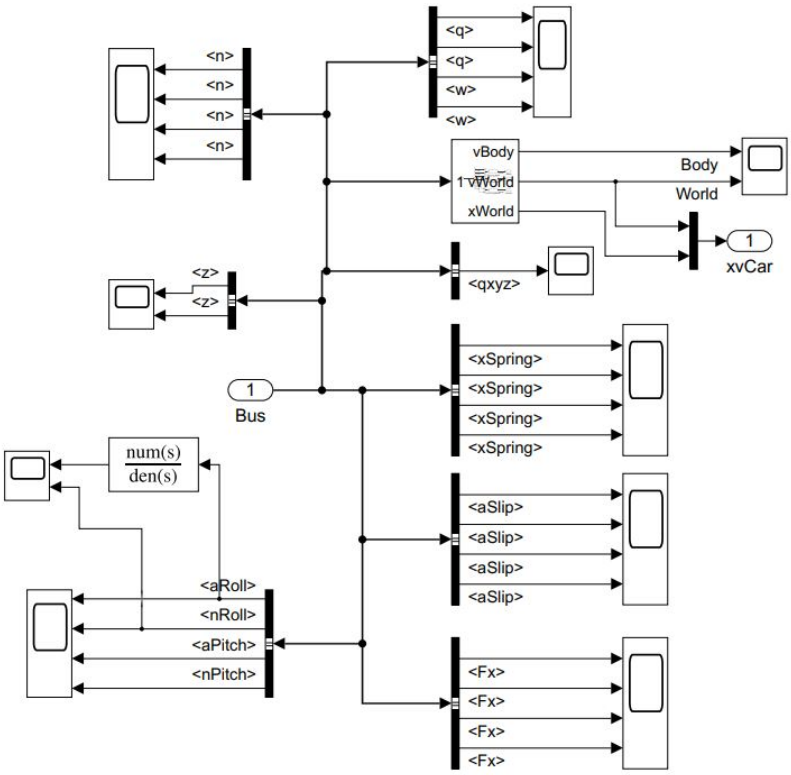


Figura 16: Interno blocco *Scopes*

Considerazioni

I sistemi descritti nelle pagine precedenti rappresentano uno dei cardini di questo lavoro di Tesi. In particolare, i blocchi *Driver*, *Road* e *Vehicle* hanno poi rivestito un ruolo centrale nell'attività di generazione dei dati. Risulta quindi di grande interesse capire come si sia effettuata principalmente l'interazione con gli stessi, introducendo la UI del sistema.

UI

La *User Interface* (UI) è una componente fondamentale del lavoro qui presentato. Tramite essa, infatti, è possibile interagire comodamente con i modelli generali visti nelle figure 1 e 2, personalizzando e componendo gli stessi. Si è scelto di parlare di composizione perché, a tutti gli effetti, tramite la UI è possibile selezionare, cliccando

sull'apposita parte dell'interfaccia, ogni variante a disposizione per caratterizzare il veicolo. Questo processo corrisponde, nella pratica, ad uno switch tra i blocchi messi a disposizione da *Simscape Vehicle Templates*, con la sostituzione del precedente con quello richiamato dalla UI, dando modo effettivamente di muoversi dinamicamente tra le librerie messe a disposizione del programma. Si ha quindi la facoltà di generare (nei limiti degli attuali blocchi a disposizione, che possono però essere integrati in futuro) numerosi modelli Simulink, basati a loro volta su modelli fisici delle varie componenti, a seconda di quelle che possono essere le esigenze di studio. Si fa notare che il funzionamento dell'interfaccia si basa su un livescript Matlab, su cui occorre necessariamente intervenire per apportare eventuali modifiche (come verrà presentato successivamente.
A livello di struttura, la UI attualmente si compone di cinque sezioni principali:

- *Presets*
- *Configure*
- *Matrix*
- *Event*
- *Results*
- *Main*

Si porta all'attenzione che, in questo lavoro di Tesi, ci si è concentrati principalmente, soprattutto in un'ottica di integrazione e ampliamento delle capacità della UI, sulle sezioni *Event*, *Results* e *Main*. Ogni elemento caratteristico dell'interfaccia, inoltre, è caratterizzato dall'avere un pulsante (*Code* o *Excel*) con rimando ai codici/file Excel richiamati dalla specifica funzione utilizzata. Sono, infine, presenti commenti per ognuno degli aspetti trattati, successivamente integrati e implementati anche per le modifiche che verranno presentate in seguito.

Presets : Questa sezione risulta essere particolarmente importante per effettuare, nell'ottica di effettuare la scelta preliminare del veicolo a disposizione, se si sceglie di non creare un modello completamente personalizzato (vedi paragrafo sottostante, *Configure*). Da qui infatti è possibile selezionarne la tipologia (berlina, berlina grande, bus e camion), decidendo anche elementi quali la presenza di un trailer, la tipologia di alimentazione del motore etc. Pur non essendo intervenuti per modificare questa parte di UI, la stessa è risultata molto importante per questa trattazione, avendo utilizzato (come si vedrà in seguito) il modello di default del Truck (privandolo semplicemente della presenza del trailer).

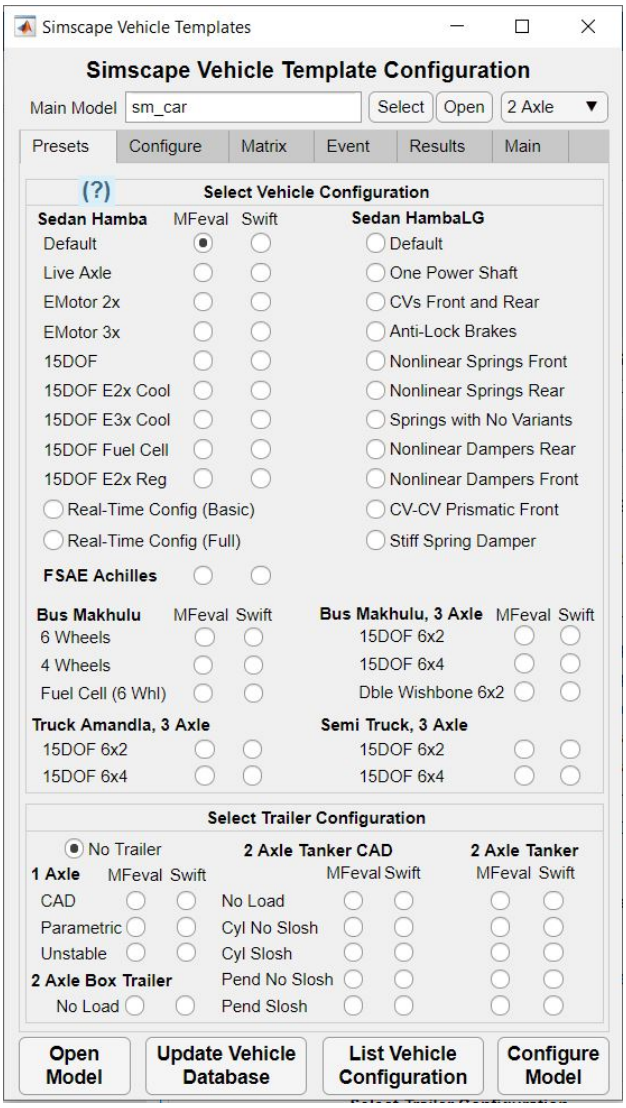


Figura 17: Modelli di veicolo immediatamente a disposizione

Configure : in questa sezione, usata in maniera marginale nella trattazione, viene fornita la possibilità di creare ex novo un modello di veicolo, sfruttando la logica a blocchi dinamici descritta in precedenza. Si sottolinea l'opportunità di ricorrere a una soluzione "ibrida" tra questo paragrafo e quello precedente (*Presets*), ovvero selezionando semplicemente il nuovo modello di interesse (ad esempio per quanto riguarda i motori, *Power*), che verrà quindi implementato al posto di quello previsto di default per il veicolo selezionato. È importante evidenziare che i parametri

presenti in questa sezione non cambiano automaticamente a seconda delle scelte fatte nei *Presets*, mostrano sempre una visualizzazione di default a livello di UI.

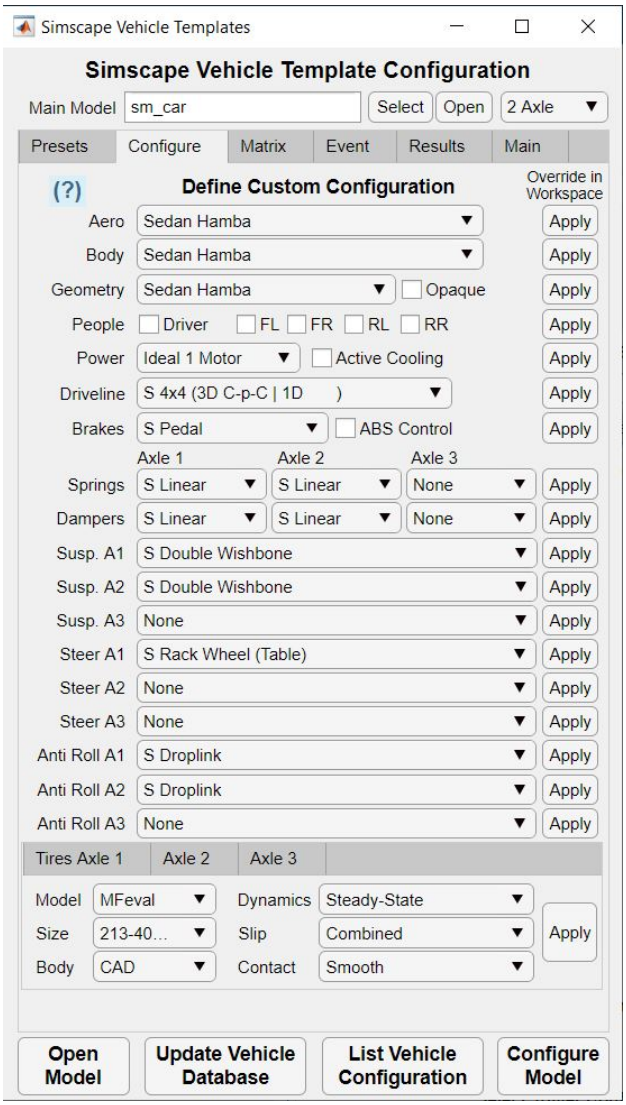


Figura 18: Possibilità di personalizzazione del veicolo

Risulta essere, infine, di interesse sottolineare che è da qui che dovrebbe essere possibile cambiare dimensione e, soprattuto, tipologia degli pneumatici da quelli di default *MF-eval* a quelli *MF-Swift* o *Delft*, indicati come necessari per poter simulare correttamente numerose manovre (parte centrale e fondamentale di questo lavoro). Tuttavia, le librerie e i modelli STL non sono presenti nel programma e

risulta essere necessario chiederli al costruttore.

Matrix : Sezione inutilizzata in questa trattazione, permette di scegliere tra varie combinazioni di pneumatici, driveline e tipologia di soluzione dinamica (stazionario, transitorio...).

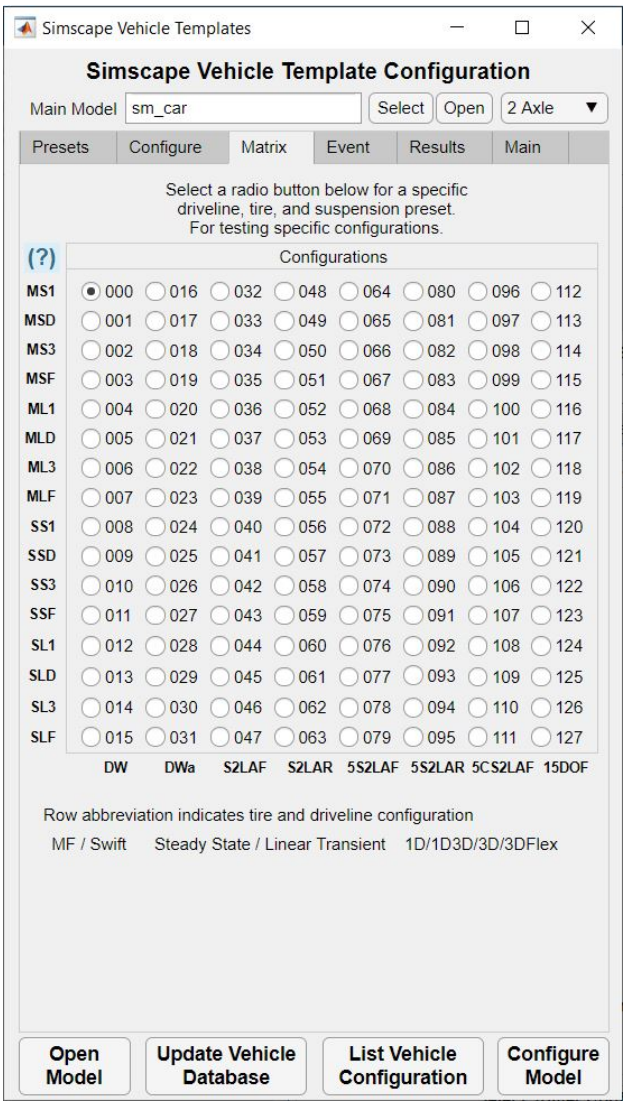


Figura 19: Possibili combinazioni di pneumatici e soluzioni dinamiche

Event : Questa parte della UI risulta essere stata, come antipato in precedenza, quella principalmente coinvolta da modifiche ed integrazioni a livello di codici e

livescript insieme a *Results* e *Main*. All'interno della stessa è possibile definire la manovra da far compiere al veicolo (e, contemporaneamente, la superficie, visto come agisce il codice *Configure Event*, di cui si parlerà più completamente poco più avanti in *Generazione dati*), i tipi di *Driver* e *Solver* (di per sé definiti automaticamente sempre in *Configure Event*), oltre a tutta una serie di parametri come vento, animazione in Unreal Engine etc.

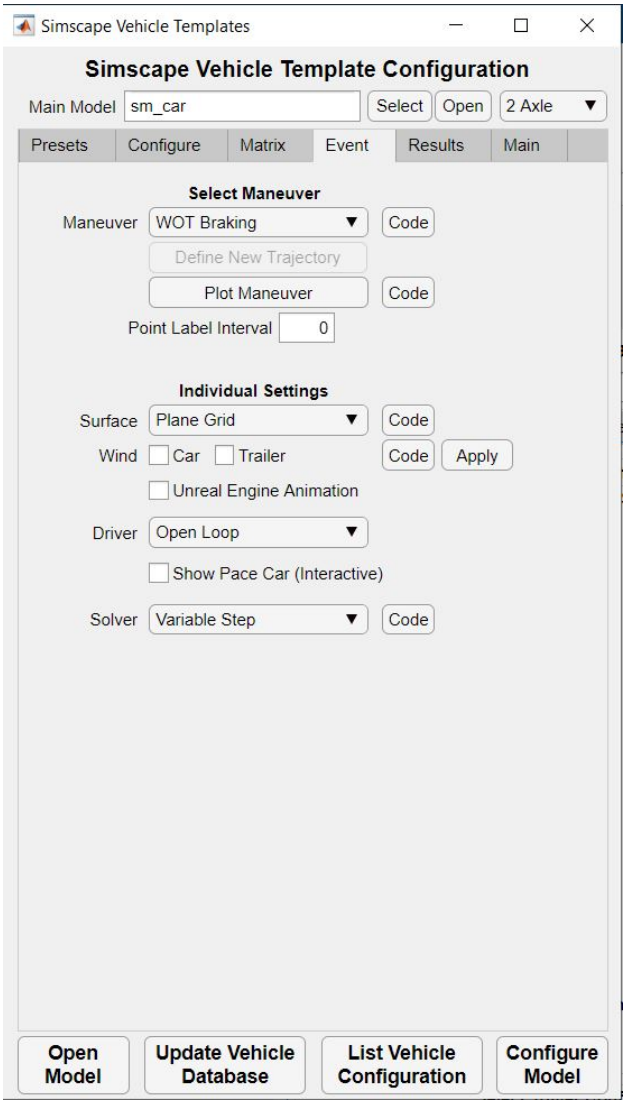


Figura 20: Scelta degli eventi cui sottoporre il veicolo

Results : Anche questa sezione, come quella precedente, è stata fortemente integrata nei passaggi successivi (in particolare in *Elaborazione dati*). All'interno della stessa, originariamente (Figura 22), era possibile osservare a livello grafico i risultati forniti dal programma tramite *logsout* post-simulazione, con il problema però di poter accedere alla maggior parte degli stessi solo tramite il *Simscape Results Explorer*, oltre ad avere forti difficoltà nel salvataggio diretto ed efficace dei medesimi. Si segnala, peraltro, l'assenza di alcuni elementi di interesse per un'analisi dinamica del veicolo, quali ad esempio la differenza tra l'angolo di sterzo dinamico e cinematico. Viene già anticipato che questi problemi sono stati risolti introducendo (e implementando anche su UI) un apposito codice di *Post Processor*.

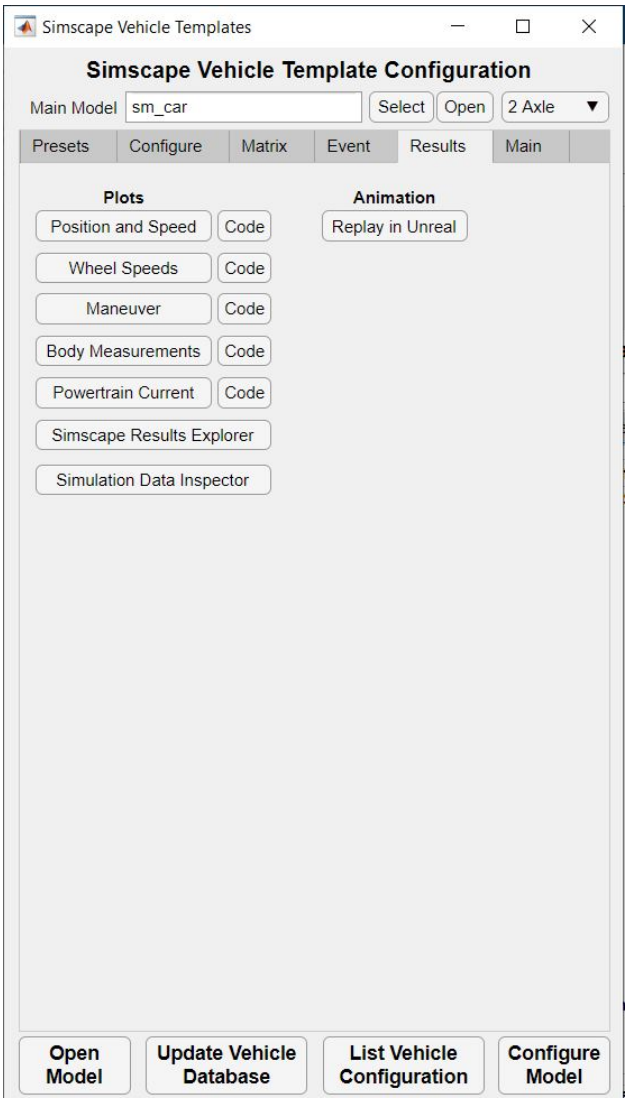


Figura 21: Risultati di simulazione

Main : L'ultima parte della UI è quella essenziale per il funzionamento della stessa. All'interno di questa sezione, infatti, è possibile caricare e rigenerare i vari database necessari per la corretta esecuzione del programma. Si evidenzia che questi sono composti da codici ed Excel, su cui è obbligatorio intervenire nell'ottica di apportare eventuali modifiche a quanto preimpostato dal programma originale. Ovviamente si consiglia di intervenire in questa sezione solo a seguito di cambiamenti a librerie e/o codici, come vedremo in *Generazione dati*.

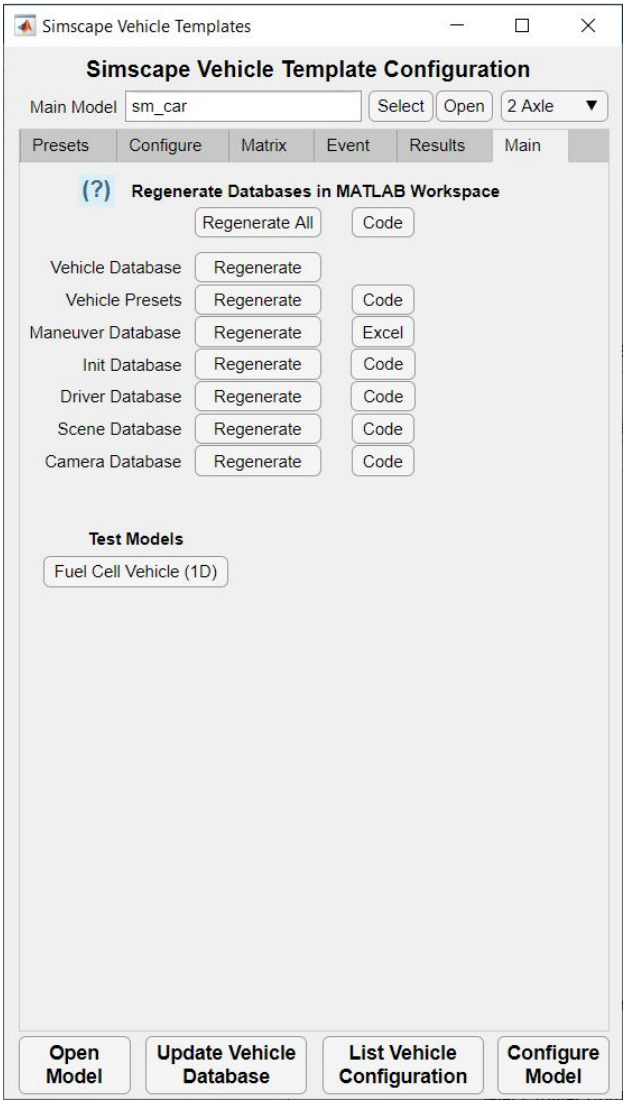


Figura 22: Caricamento e rigenerazione dei Database per funzionamento della simulazione

Considerazioni UI

Tramite la *User Interface* a disposizione, è possibile interagire in maniera semplice ed efficace con l'intero ambiente di programmazione, passando dai codici e dai file Excel agli effettivi blocchi dinamici caratterizzanti il modello in questione. Come anticipato questo è reso possibile tramite un livescript su cui si vedranno esempi di intervento nelle sezioni di questa trattazione specificatamente dedicate alla modifica della UI.

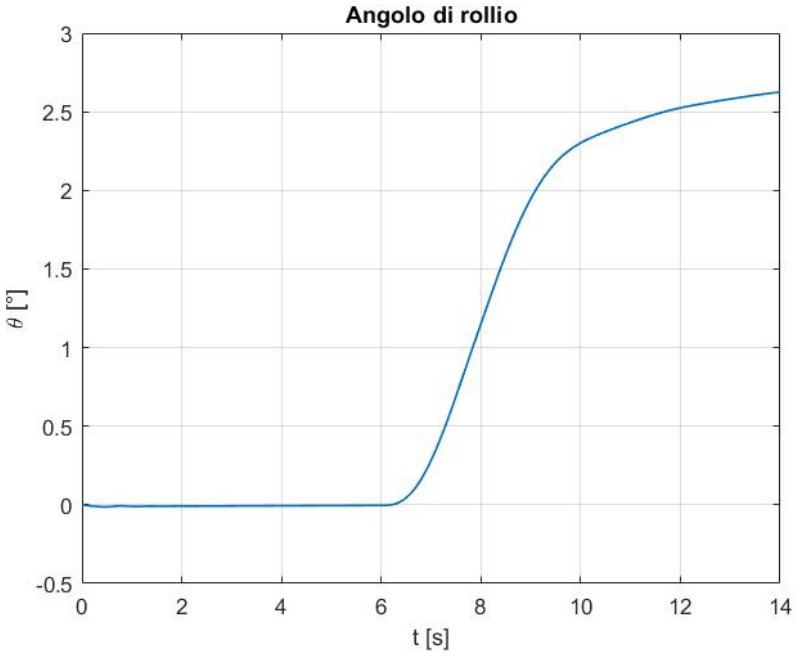
Generazione ed elaborazione dati

Dopo la presentazione del programma utilizzato per questa prima parte di Tesi, possiamo finalmente passare a quanto svolto. Il processo in questione si articola in quattro fasi principali, ovvero:

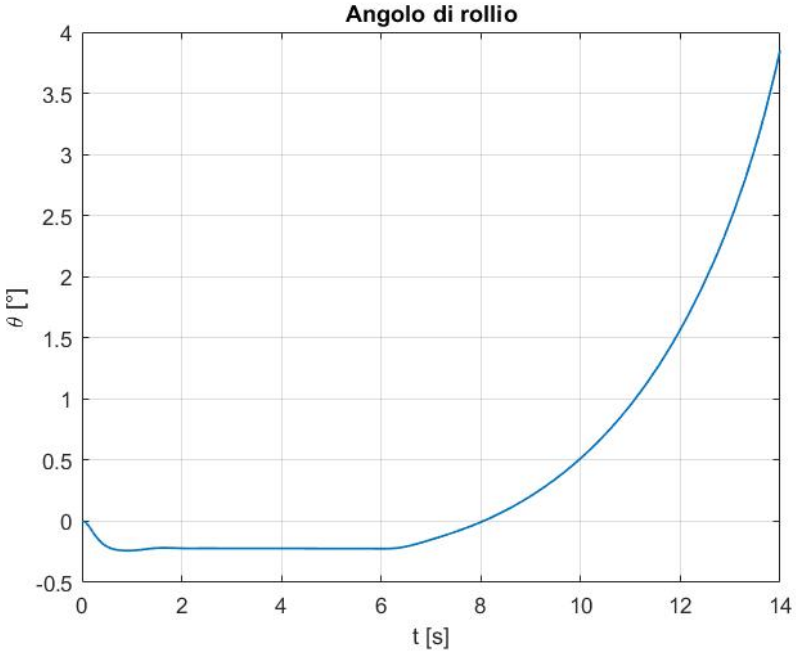
- *Post Processor*
- *Post Processor: UI*
- *Scelta del veicolo e utilizzo e creazione manovre*
- *Criteri per identificazione rischio di rollover e dataset*

Post Processor e scelta del veicolo

Come anticipato nel capitolo introduttivo, obiettivo di questo lavoro è lo studio del fenomeno di ribaltamento per i mezzi pesanti. Questi sono caratterizzati dall'avere dimensioni notevoli con alti baricentri, elementi che portano al rischio concreto del verificarsi di una condizione di rollover anche con manovre che, per vetture più piccole (o semplicemente più basse), non comporterebbero problemi.
In quest'ottica, viene sotto presentato un confronto tra gli andamenti dell'angolo di rollio tra il modello di un Truck e uno di berlina (entrambi di default e privi di trailer), cui è stato richiesto di eseguire esattamente la stessa manovra in OL (nello specifico una ramp steer, manovra a chiocciola, di cui parleremo più diffusamente nella sezione *Creazione manovre e superfici*).



(a) Andamento angolo di rollio Sedan



(b) Andamento angolo di rollio Truck

Figura 23: Confronto andamenti dell'angolo di rollio di una Sedan e un Truck cui viene richiesto di eseguire la stessa manovra

Riferendosi alle figure soprastanti, risulta evidente come i due andamenti differiscano sostanzialmente, con quello legato al camion che presenta una curva di aumento vertiginosa dell'angolo di rollio (da verifiche sperimentali, portando avanti la prova per altri 3-4 secondi si sarebbe ottenuta una condizione di ribaltamento del mezzo). Da qui l'interesse nel definire manovre opportune per ciascuna tipologia di veicolo. È importante, peraltro, evidenziare che i due diagrammi in Figura 23 sono stati ottenuti tramite un codice *Post Processor*. Si è deciso di implementare quest'ultimo nel programma in modo da avere immediatamente a disposizione gli andamenti delle grandezze di interesse (poco meno di una quarantina, per larga parte estratte e manipolate adeguatamente per mostrare le corrette unità di misura) e avere la possibilità di salvarli con estrema semplicità (basta cambiare un indice a inizio script da 0 ad 1). L'efficacia dello stesso (e intrinsecamente anche quella dell'ambiente di simulazione) può essere valutata tramite alcuni esempi facilmente riconoscibili riferiti per una vettura appartenente alla classe sedan (berlina), come osservabile nelle figure 24 e 25, ottenute con la stessa manovra riportata nell'immagine 23.

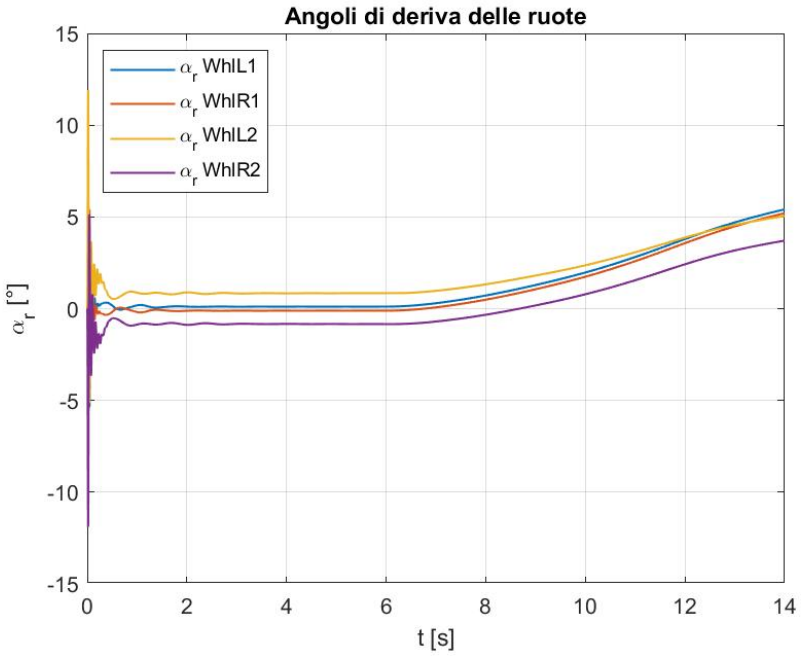


Figura 24: Angoli di deriva di una berlina in caso di manovra a chiocciola

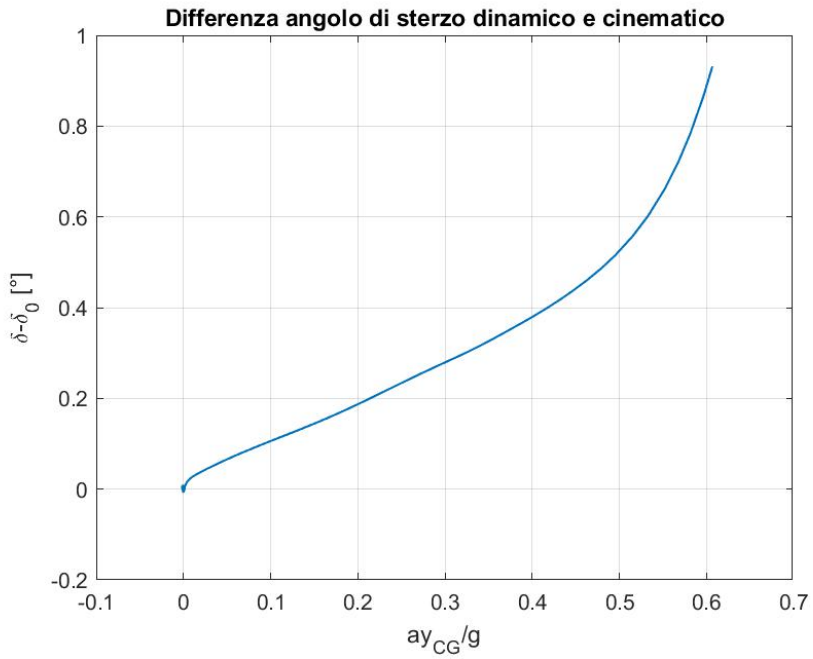


Figura 25: Differenza angolo di sterzo dinamico e cinematico berlina in caso di manovra a chiocciola

Riferendoci alle immagini sopra riportate, risulta chiara l'efficacia del programma, che permette di ottenere dei risultati molto simili a quelli presenti in teoria ((Mauro Velardocchia, 2020)). Si evidenzia che nella parte iniziale in Figura 25 si ha una prima parte di assestamento, questa risulta molto breve grazie al fatto che si è deciso di avviare la manovra effettiva dopo una fase di sei secondi di moto rettilineo accelerato del veicolo (più semplice da gestire per il programma, per quanto sperimentato, rispetto ad un inizio con angolo di sterzo fornito immediatamente). Quanto detto è ancora più facilmente riscontrabile in Figura 24. Si ricorda, infatti, che il legame tra le due immagini è dettato dalla relazione (1), dove α_{rA1} ed α_{rA2} rappresentano la media aritmetica, rispettivamente, tra gli angoli di deriva delle ruote dell'assale anteriore e di quello posteriore.

$$\delta - \delta_0 = \alpha_{rA1} - \alpha_{rA2} \tag{1}$$

È importante evidenziare, inoltre, che le figure 24 e 25 rappresentano la prima

una semplice estrazione dal *logsout* delle variabili di interesse (gli angoli di deriva delle ruote), la seconda una versione elaborata delle stesse nell'ottica di ottenere nuove informazioni. Allo stesso modo, sfruttando le medesime grandezze in uscita dalle ruote, è stato poi possibile tracciare l'andamento della differenza fra angolo di assetto dinamico e cinematico (Figura 26), basata sull'equazione (2).

$$\beta - \beta_0 = -\alpha_{rA2} \tag{2}$$

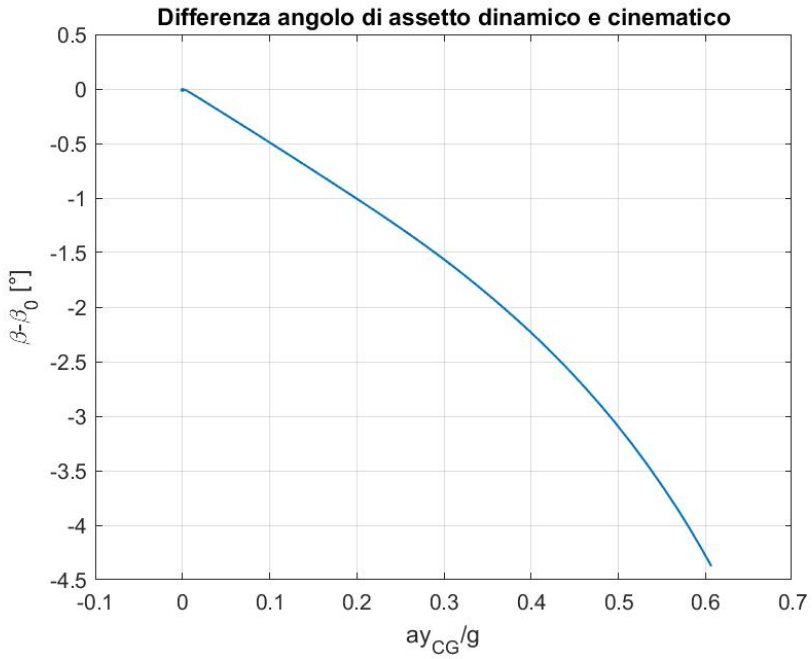


Figura 26: Differenza angolo di assetto dinamico e cinematico berlina in caso di manovra a chiocciola

I risultati sopra presentati contribuiscono a mostrare il fatto che gli stessi presentano esiti logici e sensati, motivando la scelta di effettuare simulazioni tramite il programma in questione e avvalendosi del codice *Post Processor* per analizzarne ed elaborarne le variabili in uscita. Si è scelto di mostrare alcuni risultati introduttivi utilizzando una berlina e non un camion (veicolo centrale della trattazione) sia perché per lo stesso ne verrano presentati diversi in seguito, sia per mostrare alcuni diagrammi caratteristici immediatamente riconoscibili per i veicoli a due assali (nel

caso del truck, tre assali, i grafici sopra riportati sono ugualmente disponibili, ma di comprensione meno immmediata).
Prima di procedere con la presentazione della definizione delle manovre (senza le quali il codice risulterebbe relativamente fine a sé stesso in questa trattazione), si presenta l'implementazione dello script da avviare post-simulazione a livello di interfaccia utente.

Post Processor: UI

Come anticipato, l'intervento sull'UI è stato effettuato per tutti codici centrali di questo lavoro di Tesi in modo da dare possibilità all'utente di interagire con grande facilità con gli stessi, senza avere necessità di doverli andare a cercare specificatamente nelle librerie in cui sono stati collocati. Per poter effettuare questo tipo di implementazione, è stato necessario muoversi all'interno degli script originali del programma, di cui verranno presentate le cartelle poco più avanti. In particolare, per quanto concerne l'introduzione a livello di interfaccia, si è dovuto seguire il percorso *Scripts_Data\Data_Vehicle\UI* e selezionare il codice *sm_car_vehcfg*, accedendo a quanto è possibile osservare in Figura 27. Chiaramente, si evidenzia fin d'ora che, per motivi di brevità, si è scelto di non commentare ogni singolo codice presente in ogni cartella del programma originale, altrimenti sarebbe stata necessaria aprire una trattazione a sé stante (in questa vengono messi in evidenza i cambiamenti principali effettuati).

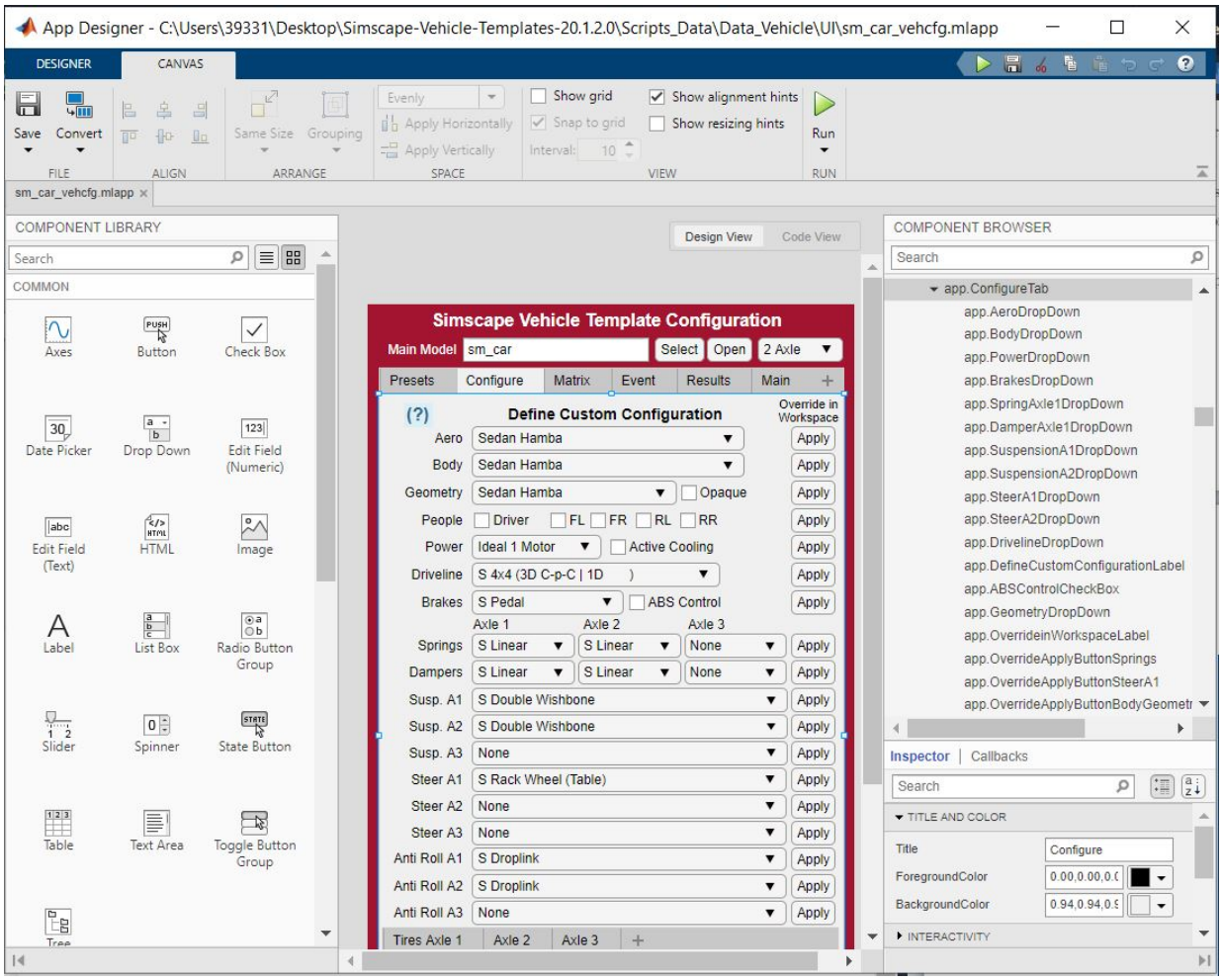
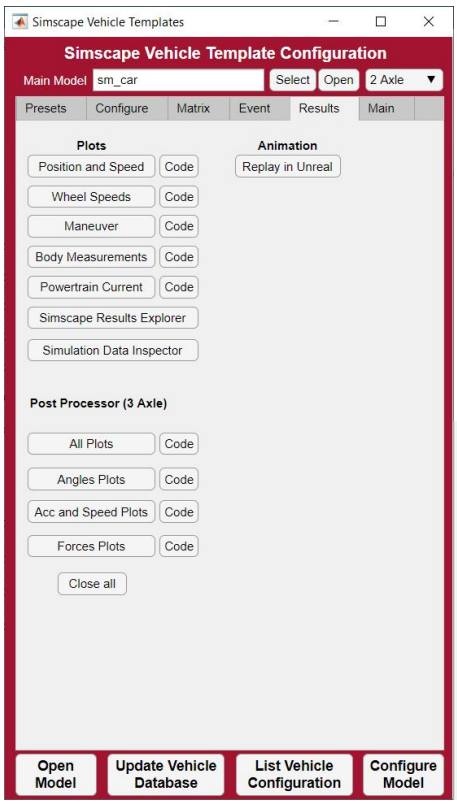


Figura 27: Editor dell'interfaccia utente

Considerando l'immagine di cui sopra, se ne evidenziano alcuni aspetti. Risulta infatti possibile, una volta entrati nell'editor in questione, implementare nuove funzioni per l'interfaccia di interesse. In particolare, è possibile aggiungere nuovi blocchi (a sinistra della figura), definendone poi le caratteristiche all'interno delle sezioni *Inspector* e *Callbacks*. Quest'ultima risulta poi essere di notevole importanza, in quanto quella di richiamo al codice. Una volta creato un nuovo pulsante sulla UI, occorre infatti, dopo averne definito i diversi aspetti preparatori nell'*Inspector*, creare una funzione a livello di livescript (specifica per ciascun blocco) in grado di collegare effettivamente l'interfaccia ai codici di interesse (compito, appunto, della

sezione *Callbacks*). Riferendoci allo specifico intervento effettuato per introdurre nell'UI la possibilità di interagire facilmente con i codici di *Post Processor*, vengono riportate di seguito due immagini in cui viene mostrata l'immagine dell'interfaccia utente modificata e il livescript corrispondente a livello di app (accessibile, come detto, sfruttando la sezione *Callbacks*).



(a) Prima modifica UI: *Post Processor*

```
1660        % Button pushed function: AllPlotsButton
1661        function AllPlotsButtonPushed2(app, event)
1662            str = 'Post_processor_TA_all_plots';
1663            disp(str);
1664            evalin('base',str);
1665        end
1666
1667        % Button pushed function: PowertrainCurrentCodeButton_2
1668        function PowertrainCurrentCodeButton_2Pushed(app, event)
1669            edit('Post_processor_TA_all_plots.m');
1670        end
1671
1672        % Button pushed function: CloseallButton
1673        function CloseallButtonPushed(app, event)
1674            str = 'Post_processor_TA_close_all';
1675            disp(str);
1676            evalin('base',str);
1677        end
```

(b) Particolare livescript UI

Figura 28: Modifica dell'interfaccia utente per introduzione *Post Processor* e codice corrispondente

Come possibile osservare in Figura 28, sono stati introdotti cinque nuovi pulsanti legati al *Post Processor*. Particolarmente importante risulta essere il primo, caratterizzato dal permettere l'elaborazione grafica di tutti i diagrammi di interesse (i pulsanti dopo altro non sono che dei codici parziali ricavati da quello generale) e il salvataggio dei risultati di una determinata manovra (in collaborazione con un altro codice, come vedremo in seguito). È possibile poi interagire con ognuno degli script

coinvolti tramite il blocco *Code* (corrispondente alla funzione *edit* del livescript), ad eccezione del pulsante *Close all*, rimandante a un codice caratterizzato dalla funzione esclusiva di chiudere le numerose figure generate (è comunque possibile accedervi, dato che si trova nello stesso percorso degli altri file). L'interazione con i singoli codici può essere utile per modificare gli stessi o, anche solo, attivare l'indice di salvataggio delle singole figure (presente all'inizio di ogni specifico script). Infine, si porta all'attenzione che, in Figura 28 si può notare come il codice post-simulazione implementato sulla UI sia definito per un veicolo a tre assali. Ne è stato sviluppato uno anche per un veicolo a due assali, ma non essendo questa tipologia di mezzi oggetto principale di questo argomento, non è stata pensata la presenza di un'alternativa anche per i medesimi (per quanto non di difficile integrazione).
Evidenziata l'utilità del codice di *Post Processor* risulta, chiaramente, interessante valutare per cosa e quale motivo effettivamente si utilizzi. È stato anticipato che scopo della trattazione è lo studio del comportamento di mezzi pesanti da un punto di vista di rischio di ribaltamento, portando, in Figura 23, una prima situazione di interesse. Diventa chiaro, però, che non ci si può limitare allo studio di un'unica manovra, anche perché, come si vedrà nel capitolo dedicato all'IA di questo lavoro di Tesi, le reti neurali necessitano di avere numerosi dati a disposizione, che è uno dei motivi per cui si ricorre a dati, per quanto realistici, derivanti da una simulazione. Conseguentemente, parte assolutamente centrale della trattazione è stata la definizione di nuove manovre da fare compiere al mezzo scelto, oltre allo sfruttamento di quelle preesistenti (messe subito a disposizione dal programma).

Scelta del veicolo e utilizzo e creazione manovre

Dopo aver presentato creazione, implementazione e primo utilizzo di un codice *Post Processor*, come detto occorre evidenziare per cosa lo si stia utilizzando.

Scelta del veicolo : dovendo studiare, come riportato più volte, un metodo in grado di identificare il rischio di ribaltamento per mezzi pesanti, il veicolo principe per questa trattazione risulta essere un truck, caratterizzato da un baricentro alto (eventualmente modificabile, come tutte le caratteristiche geometriche del veicolo) e dalla presenza di tre assali. Si sottolinea che si è scelto di utilizzare la versione di default (15 DOF 6x2) fornita da *Simscape Vehicle Templates*, privando però il mezzo del trailer originariamente previsto.

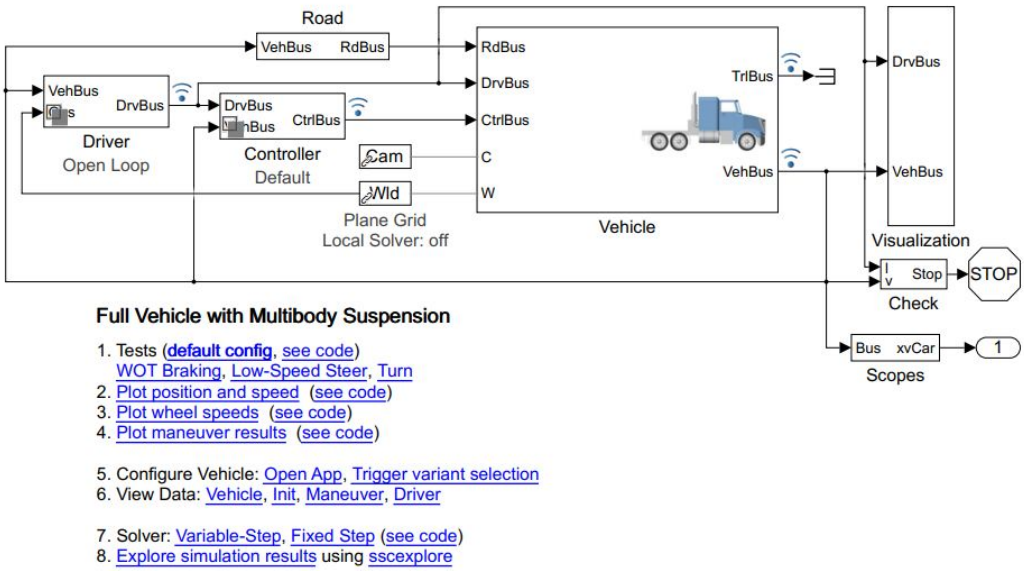
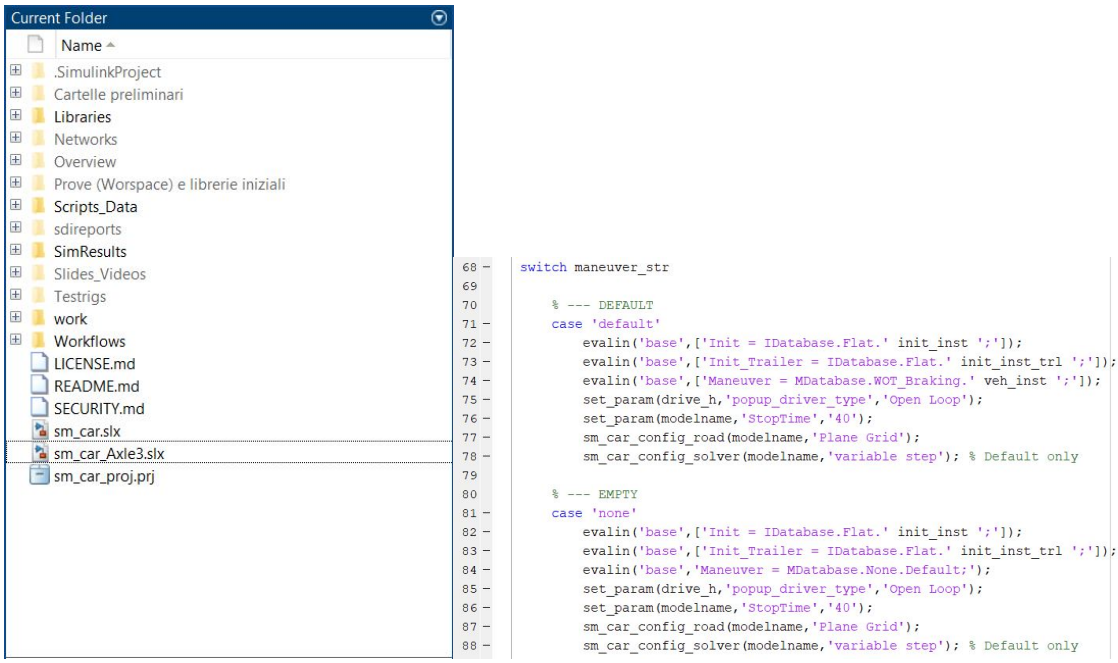


Figura 29: Truck utilizzato

Maneuver : Scelto il mezzo, occorre definire quale traiettoria richiedergli, su quale superficie e con quale tipologia di *Solver* e *Driver* affrontare il problema. È possibile impostare tutto ciò interagendo a livello di UI con la sezione *Event*, in particolare con il pulsante *Maneuver* e, a livello di script, con il codice *sm_car_config_maneuver*, accessibile tramite il percorso *Scripts_Data\Configure_Event*. Per permettere meglio la comprensione, viene sotto presentata una vista delle cartelle utilizzate per questo progetto, in unione con un esempio di codice di manovra estratto dallo script sopra citato.



(a) Cartelle Simscape (b) Particolare del codice del maneuver

Figura 30: Librerie Simscape ed esempio di codice di definizione delle manovre

Come osservabile nell'immagine a destra di Figura 30, il codice in questione, avviato tramite il pulsante *Maneuver* della UI, permette di definire i parametri essenziali per poter avviare la simulazione e sfruttare quindi il *Post Processor* per osservare e iniziare a raccogliere i primi dati di interesse. In modo da facilitare maggiormente la comprensione di quanto fatto e possibili interventi futuri, viene ora descritto il significato delle righe dello script sopra riportato e, soprattutto, i legami tra le varie componenti delle stesse (ovvero le interazioni tra Workspace di Matlab, codici, Excel e UI).

Le prime due righe hanno funzione di inizializzare i parametri di posa (ovvero posizione, in metri, e angolazione, in radianti) e velocità sia del veicolo, che dell'eventuale rimorchio selezionato, in quella che è a tutti gli effetti la procedura di avvio di due stringhe. All'interno di questa tipologia di scrittura (*evalin*) vengono richiamati dei database, generati dal programma in fase di apertura dello stesso (osservabili nel Workspace di Matlab) e modificabili tramite la sezione *Main* dell'interfaccia

utente (cliccando sul solito pulsante *Code* o su quello *Excel*, come vedremo tra poco). In particolare, per quanto riguarda queste due righe iniziali, si fa riferimento all'*IDatabase*. Questo fornisce, per ciascuna manovra desiderata (in genere quelle con *Driver* OL si trovano nella categoria generale *Flat*) i parametri di posa e velocità citati in precedenza (che vengono mandati al Workspace del programma), generando una specifica struttura nel Workspace, *Init*. Si evidenzia che, volendo, per attribuire ad una specifica manovra dei dati di un'altra, è sufficiente sostituire, riferendoci per semplicità alla Figura 30, ad esempio *.Flat* con *.Rough Road*, ottenendo i parametri citati descritti in precedenza, ma con valori pensati per una manovra (e una superficie) differente. Nel caso in cui si fosse interessati a cambiare, sempre permanentemente, i valori di posa e velocità in questione per una specifica tipologia di manovra (e.g. *.Flat*), sarebbe sufficiente aprire la UI, che evidenziamo essere direttamente collegata al modello Simscape in Figura 29 tramite il collegamento 5 (*Open App*), andare nella sezione *Main* e premere sul pulsante *Code*. A questo punto si avrebbe accesso ai dati caratteristici dell'*IDatabase* e, una volta effettuata la modifica, salvare il tutto e, tornando sulla UI, premere sul pulsante *Regenerate* per implementare effettivamente le modifiche. Questa operazione è stata provata a scopo sperimentale in questa trattazione ed è possibile garantire che, per quanto riguarda l'aggiornamento dei codici, il tempo di computazione è estremamente ridotto. Chiaramente, le azioni descritte a livello di interfaccia corrispondono a precisi codici implementati a livello di livescript sulla stessa (come visto in precedenza).
La terza riga che andiamo ad esaminare è quella dove effettivamente viene richiamata la manovra desiderata, contenuta all'interno del *MDatabase*. L'interazione con lo stesso segue gli stessi principi e passaggi descritti per l'*IDatabase*, con la grande differenza che, in questo caso, i dati delle manovre non sono forniti tramite script Matlab, ma attraverso fogli Excel (su cui si suggerisce di intervenire senza aprire il programma, più rapido), che devono poi essere convertiti in strutture leggibili dal software. Interagendo con gli stessi è possibile modificare manovre

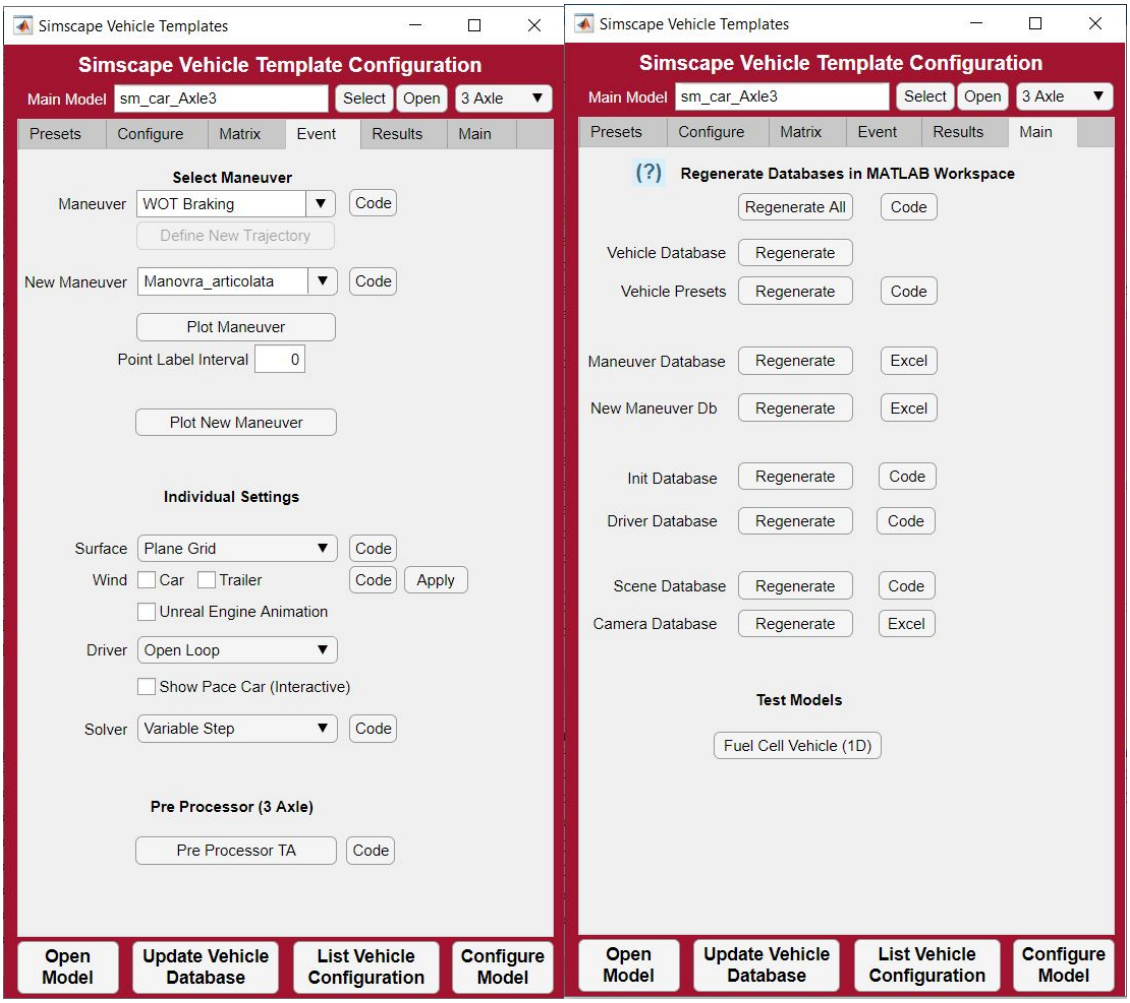
esistenti o generarne di nuove, facendo attenzione però a implementare le stesse, a livello nominativo, dentro il corrispettivo *Inspector* correlato al pulsante *Maneuver* della UI, da dove è effettivamente possibile cambiare (utilizzando una funzione di switch) tra una manovra e l'altra prima di avviare la simulazione (come verificabile osservando il prompt sulla Command Window in ambiente Matlab), ottenendo poi i parametri selezionati nella struttura del Workspace *Maneuver* (che, chiaramente, si aggiorna ogni volta che si cambia scelta). È molto importante fare notare che, la rigenerazione dei fogli Excel, vista la numerosità degli stessi, dovuta anche al fatto che ogni singola manovra presenta valori di posizione pedale freno (variazione parametri tra 0 e 1), posizione pedale acceleratore (sempre 0-1) e angolo di sterzo del volante (il programma non implementa però un rapporto di sterzo) in caso di OL, e di parametri di posizione e velocità in caso di CL (riferendoci al discorso fatto sul blocco *Driver* nella presentazione del programma), *diversi* per i vari mezzi a disposizione (altrimenti con una manovra potrei avere una berlina che effettua il movimento senza problemi, e un truck che con la stessa si ribalta). Conseguentemente, il tempo di rigenerazione risulta essere non elevato, ma non rapido quanto quello caratterizzante i database definiti immediatamente a livello di codice (gli Excel in questione, nel processo di aggiornamento, vengono convertiti in file Matlab tramite apposito script).

Le ultime quattro righe risultano di facile spiegazione, dato che non coinvolgono interazioni molteplici, ma solo alcuni collegamenti tra ambienti comunicanti tra loro. La prima delle stesse, semplicemente, ha funzione di impostare la tipologia di *Driver* che verrà automaticamente passata come informazione al modello Simscape. Si fa notare che, nel caso in cui si definisse una manovra con *Driver* di tipo CL, risulterebbe necessario definire l'inizializzazione di un'ulteriore stringa, richiamante un database *DDatabase*, derivante da un codice che permeterebbe, come quelli visti in precedenza, di interagire con la UI e passare informazioni specifiche (relative alla scelta fatta) al programma fino ad arrivare alla definizione delle stesse sul Workspace

di Matlab (dove si ricorda che si trovano anche tutti i database di cui stiamo parlando e non solo le strutture, e.g. *Maneuver* o *Init*, derivanti dalle selezioni effettuate sui dataset definiti in precedenza). Si evidenzia che è possibile cambiare la tipologia del *Driver* anche tramite apposito comando della UI, ma se ciò dovesse essere fatto per manovre non pensate per lo stesso, si presenterebbe un messaggio di errore o la scelta in questione risulterebbe ininfluente.
Per quanto concerne le altre righe, una definisce il tempo di durata che si vuole dare alla simulazione (coincidente con quando si vuole fare finire la manovra), *Stop Time*, una, molto importante per questa trattazione, la superficie su cui si svolgerà la manovra (correlata ad apposito codice dove è possibile definire, ad esempio, i parametri di attrito ruote/terreno) e l'altra la tipologia di *Solver*. Quest'ultima può essere di vari tipi, in questo lavoro di Tesi si è scelto di adottare una tipologia di risolutore a passo variabile *ode23t* in quanto era quella che permetteva di ottenere i grafici e i diagrammi migliori.
Si evidenzia che, a livello di UI, Figura 20, è possibile definire anche condizioni aggiuntive rispetto a quelle riportate nel codice *sm_car_config_maneuver*, come ad esempio quelle del vento su veicolo e trailer. Da questo punto di vista, pur potendo implementare queste condizioni nello script di cui sopra e rendere automatico tutto il processo con un unico click, si è scelto di non usufruire della possibilità di aggiungere queste ulteriori variaizioni.

Creazione e utilizzo manovre : Risulta estremamente importante, alla luce di quanto detto, evidenziare quale sia stata la procedura seguita per l'utilizzo delle manovre esistenti e la creazione di nuove. Come presentato, centrale in questa trattazione è risultato il codice *sm_car_config_maneuver*, tramite cui è possibile definire gli aspetti caratteristici di quanto vogliamo far compiere al veicolo e l'ambiente in cui questo movimento si deve realizzare. Conseguentemente, basandosi su questo codice (e sui commenti fatti sul medesimo), ne è stato creato un altro

sm_car_config_New_maneuver, ritrovabile nello stesso percorso e contenente cinque manovre in grado di evidenziare efficacemente una condizione di rollio sostenuto e una di vero e proprio ribaltamento del camion (alla fine non utilizzata in quanto valida solo a livello simulativo). Questo nuovo script è stato implementato efficacemente sia in termini di interfaccia utente che in termini di interazione con l'ambiente di lavoro ed ha un proprio database dedicato, *New Maneuver Db*, creato anch'esso secondo i criteri già visti in precedenza e nell'ottica di risparmiare del tempo di computazione (se si ha l'intenzione di usare solo le manovre nuove non ha senso aggiornare anche quelle "vecchie"). Si fa notare che, per utilizzare il nuovo script, come logico, sia opportuno caricare (premendo sull'apposito pulsante *Regenerate* nella sezione *Main*) il *New Maneuver Db* perché, al momento, si è lasciato di default quello legato al codice iniziale. È importante evidenziare che, nell'ottica di un'efficace implementazione degli script e dei database in questione, si è intervenuti anche su altri codici contenuti in *Scripts_Data*, in maniera minore ma necessaria, quali ad esempio il codice *sm_car_gen_upd_database*.



(a) Modifica sezione *Event* UI (b) Modifica sezione *Main* UI

Figura 31: Introduzione a livello di interfaccia di nuovi codici e database

Osservando la parte sinistra di Figura 31, è possibile notare come sia stato implementato un ulteriore codice, *Pre Processor*. Questo, aggiunto secondo la medesima logica già presentata per il *Post Processor*, ha solo scopo di back-up a seguito di utilizzo dei codici che vedremo essere tipici della seconda parte della trattazione, che contengono dei *clear all* all'interno (per riuscire a muoversi ancora all'interno del Workspace). Funzione di questo *Pre Processor*, se attivato, è semplicemente quella di ricaricare nel Workspace le librerie caratteristiche del veicolo scelto (ovvero il truck),

in modo da permettere l'avvio di una qualsiasi simulazione senza avere necessità di riavviare il programma (rigenerare tutti i database sarebbe relativamente lungo, oltre che non sufficiente).

Si ricorda che tutte le manovre (e conseguentemente gli elementi correlati ai codici presentati precedentemente), siano esse nuove o vecchie, sono selezionabili tramite un semplice click sui blocchi *Maneuver* e *New Maneuver* e che ogni blocco aggiunto a livello di UI su questa interfaccia contiene dei commenti esplicativi, in modo da rendere il tutto il più possibile immediato.

```
1611        % Value changed function: NewManeuverDropDown
1612        function NewManeuverDropDownValueChanged2(app, event)
1613            value = app.NewManeuverDropDown.Value;
1614            model_name = app.MainModelEditField.Value;
1615            str = ['sm_car_config_New_maneuver(''' model_name ''',''' value ''');'];
1616            disp(str);
1617            evalin('base',str);
1618            app.ConfigureModelButton.BackgroundColor = app.highlightColor;
1619        end
1620
1621        % Button pushed function: ManeuverCodeButton_2
1622        function ManeuverCodeButton_2Pushed(app, event)
1623            edit sm_car_config_New_maneuver.m
1624        end
1625
1626        % Button pushed function: PlotNewManeuverButton
1627        function PlotNewManeuverButtonPushed(app, event)
1628            ManeuverValue = app.NewManeuverDropDown.Value;
1629            PointLabelIntervalValue = app.PointLabelIntervalEditField.Value;
1630            sm_car_plot_maneuver(evalin('base','Maneuver'),PointLabelIntervalValue);
1631        end
1632
1633        % Button pushed function: RegenerateManueverDatabaseButton_2
1634        function RegenerateManueverDatabaseButton_2Pushed(app, event)
1635            str = 'sm_car_gen_upd_database(''New_Maneuver'',0);';
1636            disp(str)
1637            evalin('base',str);
1638        end
1639
1640        % Button pushed function: ManeuverDatabaseFileButton_2
1641        function ManeuverDatabaseFileButton_2Pushed(app, event)
1642            sm_car_winopen_file('sm_car_database_New_Maneuver.xlsx')
1643        end
```

Figura 32: Livescript per il nuovo maneuver e corrispettivo database

Terminata la presentazione sull'interazione con l'ambiente di lavoro e la creazione di nuove manovre, occorre soffermarsi su quello che è stato l'effettivo utilizzo di quelle vecchie. All'inizio di questa trattazione, infatti, l'uso delle stesse non era stato previsto se non per scopi conoscitivi e per riuscire a muoversi meglio all'interno dell'ambiente di lavoro. Dopo alcune prove sulla seconda parte del lavoro di Tesi,

però, per ampliare ancora di più lo stesso e, contemporaneamente, la sua validità, si è scelto di usufruire delle stesse, in un modo che sarà più chiaro nel capitolo della trattazione dedicato all'IA. Conseguentemente, si è intervenuti su ognuna delle manovre a disposizione nel *Maneuver* originale (principalmente a livello di fogli Excel, ma anche sul corrispettivo codice), nell'ottica di renderle compatibili con il mezzo scelto per lo studio del problema del ribaltamento dei veicoli pesanti. Questa operazione è stata necessari in quanto, la maggior parte delle stesse, per quanto numerose e messe automaticamente a disposizione, risultavano avere gli stessi parametri caratterizzanti, indipendentemente dalla tipologia di veicolo (come visto in Figura 23, stessa manovra porta ad esiti molto diversi), portando spesso e volentieri ad un rollover non desiderato del truck. Si evidenzia, peraltro, che alcune manovre sono state fornite tramite file CRG (Curved Regular Grid, un sistema per salvare quella che è, a tutti gli effetti, una scansione 3D della strada, che può essere fatta anche con un'analisi GPS), senza però che nel codice adibito alla trattazione delle stesse fosse presente un elemento in grado di aprirli. Nell'ottica di ovviare al problema, si è quindi scaricato il file *ASAM OpenCRG®*, per poi effettuarne l'upload (per quanto riguardava la parte compatibile con Matlab) nel percorso, riferendoci a Figura 30, *Libraries\Event\Tools \CRG_Tools*, permettendo così di interagire con le manovre in questione, che potevano a questo punto essere generate in formato STL tramite il codice *sm_car_crg_to_stl*. Tuttavia, non è stato comunque possibile sfruttare appieno il potenziale dei tracciati contenuti nei file in questione, dato che la totalità degli stessi, per poter avere un'interazione corretta, richiedeva che il veicolo fosse dotato di pneumatici *MF-Swift* o *Delft* (non solo a livello di codice, dove gli stessi erano previsti, ma a livelo di modello fisico, file STL), non reperibili online se non dal costruttore stesso (che non rispondeva da mesi). Si è provato ad intervenire a livello di codice (*sm_car_config_road*, nella stessa cartella del codice del *Maneuver*) per forzare l'utilizzo di pnemautici classici basati sulla formula di Pacejka (*MF-eval*) ma, sebbene si sia riusciti nell'intento, l'esito è stato molto poco

soddisfacente in termini di percorrenza dei tracciati in questione (tra cui numerosi percorsi di Formula 1, come Suzuka o il Nurburgring Nordschleife), costringendo a ripiegare su una versione semplificata (sempre fornita da Steve Miller nel programma originale) dei medesimi. Si fa notare, peraltro, che anche le superfici (definite nello specifico, una ad una, nel percorso *Libraries\Event\ \Scene*), richiamate in formato dal codice del *Maneuver* (*sm_car_config_maneuver)* sia in termini strutturali (all'interno del database *Scene*, non raggiungibile da UI) che in termini di valori pratici (sempre *sm_car_config_road*) risultavano essere pensate principalmente per veicoli a due assali, richiedendo quindi un intervento, che chiaramente è stato fatto, per aggiornare e riadattare le stesse al veicolo in questione.
Una volta ultimata questa procedura è stato poi possibile passare alla vera e propria parte di Simulazione e creazione dei dataset utilizzati nella seconda parte della trattazione.

Criteri per identificazione rischio di rollover e dataset

In questa sezione, alla luce di quanto presentato per *Post Processor* e manovre (termine con cui si intenderà, da ora in poi, l'insieme di tutti gli elementi caratterizzanti il codice *sm_car_config_maneuver* descritto in precedenza), viene trattata un'altra parte fondamentale della trattazione, ovvero la scelta dei criteri utilizzati per identificare il rischio di rollover e la generazione dei dataset di variabili ottenute tramite il *Post Processor* e un ulteriore codice (di cui si scriverà poco dopo).
Riferendoci al primo aspetto dei due sopra citati, uno degli indici di ribaltamento del truck considerato è chiaramente l'angolo di **rollio** stesso, che si è visto essere critico una volta raggiunti gli 8°/10°, dopo i quali il mezzo inizia la fase di rollover, raggiungendo irrealistici (e conseguentemente non considerati) valori di 200°, ottenuti solamente per il fatto che la superficie implementata su Simscape non impedisce una compenetrazione con il veicolo, ormai in rotazione intorno al suo asse longitudinale (asse x di Figura 13, movimento riscontrabile selezionando la

manovra *Rollover* nel *New Maneuver* della UI). L'angolo in questione è fortemente legato al tipo di manovra assegnata al veicolo (cui viene richiesto, ad esempio, di affrontare una curva su un percorso ghiacciato) e, conseguentemente, ad elementi quali l'accelerazione (nelle tre direzioni) del corpo in questione, considerata nel baricentro CG (a_{CG}, di cui la componente laterale risulta essere quella più critica in caso di ribaltamento), la sua velocità longitudinale ($v_{x_{CG}}$), sempre in CG, quella di imbardata ($\dot{\psi}$) e l'angolo di sterzo del volante (δ_{SA}, angolo volante viene passato all'assale anteriore senza che sia stato introdotto un rapporto di sterzo). Ciascuno di questi elementi è facilmente ottenibile tramite i sensori normalmente presenti a bordo di un qualunque veicolo (ad es. l'ECU, Electronic Control Unit). Per poter effettuare una stima e una previsione del rischio di rollover, però, riuscire ad ottenere, dai sei parametri legati alla manovra (e alle caratteristiche del mezzo), tramite reti neurali, il solo rollio, non è sufficiente. In quest'ottica occorre quindi introdurre altri indici rilevanti, i **Load Transfer Ratio** (LTR), ovvero i parametri di trasferimento di carico verticale (F_z, osservato su ciascuna ruota) per ognuno degli assali coinvolti (sei pneumatici, tre assali in tutto). Questi, così come il rollio, non sono facilmente ottenibili da sensori a bordo, rendendo necessario ricorrere a metodi alternativi per poterli calcolare ((Xinjie Zhang and Peng, 2017)).
In questa trattazione, rollio ed LTR sono direttamente ricavati, estraendo gli opportuni parametri, tramite il *Post Processor*. Questi risulteranno essere i termini di confronto, considerati **reali** (le simulazioni hanno proprio, come detto, funzione sostitutiva di dati normalmente ricavabili da effettive situazioni a bordo veicolo), con rollio ed LTR derivanti invece dalle reti neurali che verrano presentate in seguito, rappresentanti i valori *simulati* di questi indici. Si anticipa già ora che i network utilizzati si basano sui sei parametri legati alla manovra citati in precedenza, ovvero a_{CG} (cui corrispondono $a_{x_{CG}}$ $a_{y_{CG}}$ e $a_{z_{CG}}$), $v_{x_{CG}}$, $\dot{\psi}$ e δ_{SA}. Questi elementi rappresenteranno gli *input* delle reti, mentre il rollio e i tre LTR gli *output* in uscita dalle stesse. Si porta poi all'attenzione il fatto che, in fase preliminare del lavoro di Tesi,

sono state studiate anche reti basate su 21 input in ingresso (contenenti elementi ricavabili tramite un IMU su veicolo e circuiti di prova), che verranno anch'esse brevemente presentate.

Come riportato sopra, aspetti particolarmente importanti della trattazione sono i sei input e i 4 output che andranno a definire l'attività principale a livello di IA. Si presentano, di conseguenza, i risultati per la *Manovra_articolata* del *New Maneuver* per gli stessi, in modo da poterne visualizzare meglio le proprietà. Viene evidenziato che la manovra adottata a titolo esemplificativo è caratterizzata dal muoversi su una superficie piana (parametri di attrito dinamico delle ruote, μ, tutti pari ad 1) e da una doppia sequenza di step steer (manovra a gradino) e ramp steer (manovra a chiocciola), con differenti posizioni dei pedali di acceleratore e freno, ispirata ad un percorso sulla collina torinese. Si fa notare, inoltre, che è stata pensata per una logica in *Open Loop*.

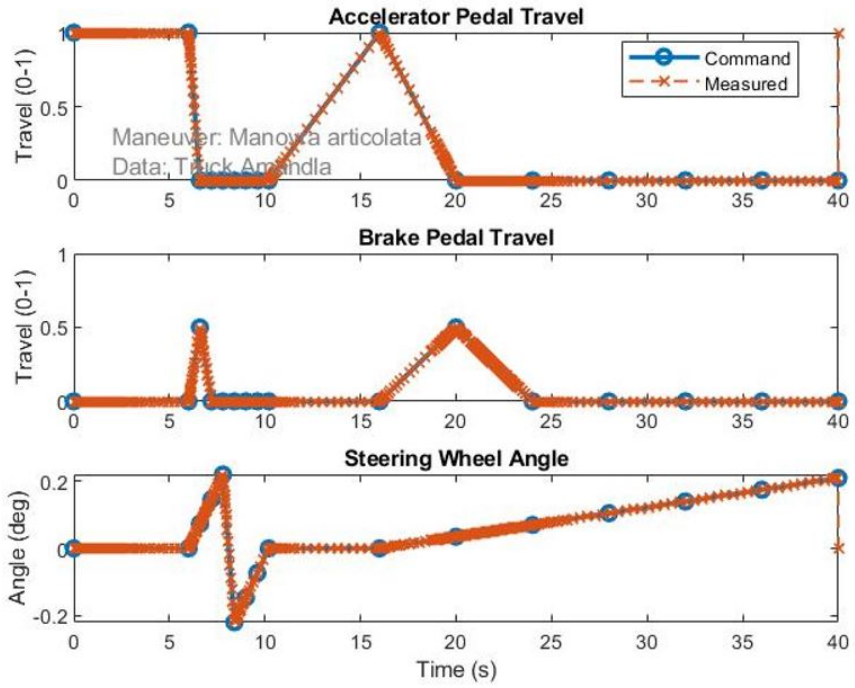


Figura 33: Caratteristiche della manovra

Selezionata la manovra di cui sopra (le cui particolarità sono visibili in Figura 33, ottenibili anche tramite lo script di post-simulazione, ma qui riportate attraverso uno dei plot di default del programma), è importante sottolineare che, per quanto concerne la posizione dei pedali di acceleratore e freno, questi risultano essere in uso contemporaneamente nel tratto che va dai 15 ai 20 secondi. Questo è stato, semplicemente, fatto in modo da ridurre la velocità del veicolo (che altrimenti si sarebbe ribaltato) e simulare così un comune controllo elettronico.
Per osservare i risultati della manovra in questione, è stato poi avviato il modello (Figura 29) composto per lo studio di interesse. Al termine della procedura di simulazione, con la generazione del file *logsout* nel Workspace, è stato poi possibile, secondo una logica già anticipata in precedenza, sfruttare il codice di *Post Processor* per ottenere gli esiti cercati, che vengono riportati immediatamente di seguito, commentati nei loro aspetti principali, partendo dalle grandezze che saranno l'input delle reti neurali.

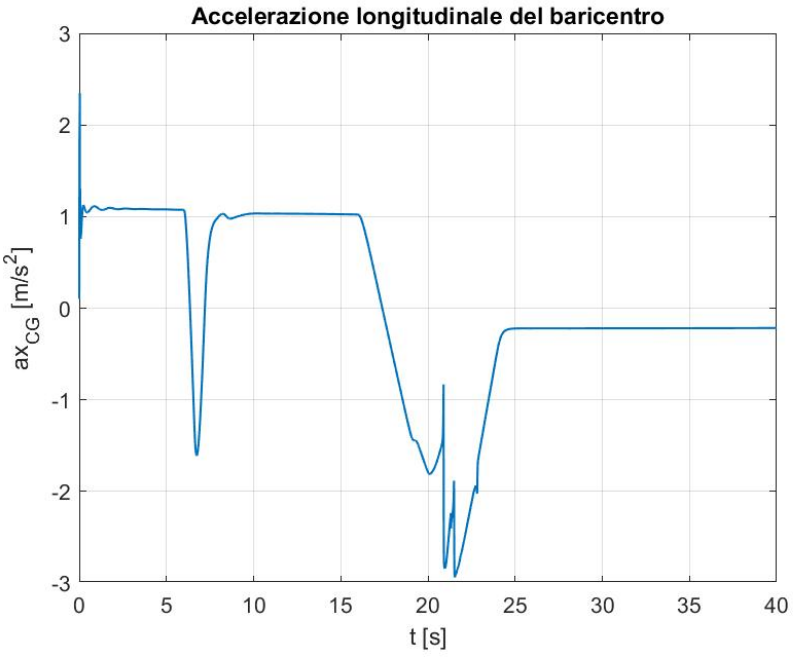


Figura 34: Accelerazione longitudinale del baricentro $a_{x_{CG}}$ con manovra personalizzata

Per quanto concerne l'andamento dell'accelerazione $a_{x_{CG}}$, esso riflette appieno quanto realizzato nella manovra in termini di pedali acceleratore e freno. Il primo tratto, infatti, è caratterizzato da un'accelerazione pressoché costante (al di là dell'iniziale e molto breve momento di assestamento), cui fa seguito una forte decelerazione legata ad un andamento molto simile a un gradino dell'acceleratore da una posizione 1 ad una posizione 0 (pedale completamente libero) e al contemporeneo intervento del pedale del freno (che passa, nell'arco di un secondo, da una posizione 0 ad una 0,5, per poi tornare in posizione zero, in cui il freno non viene più premuto). L'azione del freno è comunque troppo repentina per arrestare il moto completamente, che riprende con un nuovo intervento (questa volta più simile a una rampa) del pedale acceleratore, nuovamente riportato a un valore obiettivo 1. Con ogni probabilità, non si assiste a delle variazioni maggiori nel diagramma in questo tratto perché il veicolo è stato sottoposto per un tempo molto ridotto a frenata (ma quel tanto che bastava per permettere l'esecuzione della manovra di brusca sterzata, come osservabile nella terza immagine in Figura 33). In tal senso, gli effetti minori si osservano nel tratto che va dai 15 ai 20 secondi, dove non solo si inizia a "premere" sempre meno il pedale dell'acceleratore, ma si aggiunge anche l'effetto del pedale del freno (in ottica di simulazione di un sistema di controllo elettronico). Si ottiene così una decisa decelerazione che, una volta assestata, permane in maniera minore fino all'arresto del veicolo.

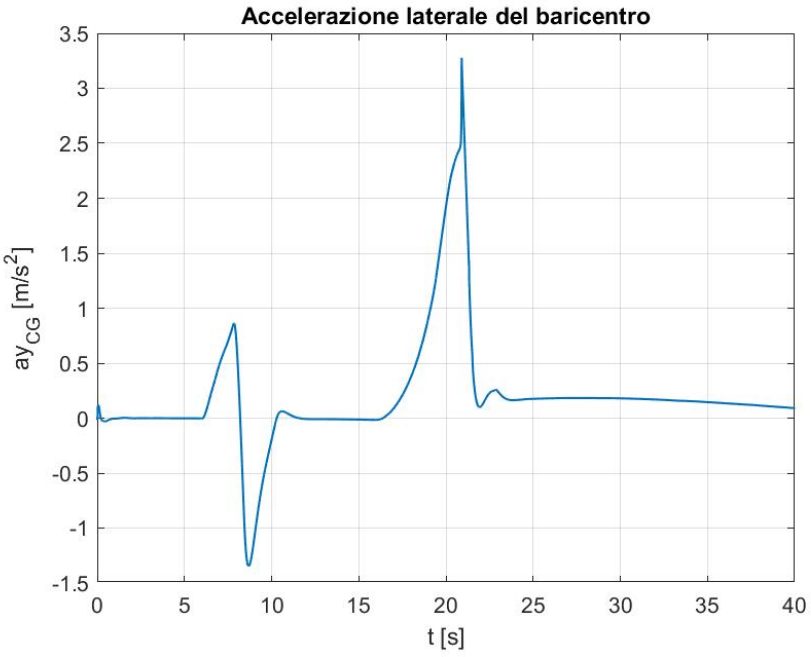


Figura 35: Accelerazione laterale $a_{y_{CG}}$ del baricentro con manovra personalizzata

Come anticipato, l'accelerazione $a_{y_{CG}}$ risulta essere quella principale tra le tre componenti di a_{CG} per questa trattazione, in quanto quella maggiormente legata agli effetti di ribaltamento del veicolo (per via delle azioni di inerzia che porta con sé). Per limitare la stessa, che come si può osservare in Figura 35 presenta comunque un picco in corrispondenza dei 20 secondi (legato al fatto di avere prima accelerato sostenutamente, contemporaneamente all'esecuzione della parte ramp steer della manovra), si è utilizzato il pedale del freno, senza il quale si sarebbe andati molto oltre i 3-3,5 m/s^2 osservabili nel massimo in figura, dato che avremmo osservato (per verifiche sperimentali fatte senza utilizzo dei freni) una condizione di ribaltamento del mezzo. Per ovviare alla stessa, chiaramente, si poteva decidere di cambiare semplicemente la manovra, ma si è ritenuto che la stessa, strutturata in questo modo, rappresentasse un esempio significativo del potenziale del programma.

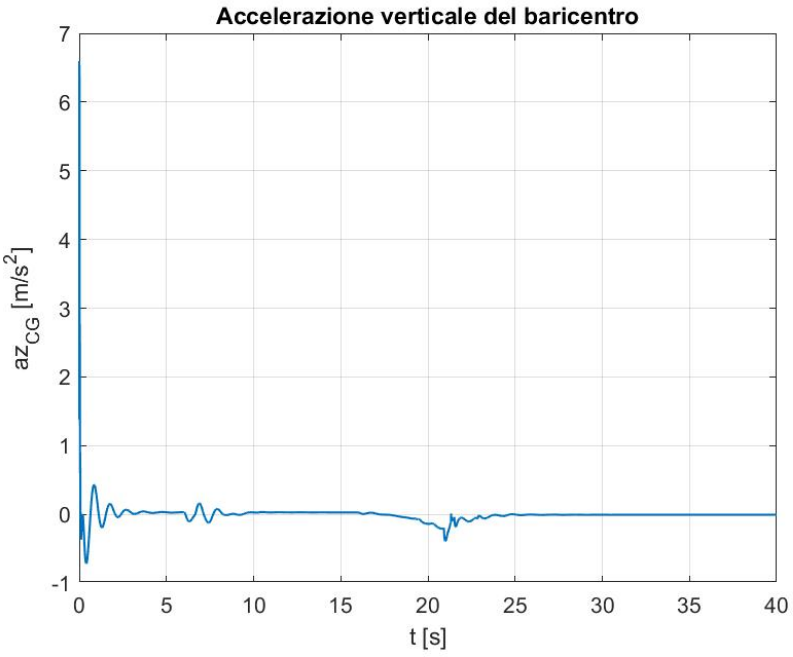


Figura 36: Accelerazione verticale $a_{z_{CG}}$ del baricentro con manovra personalizzata

Per quanto concerne la componente verticale di a_{CG}, essa risulta correttamente pari a circa zero, dato che si è scelto di eseguire la manovra su una superficie piana (dal punto di vista dell'andamento, $a_{z_{CG}}$ è infatti l'accelerazione che offre meno spunti di analisi in questo caso). In un'ottica comparativa, in Figura 37 viene proposto un confronto tra gli andamenti (e i conseguenti valori) delle tre accelerazioni sopra esaminate.

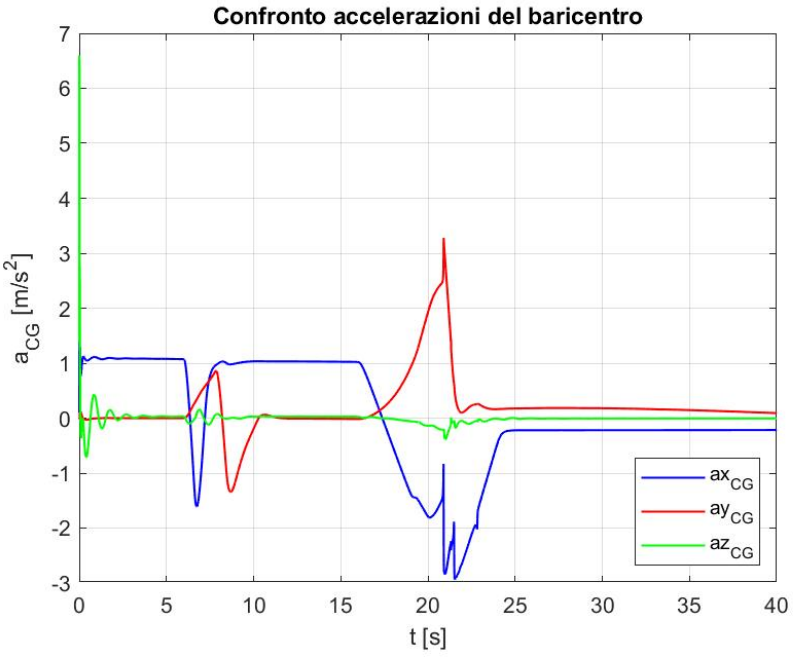


Figura 37: Confronto accelerazioni del baricentro in caso di manovra personalizzata

Altro elemento rilevante e facilmente rilevabile a bordo veicolo è poi la velocità longitudinale dello stesso, il cui andamento viene riportato in Figura 38. Anche questo altro non è che una diretta conseguenza delle azioni impartite tramite i pedali di acceleratore e freno, che a loro volta tengono conto dell'angolo di sterzo fornito al volante (si ricorda ancora una volta che, per come è strutturato il modello, questo coincide con l'angolo di sterzo delle ruote, non essendo presente rapporto di sterzo).

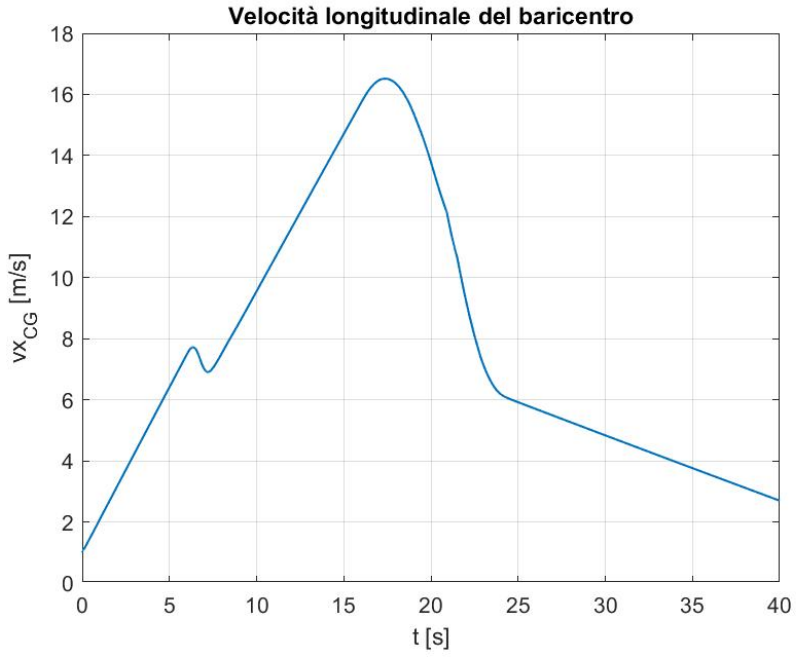


Figura 38: Velocità longitudinale del baricentro $v_{x_{CG}}$ con manovra personalizzata

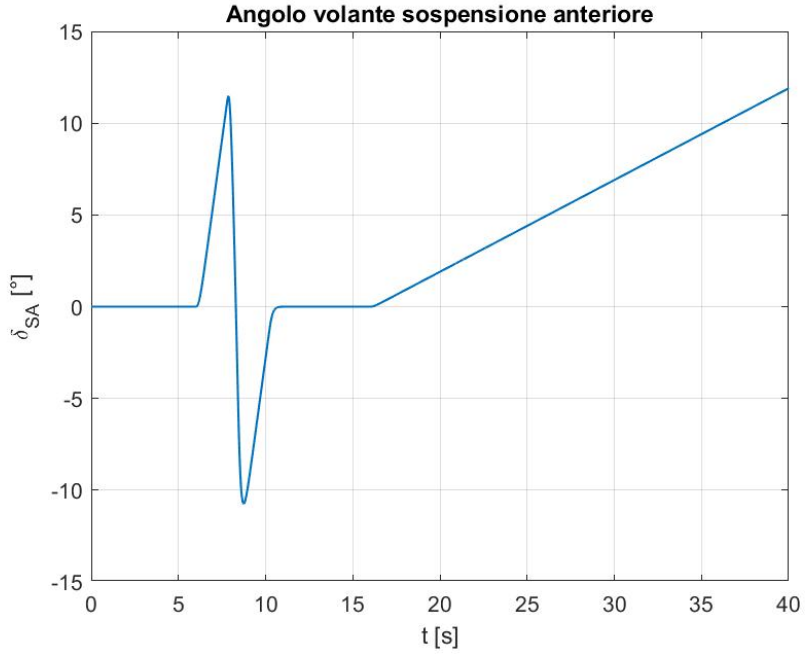


Figura 39: Angolo di sterzo al volante δ_{SA} con manovra personalizzata

Si presenta infine, per quanto concerne gli input, il diagramma legato alla velocità angolare di imbardata. Questo risulta essere, come lecito aspettarsi, molto simile a quello ottenuto per l'accelerazione laterale.

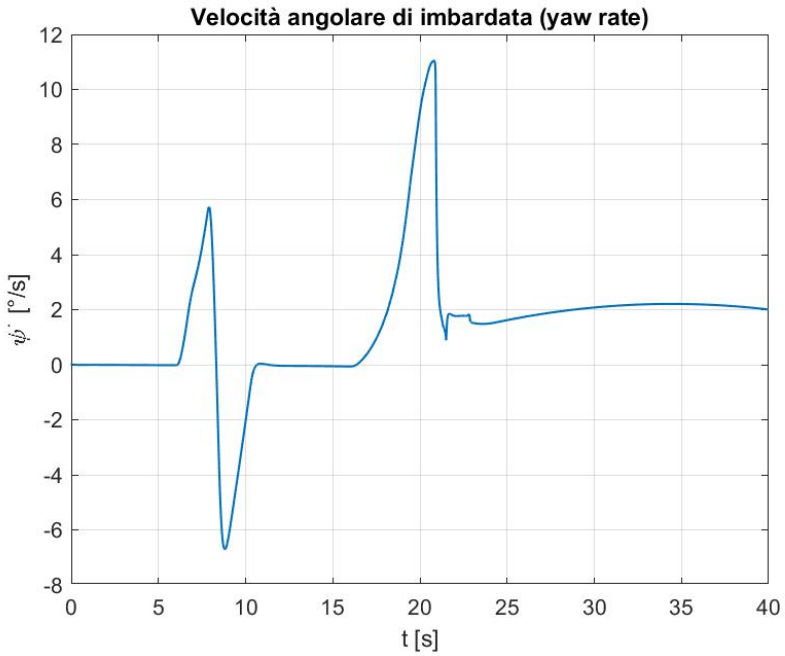


Figura 40: Velocità di imbardata $\dot{\psi}$ con manovra personalizzata

Passando agli elementi che saranno veri e propri indici di incipienti condizioni di rollover, nonché obiettivo di espressione finale da parte delle reti neurali, si presentano ora i risultati ottenuti per rollio ed LTR dei tre assali.
Partendo dal primo, come osservabile in Figura 41, presenta un andamento, come lecito attendersi, molto simile a quello osservato in Figura 35.

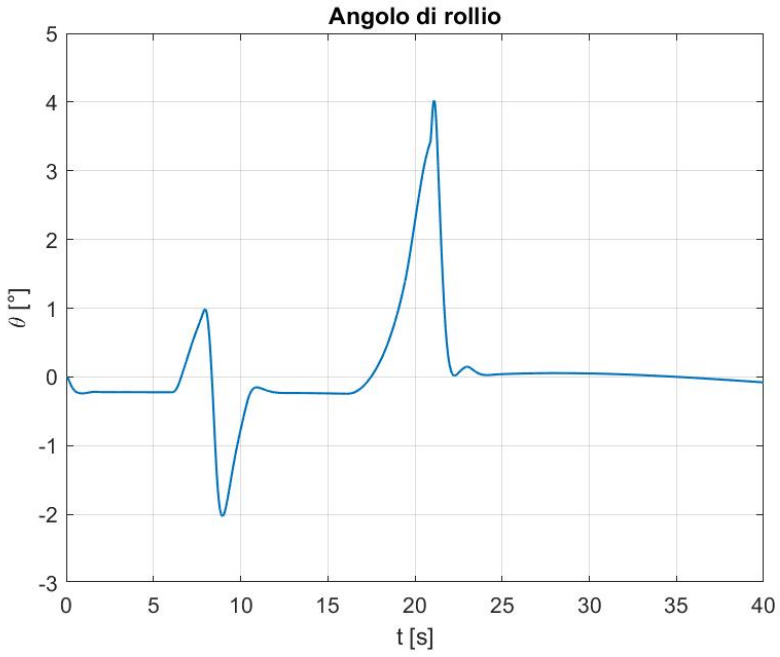


Figura 41: Angolo di rollio θ con manovra personalizzata

Per quanto concerne i trasferimenti di carico, questi risultano essere molto variegati tra loro, ma con andamenti assolutamente coerenti con la tipologia di mezzo adottata e composta.

Riferendoci a Figura 42, possiamo notare come il trasferimento di carico verticale, riferito all'assale anteriore, risulti essere particolarmente ridotto. Questo è un andamento assolutamente corretto se teniamo conto del fatto che il truck sia, come norma, a trazione posteriore (motore sull'assale centrale in questo caso) e che quindi osservi degli effetti inerziali maggiori nella zona dove è collocato il motore (si ha una distribuzione diversa delle masse), come mostrato nelle figure 43 e 44. Si fa notare che il parametro LTR (Load Transfer Ratio) non presenta unità di misura, essendo definito attraverso il rapporto:

$$LTR = \Delta Fz/Fz \tag{3}$$

Dove ΔFz rappresenta, per ciascun assale, la differenza tra carico verticale cui è sottoposta la ruota di destra rispetto a quella di sinistra, mentre Fz la semplice

somma aritmetica delle due componenti in questione. Questo porta ad avere una grandezza complessiva di natura adimensionale e variabile all'interno di un intervallo che va da -1 (carico interamente sorretto dalla ruota a sinistra) e +1, limiti in corrispondenza dei quali si verifica il sollevamento di una delle due ruote (-1 sollevamento ruota di destra, +1 sollevamento ruota di sinistra). Si è in generale verificato, con tutte le manovre utilizzate, che queste condizioni estreme possono essere mantenute solo per intervalli di tempo ridotti (pochi secondi), al termine dei quali si verifica il ribaltamento del mezzo. Conseguentemente, è possibile assumere che se si arrivi ad avere dei valori di LTR pari ad 1 o -1, il veicolo, effettivamente, in un'ottica conservativa, stia per incorrere in rollover (secondo quanto visto anche in (Tianjun Zhu and Ma, 2020)), validando l'ipotesi di utilizzo degli LTR come indici di rischio del ribaltamento del mezzo. Questi risultano essere ancora più efficaci della conoscenza dell'angolo di rollio, dato che leggendo gli andamenti dei trasferimenti di carico verticale, è possibile individuare con esattezza dove si stia per verificare una condizione di rollover, mentre nel caso dell'angolo θ occorre essere in grado di interpretare effettivamente quanto espresso dal diagramma.

Esaminando nel dettaglio i diagrammi ottenuti tramite la manovra di prova, possiamo notare, come anticipato, dei valori molto ridotti per quanto concerne il trasferimento di carico associato all'assale anteriore (A1).

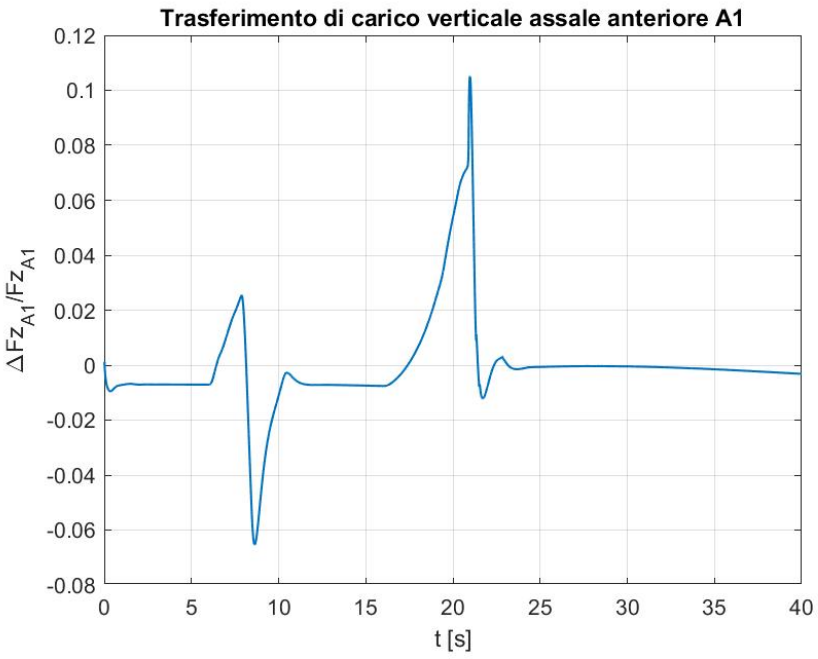


Figura 42: LTR assale anteriore $\dot{\psi}$ con manovra personalizzata

Andamenti simili, ma con valori estremamente più elevati (circa un ordine di grandezze) vengono poi riscontrati per quanto concerne i trasferimenti di carico legati all'assale centrale (A2) e a quello posteriore (A3), dove si ha un effettivo raggiungimento della condizione limite 1. Da qui è possibile estrapolare, come detto, numerosi spunti legati alla manovra. Sapendo infatti che LTR ha raggiunto il suo valore massimo positivo, automaticamente è chiaro che, in fase di esecuzione del movimento, in quel periodo di tempo la ruota sinistra si è sollevata, suggerendo quindi che si stesse effettuando una curva nella medesima direzione. Da questo diagramma possiamo anche capire ancor meglio il perché della necessità dell'intervento del pedale del freno (Figura 33), senza cui avremmo osservato un completo ribaltamento del veicolo (come si può notare, anche il trasferimento di carico legato all'assale centrale aveva ormai iniziato a toccare lo stesso limite, con sollevamento quindi di due delle tre ruote del lato sinistro del veicolo).

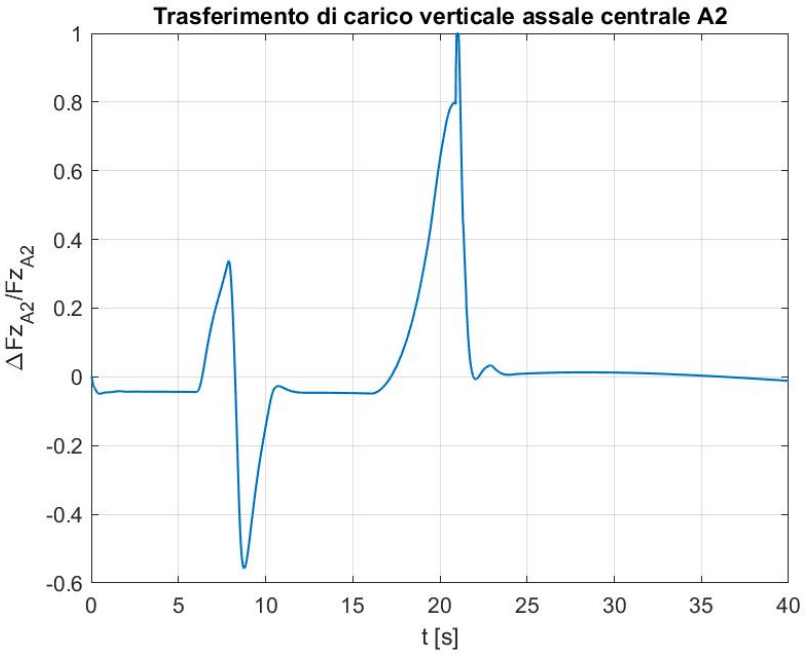


Figura 43: LTR assale centrale $\psi^{\cdot}$ con manovra personalizzata

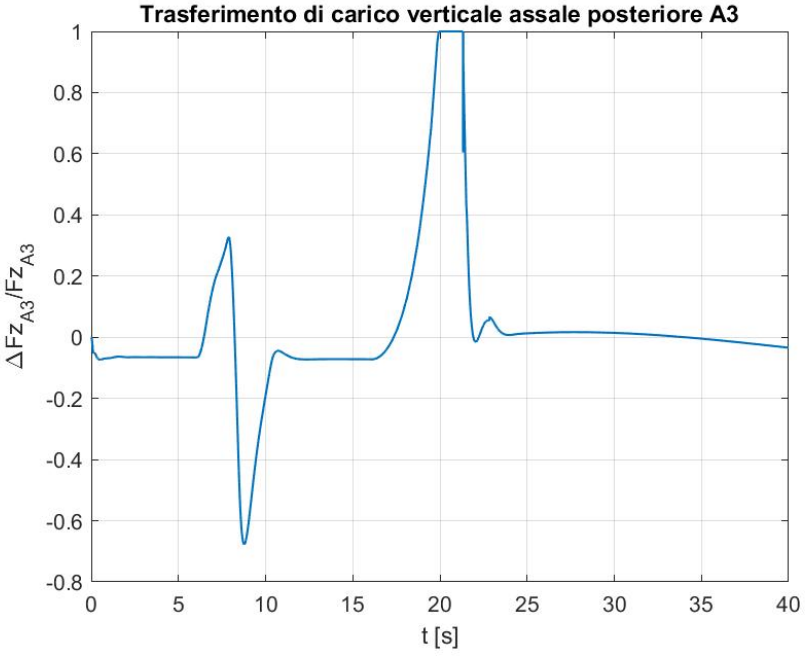


Figura 44: LTR assale posteriore $\psi^{\cdot}$ con manovra personalizzata

Terminata la parte di presentazione delle variabili di interesse (con relativo esempio specifico) e dei criteri di identificazione dei rischi di ribaltamento, risulta di fondamentale importanza evidenziare come avvenga il salvataggio, per ognuna delle manovre coinvolte nel processo (23 in tutto), dei parametri di cui sopra. Si sottilinea, infatti, la necessità di predisporre un collegamento tra i dati generati grazie al *Post Processor* e l'IA, ovvero le reti neurali. Per fare ciò, al termine di ogni simulazione per ciascuna manovra, è stato salvato un *dataset* attraverso il codice *Post_processor_TA_save_maneuver*, nello stesso percorso di *sm_car_config_New_maneuver* e accessibile anche a livello di UI, come osservabile in Figura 45.

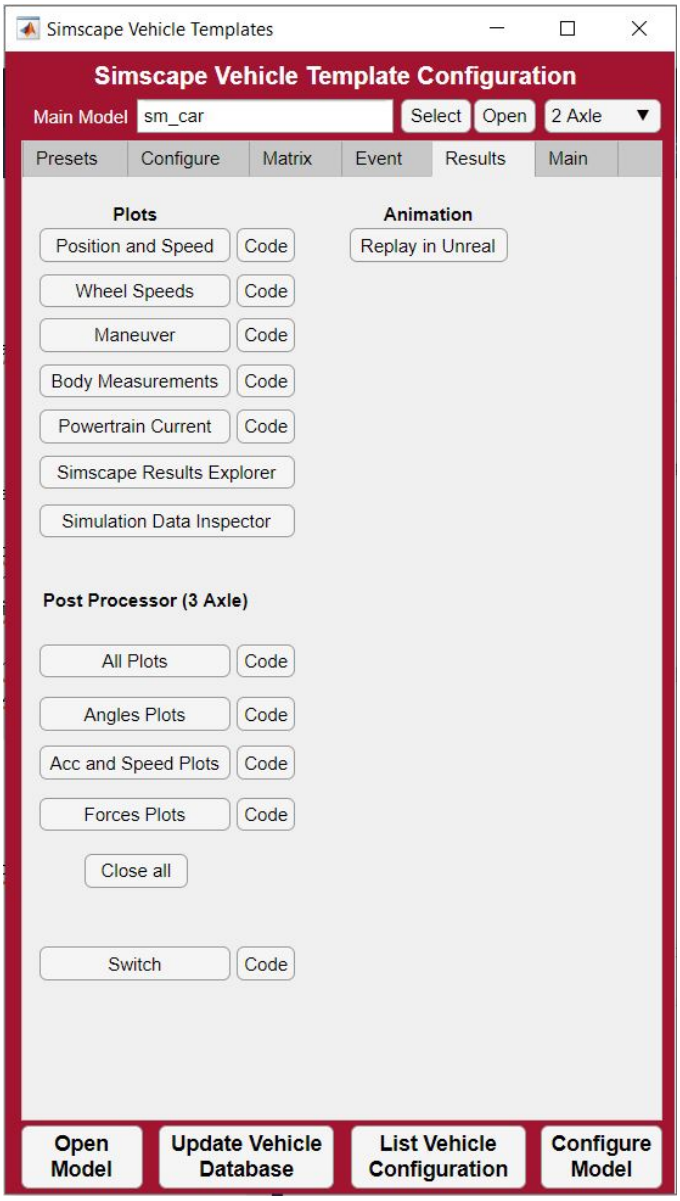


Figura 45: UI modificata per permettere salvataggio dataset manovre

Il codice immplementato, funzionante tramite stringhe e un comando di switch, permette, una volta selezionata la manovra, effettuata la stessa e avviato il *Post Processor* di salvare in una cartella appositamente dedicata i dataset (sotto forma di un foglio Excel e una tabella Matlab per singola prova) contenenti le informazioni di ogni variabile rilevante (si è deciso di mettere tutte le 23, più il tempo, ricavabili

da una IMU, anche se delle stesse se ne sfrutteranno solo 10). Grazie all'aver introdotto uno script basato su una funzione di swtich, cambiando manovra è sufficiente premere sul pulsante in questione perché il dataset della stessa venga automaticamente salvato nella cartella corretta. Da questa, i codici elaborati per le reti neurali attingeranno in modo da caricare i dataset ed elaborarne le informazioni nell'ottica di ottenere degli indici di rischio di rollio *simulati* prossimi a quelli *reali* (contenuti nei dataset e derivanti direttamente da Simscape).

Considerazioni e possibili sviluppi futuri: Simscape

In questa prima parte della trattazione è stata fornita una panoramica generale del programma Simscape e, nello specifico, della variante *Simscape Vehicle Templates*. Tramite questa e l'implementazione e modifica di numerosi codici, è stato possibile ottenere delle simulazioni realistiche con dati rilevanti dal punto di vista dello studio di rollover di mezzi pesanti, per poi salvare gli stessi e utilizzarli per addestrare e testare le retu neurali, costituenti la parte di IA di questo lavoro di Tesi.
In ottica di eventuali sviluppi futuri, nel caso in cui fosse possibile contattare un produttore, sarebbe opportuno implementare nel programma i modelli STL degli pneumatici *MF-Swift* e *Delft*, in modo da utilizzare meglio (ovvero sfruttandone la superficie più verosimile e non una sua versione semplificata) le manovre CRG. In questo senso, potrebbe essere molto interessante provare a introdurre delle superfici ricavate tramite GPS, per poi elaborle in formato CRG e vedere come possa funzionare l'implementazione sul programma (questo eventualmente dopo essersi procurati le tipologie di pneumatici di cui sopra). Sempre a livello di miglioramento della simulazione e dell'ambiente riprodotto dalla stessa, sfruttare l'*Unreal Engine* in maniera più completa, e non solo nell'ottica di riproduzione grafica, potrebbe permettere grandi sviluppi, anche solo per tutti gli studi già sviluppati in quell'ambiente (sarebbe però necessario imparare un'altra logica di programmazione, per quanto non sia sembrata così complessa). Si evidenzia, infine, la possibilità, pressoché illimitata, di introdurre quante nuove manovre e superfici si vogliano. Per queste ultime occorre però fare attenzione, in quanto è possibile incorrere in problemi di interpretazione del software a livello pratico e di compenetrazione del veicolo con le stesse da un punto di vista visivo. Le opportunità di sviluppo, però, sono, come detto, virtualmente infinite, anche solo se si decidesse di intervenire, oltre che sulla UI e sui codici, anche sui blocchi fisici del programma, introducendo ad

esempio nuove librerie o modificandone i componenti (per alcuni questa operazione non risulta possibile, ma implementare blocchi ex novo e favorirne l'interazione con quelli preesistenti è una possibilità, per quanto questi ultimi siano già piuttosto vari, completi e complessi).

IA: analisi dati

Prefazione

Dopo aver presentato come ottenere ed elaborare i dati in ambiente Matlab grazie al programma Simscape, è possibile ora passare all'analisi degli stessi nell'ottica di ricavare, tramite l'utilizzo di reti neurali, delle stime e delle previsioni degli indici di ribaltamento, a partire da dei parametri di input (come anticipato al termine dello scorso capitolo). In questa parte della trattazione verrà quindi mostrato quanto fatto in termini di addestramento e verifica dell'efficacia delle reti (in particolare per quattro manovre, ciascuna con scopo di evidenziare aspetti diversi), presentando poco prima il perché si sia scelto di usarle e, nel mentre, le caratteristiche dell'ambiente di lavoro e l'architettura delle stesse. Si fa notare, peraltro, che anche in questa parte della Tesi le implementazioni effettuate sono poi state introdotte a livello di UI. Una volta terminata la descrizione di quanto fatto, vengono poi presentate nuovamente delle considerazioni e dei suggerimenti in ottica di eventuale prosecuzione ed ampliamento del lavoro.

Reti neurali

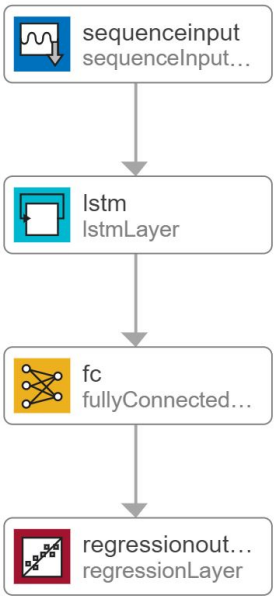
Presentazione

Effettuata la generazione e selezione dei dati di interesse nella prima parte della trattazione, è stato necessario capire come poter analizzare gli stessi in modo da ottenere delle grandezze indici di rollover, di cui si avevano i valori *reali* (ovvero provenienti dal *Post Processor*), in un'ottica di logico confronto e comparazione per valutazione della bontà del sistema di IA predisposto. In questo senso, dopo aver provato un software di data mining, *Rapid Miner*, e aver realizzato tramite il medesimo diversi modelli, ci si è resi conto che, oltre a non essere completamente chiaro il processo costruito (non essendo possibile accedere direttamente al codice), i risultati erano molto insoddisfacenti, oltre ad avere una scarsa capacità di intervento e una libertà di implementazione limitata. Conseguentemente, dopo essersi informati in letteratura ((Xuanwei Chen and Zhu, 2020)) e anche nell'ottica di rendere il più possibile user friendly la procedura di analisi dei dati contenuti nei dataset predisposti grazie al codice *Post_processor_TA_save_maneuver*, si è deciso di tornare in ambiente Matlab. All'interno dello stesso ci si è poi avvalsi del *Deep Learning Toolbox*, spostandosi dallo studio di generici modelli statistici a quello più specifico delle *deep neural networks*, ovvero reti neurali caratterizzate da almeno quattro livelli (*layers*) di profondità (da 1 a 3 si parla di *shallow neural networks*, reti neurali superficiali, utilizzabili, ma meno adatte per i nostri scopi). Dopo numerose prove, dove ci si è avvalsi anche delle diverse app correlate al *Deep Learning Toolbox*, è stato quindi possibile ottenere i risultati cercati sotto diversi punti di vista. Il processo seguito può essere ora riassunto in due momenti principali:

- *Recupero dataset e addestramento*
- *Analisi e risultati fase di testing*

Recupero dataset e addestramento

Ultimata la raccolta dei dati necessari nella prima fase (Simscape), si passa a recuperare gli stessi per avviare la procedura di analisi e di ottenimento delle grandezze cercate (indici di ribaltamento). Questo viene fatto, molto semplicemente, con una stringa di codice all'interno dello script generale pensato per l'addestramento, ovvero *Ultimo_training_2*, osservabile nella cartella *Networks*, all'interno della quale sono inseriti tutti i codici principali finali e le librerie (chiaramente dentro altre cartelle) dei dataset e degli script creati per ottenere i codici di cui sopra.
Risulta quindi particolarmente rilevante concentrarci su come sia stato implementato il codice *Ultimo_training_2*. Per fare ciò, è quindi bene fornire una panoramica generale di come sia stato strutturato lo stesso.
Obiettivo principale era quello di ottenere una rete neurale addestrata su manovre predefinite (conoscenza sia degli *input* che degli *output*), per poi andarla ad utilizzare per manovre i cui *output* (ovvero le grandezze indici di rollover) risultassero ignoti, da ricavare. In quest'ottica, è stata quindi definita innanzitutto un'architettura del network, per poi utilizzarla in un processo di continuo addestramento e riaddestramento con il fine di ottenere la rete desiderata. Si evidenzia che si è scelta, come struttura della rete, una composizione a quattro layers, come osservabile in Figura 46, dove viene riportato anche il codice della stessa.



(a) Architettura prima rete neurale, *net_1*

```
%% Definizione dell'architettura della rete neurale-1
% NOTA
% Le risposte YTrain sono state organizzate in modo da essere riferite alle colonne
numFeatures = length(XTrain_1(:,1));
numResponses = length(YTrain_1(:,1));
numHiddenUnits = 100; % aumentando il numero di hidden units aumenta logicamente il
% ma anche la pendenza della curva di regressione (più rapida). Con 100 si
% hanno comunque buoni risultati

layers_1 = [ ...
    sequenceInputLayer(numFeatures)
    lstmLayer(numHiddenUnits)
    fullyConnectedLayer(numResponses)
    regressionLayer];

options_1 = trainingOptions('adam', ...
    'MaxEpochs',200, ... % ogni epoca corrisponde a un passaggio completo dei dati,
    'GradientThreshold',1, ...
    'InitialLearnRate',0.01, ...
    'LearnRateSchedule','piecewise', ...
    'LearnRateDropPeriod',120, ...
    'LearnRateDropFactor',0.2, ...
    'Verbose',0, ... % settare a 1 (true) per vedere andamento parziale nella comma
    'VerboseFrequency',50,...
    'Shuffle','never',...
    'ResetInputNormalization',0,...
    'DispatchInBackground',1,...
    'Plots','training-progress');
```

(b) Codice prima rete neurale, *net_1*

Figura 46: Presentazione caratteristiche della prima rete neurale utilizzata in fase di addestramento

Riferendoci all'immagine di cui sopra, come anticipato, possiamo distinguere quattro *Layers* (livelli) pricipali. Il primo è associato a un blocco di ingresso delle variabili all'interno della struttura della rete. Particolarmente importante risulta poi essere il secondo, ovvero il layer *lstm* (Long Short Term Memory). All'interno dello stesso (molto usato per variabili che si sviluppano nel tempo, come nel nostro caso), infatti, è dove avviene il vero e proprio processo di apprendimento del network tramite il passaggio delle informazioni in un numero ben precisato di unità nascoste, i neuroni della rete neurale. Entrando più nello specifico, attraverso l'interazione tra *input* e neuroni, la rete elabora, impostando relazioni matematiche e un peso per ciascuna variabile, un vero e proprio modello, volto a ricavare il legame tra input ed output (che verranno indicati come valori a cui tendere tramite la funzione specifica *trainNetwork*). Particolarmente rilevante diventa quindi il fatto che questo modello possa essere aggiornato a seconda dei casi, inviando ad esempio (come è poi stato

fatto) nuovi input a una rete già definita. Per quanto concerne gli ultimi due livelli, è bene fare notare che il *fullyConnected layer* permette di tenere conto del peso di ciascuna variabile in ingresso, mentre l'ultimo livello ha funzione di restituire la radice dell'errore quadratico medio per valutare l'andamento della funzione di training (la già citata *trainNetwork*).

Assume notevole rilevanza specificare che, oltre all'architettura in sé, un ruolo centrale per il corretto funzionamento di una qualunque rete neurale è dato dagli input e dagli output che vengono richiesti alla stessa, senza dimenticare la definizione delle opzioni di addestramento (funzione *trainingOptions*). Infatti, sviluppato un determinato network, questo potrà lavorare esclusivamente con il numero di *input* e di *output* per cui è stato generato, dando altrimenti errore una volta passati alla funzione di addestramento vera e propria. Da questo si intuisce come sia opportuno, prima di avviare una fase di training multipla (caratterizzata da una rete continuamente aggiornata), decidere con chiarezza quali debbano essere gli *input* (e rispettivo numero) e quali gli *output* (e rispettivo numero). Particolarmente rilevante è poi sottolineare che i valori in ingresso ed in uscita devono essere, soprattutto nel nostro caso (dove si usano, solo in *input*, accelerazioni, angoli e velocità), standardizzati, ovvero resi adimensionali, almeno per la fase di addestramento della rete (buona parte del codice *Ultimo_training_2* è proprio adibita a questo scopo, in modo da rendere adatti i valori contenuti nelle tabelle Matlab derivanti dal *Post Processor*).

Per quanto concerne le opzioni di allenamento, esse assumono una certa rilevanza nell'ottica ad esempio di definizione della durata dello stesso e nell'organizzazione dei dati, mentre per la visualizzazione di *RMSE* (radice dell'errore quadratico medio) ed altre variabili derivanti dalla funzione di training si può certamente utilizzare l'opzione *'Plots','training-progress'* (Figura 46 a destra), ma per evitare di appesantire troppo la procedura di addestramento è stata poi predispostata appositamente una sezione per ottenere graficamente questo tipo di risultati.

Tornando a presentare il caso specifico, la rete di cui abbiamo descritto l'architettura,

net_1, è caratterizzata, come detto, da 4 layers, 100 neuroni, le opzioni in Figura 46 e, soprattutto, da 6 *input* e 4 *output*. I primi sono rappresentati dai valori di accelerazione nel baricentro (nelle tre componenti), velocità longitudinale, angolo di sterzo del volante e velocità di imbardata, mentre i secondi dai valori di rollio ed LTR (sui tre assali), come già anticipato al termine della prima parte (Simscape) di questa trattazione. In fase di addestramento, chiaramente, è necessario avere a disposizione i valori in uscita, altrimenti non sarebbe possibile allenare efficacemente la rete. In un caso reale, con dati derivanti da una ECU, si può supporre di ottenere i parametri di rollio ed LTR richiesti per addestrare le reti da alcuni metodi sperimentali, come suggerito in (H. Yu and Özgüner, 2008).

Fatte quindi le doverose premesse, si è passato ad addestrare la rete in questione. Per fare ciò sono stati utilizzati 19 dei 23 dataset ricavati dal *Post Processor*. Questi sono stati caricati nel codice implementato per il training, e manipolati in modo da restituire le grandezze di interesse in forma adimensionale. Successivamente, la rete ha ricevuto quindi in input, per ognuna delle manovre di allenamento, i 6 parametri di ingresso e i 4 parametri di uscita sopra citati, chiaramente specifici per singola manovra. Evitando di generare nuovi *Layers*, ma continuando a sfruttare quelli preesistenti, è stato dunque possibile, nella pratica, generare 19 reti neurali basate sugli input ed output presentati in precedenza. Ognuna di queste è stata poi salvata in un'apposita cartella (*Ultime reti*, sempre all'interno delle librerie di *Networks*, da non confondere con *Ultime reti 2*) con annesse informazioni. Come intuibile, andando dalla prima all'ultima rete, il network risulta essere maggiormente addestrato. Viene fatto notare che tutte le reti ottenute, e relativi dettagli, possono essere salvate tramite la semplice attivazione di uno specifico indice all'inizio del codice *Ultimo_training_2*. Risulta peraltro importante evidenziare alcuni degli aspetti caratterizzanti questo procedimento, a partire dal numero di manovre scelte. All'inizio, infatti, si era pensato di usare solamente le 6 (5 alla fine, escludendo la condizione di rollover, introdotta puramente per scopo di osservazione) associate

al *New Maneuver* dell'interfaccia utente, anche perché si prevedeva di utilizzare le variabili ottenibili da un IMU (23), non da una ECU (6), e una casistica d'esame molto più ridotta. Tuttavia, alla fine si è preferito ampliare il discorso, avvicinandolo ad una situazione reale e dando maggiore credito all'attuale trattazione. Nell'ottica però di rendere efficace una stima basata su 6 input (e non 23), si è pensato di aumentare il numero delle manovre coinvolte nella fase di allenamento. In questo modo si è arrivati ad avere 19 dataset corrispondenti ad altrettante reti neurali, ognuna basata sull'esperienza di addestramento accumulata dalla precedente. In fase di analisi dei risultati verrà poi fatta una premessa sulle differenze interrecorrenti, a livello di risultati, ad esempio tra la rete neurale associata alla prima manovra di training e quella, invece, ad esempio, legata alla diciannovesima. Altro aspetto molto rilevante è poi il fatto che i dataset utilizzati per allenare le reti sono caratterizzati dall'avere contenuti uguali in termini di variabili considerate, ma le stesse, oltre che chiaramente a livello di valori, presentano anche durate molto differenti. Possiamo, infatti, avere prove da 40 secondi come prove da 310, fino ad arrivare oltre il migliaio di secondi per il Nurburgring. Questo genera differenti passi temporali (non costanti per via del fatto che si è scelto di adottare un *Solver* con *ode23t*, passo variabile) e un'influenza diversa sull'addestramento per ognuna delle manovre. Tuttavia, anche se idealmente si sarebbe dovuto avere tutte manovre più o meno uguali nella durata (in questo caso non possibile da realizzare perché si sarebbero perse troppe informazioni rilevanti per alcune manovre riducendone eccessivamente le tempistiche), i risultati ottenuti sono comunque stati soddisfacenti.

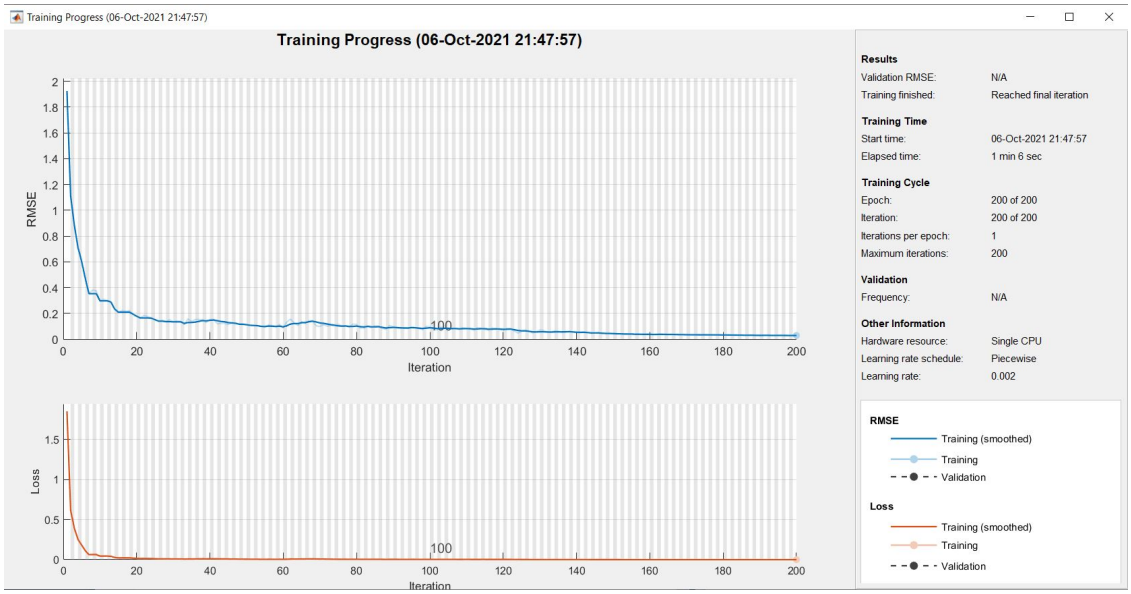


Figura 47: Esempio di training di *net_1* attraverso l'opzione *'Plots','training-progress'*

Prima di entrare nel merito della presentazione dei risultati, occorre però evidenziare che è stata addestrata un'ulteriore tipologia di rete neurale, *net_1_1*. Questa risulta essere direttamente correlata a *net_1*, con cui condivide architettura e principi generali, ma da cui si discosta per *input* ed *output*, oltre che chiaramente per obiettivi. Se la prima aveva funzione, partendo dai 6 input di accelerazione, angolo e velocità, di trovare un legame tra gli stessi e rollio e trasferimenti di carico verticale, in un'ottica di **stima** di questi ultimi in fase di testing, la seconda ha invece scopo **previsionale**. Sviluppando meglio quanto affermato, obiettivo di questa seconda rete era di essere addestrata in modo da poter prevedere preventivamente e non contemporaneamente agli input i valori di rollio ed LTR. Per fare ciò, i valori in ingresso e quelli in uscita da questo tipo di network (di cui esistono sempre 19 varianti progressive) sono esattamente gli stessi 4, semplicemente sfasati di una unità (passo) temporale, seguendo una logica molto simile a quella di uno dei numerosi esempi di utilizzo di deep neural network in ottica predittiva (contenuti in alcune cartelle specifiche, come ad esempio *Esempi utilizzati per capire funzionamento rete*

all'interno della folder *Networks*). Viene anticipato fin d'ora che però, contrariamente all'altra rete, questa ha fornito dei risultati piuttosto scadenti, come sarà possibile vedere nella prossima sezione della trattazione.
Prima di proseguire, risulta molto importante evidenziare quali siano le tempistiche di addestramento delle reti neurali. Attualmente, con l'opzione di *'Plots','training-progress'* attiva, in modo da vedere come ciascuna manovra influenzi il training, la durata del processo è stimabile in circa 40-45 minuti, ma è possibile ridurre notevolmente gli stessi trascurando l'opzione di cui sopra (scendendo a circa una decina). Chiaramente, molto dipende anche dalla macchina su cui si fa girare il programma. Risulta poi essere di interesse il fatto che anche questo aspetto della trattazione, ovvero il training delle reti neurali, è stato reso accessibile tramite UI del programma, come osservabile in Figura 48. Si sottolinea che questo è semplicemente un pulsante di rimando al codice, per poter intervenire sullo stesso (anche solo per attivare gli indici di salvataggio delle reti e delle figure) evitando al contempo di fare partire per errore un training indesiderato (procedura come detto non breve).

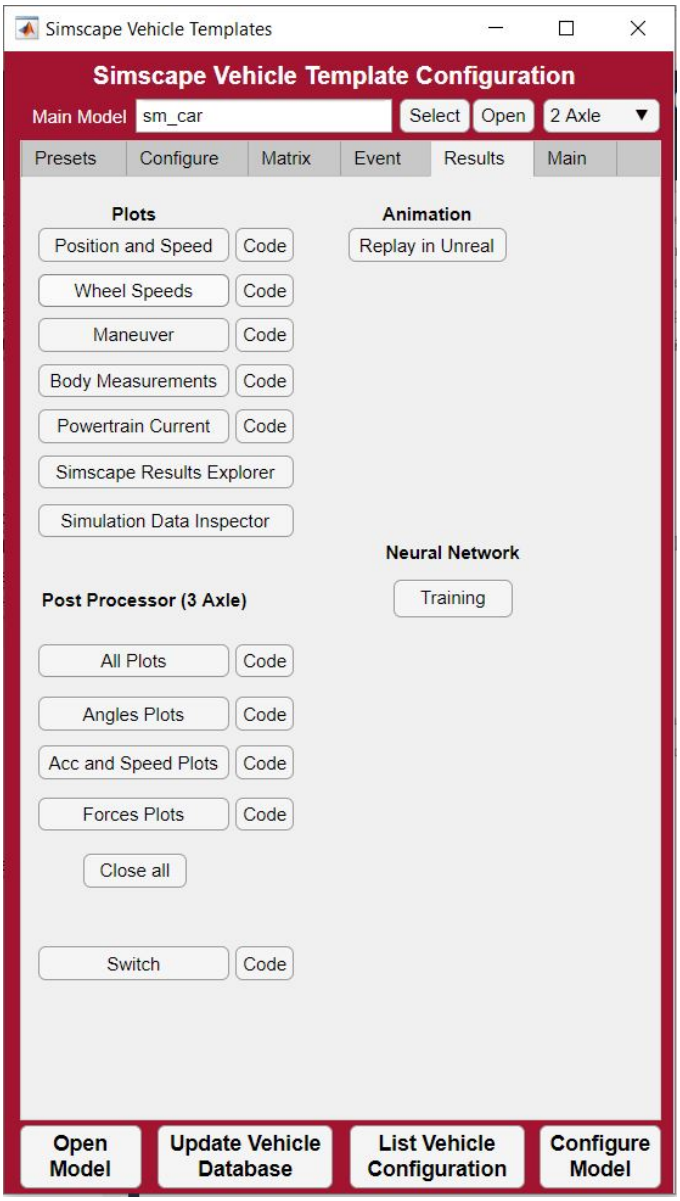


Figura 48: UI del programma aggiornata con addestramento delle reti neurali

Si evidenzia, infine che, come logico, terminata la fase di addestramento, le reti possano essere estratte e utilizzate in modo da verificarne l'efficacia per gli scopi per i quali sono state programmate. Di questa tematica, l'ultima di questa trattazione, si parlerà nella sezione immediatamente successiva.

Analisi e risultati fase di testing

Premessa

Effettuato l'addestramento delle reti, di cui abbiamo distinto due tipologie, *net_1* e *net_1_1*, rispettivamente pensate in un'ottica di stima e previsione delle variabili indici di ribaltamento del truck, vengono ora presentati dei risultati per 4 manovre differenti, chiamate con il nome con cui sono ritrovabili nel *Maneuver*:

- *Ice_patch*
- *CRG_Suzuka*
- *Rough_Road*
- *Plateau*

Ciascuna delle manovre sopra riportate è stata inserita in questa trattazione in modo da identificare differenti aspetti dell'efficacia delle reti neurali.
Particolarmente importante è evidenziare il principio con cui ognuno dei codici associati a ciascuna di esse sia stato composto. Si è, infatti, pensato di simulare con la maggiore fedeltà possibile, una situazione di applicazione reale. Conseguentemente, il codice viene articolato in modo che si abbia una prima fase di caricamento del dataset di una delle manovre di verifica (testing), contenente le informazioni usuali. Dopo aver standardizzato i dati in ingresso in questione (supposti ottenuti da una ECU), ci si è avvalsi della funzione *predictAndUpdateState*, utilizzabile grazie al *Deep Learning Toolbox*, per ottenere, a partire dai primi 6 *input*, i 4 *output* corrispondenti. Questo viene chiaramente fatto tramite la rete neurale *net_1* generata nella fase di training, e qui *non* riaddestrata, dato che, logicamente, i valori cercati non sono noti a priori attraverso altri metodi. In questo senso, è possibile sfruttare poi gli *output* ottenuti per addestrare *net_1* una volta terminata la procedura di stima degli stessi, ma l'attendibilità varia a seconda della procedura di training seguita e della manovra analizzata, come si vedrà entrando nello specifico dei risultati della fase

di testing. Si suppone, in ogni caso, di poter effettuare una procedura di training parallela a quella di risposta delle rete, anche nell'ottica di evitare di incorrere in tempi di addestramento lunghi che potrebbero pregiudicare le tempistiche risolutive del programma.

Tornando all'effettiva struttura tipica del codice sviluppato per le manovre di verifica, una volta ottenuti, in uscita da *net_1*, i valori *adimensionali* di rollio ed LTR, gli stessi sono poi diventati gli *input* della seconda rete neuralte, *net_1_1*. Qui, sfruttando sempre la funzione *predictAndUpdateState*, associata ad un ciclo for, è stato possibile scegliere di quanti passi temporali prevedere l'andamento delle grandezze cercate (come detto nella descrizione della fase di training, sempre rollio e trasferimenti di carico verticale), in un'ottica previsionale e non più di semplice stima.

Viene fatto notare, infine, che ognuna delle manovre ha, come solito, all'interno del proprio codice, degli indici di salvataggio delle figure associate. Risulta interessante portare poi all'attenzione che tutte le manovre pensate per la fase di testing dei network siano state implementate a livello di UI, come osservabile in Figura *UI finale*.

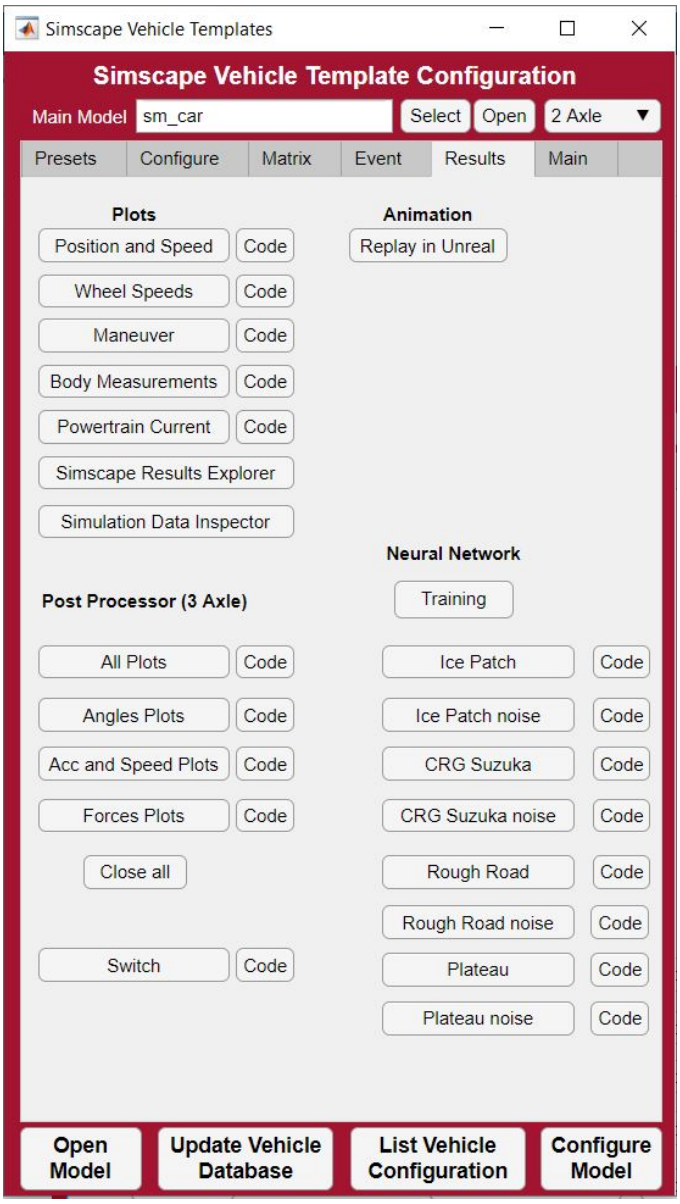


Figura 49: Interfaccia utente finale

Come osservabile nell'immagine immediatamente soprastante, per ogni manovra, oltre al caso in cui il codice tratta dei valori di input provenienti da un segnale completamente pulito, è stato poi previsto uno script dedicato allo studio dei risultati ottenuti per un caso di forte rumore. La presenza dello stesso è stata simulata tramite una funzione di aggiunta di rumore bianco gaussiano, supposto

completamente sovrapposto e di medesima intensità dei segnali in ingresso. In questo senso, si è probabilmente esagerato perché il programma restituisce dei grafici corretti (le reti funzionano) ma abbastanza difficili da leggere, e quindi non molto significativi se non, come si vedrà, per il caso della manovra *CRG_Suzuka*. Per ovviare a questo problema si può, semplicemente, intervenire sulla funzione *awgn* (riscontrabile sulle variabili di input per ogni codice "_noise" all'interno della cartella *Networks*), cambiando la potenza del segnale da 0 ad esempio a 10, dando così un ordine di grandezza di differenza tra il rumore e l'input effettivo, portando a grafici più leggibili e realistici (il caso di 0, sovrapposizione dei due segnali, è un estremo). In alternativa, è possibile prevedere, come si pensava di fare in questa trattazione, un'ulteriore implementazione, ovvero quella di un filtro, in ottica di simulare il più fedelmente possibile una casisitica reale. Purtroppo, per mancanza di tempo non è stato possibile farlo, ma può tranquillamente essere uno degli sviluppi futuri di questo lavoro di Tesi. Si sottolinea, peraltro, che, anche per il caso di aggiunta di rumore, si sono utilizzate le stesse reti addestrate su segnali "puliti". Queste danno risultati molto migliori rispetto a reti caratterizzate da "sporcizia" dei segnali, come sperimentato in varie alternative proposte nella cartella (sempre di *Networks*) *Alternative a rete completa*.

Ice_patch

La prima manovra presa in considerazione risulta essere quella di *Ice_patch*, generata (tramite omonimo codice *Ice_Patch*, cartella *Networks*) in modo da fare percorrere al veicolo scelto una traiettoria a chiocciola su percorso ghiacciato (coefficienti di attrito delle ruote ridotti rispetto a un caso di superficie normale). Questa manovra è rappresentativa di un caso in cui la rete neurale è messa davanti a una manovra conosciuta su una superficie non nota, ma ugualmente piana.
Sfruttando l'intero dataset in un'ottica di ottenimento dei 4 indicatori di ribaltamento (ovvero solo la rete *net_1_1*), sono stati ricavati gli andamenti subito sotto riportati.

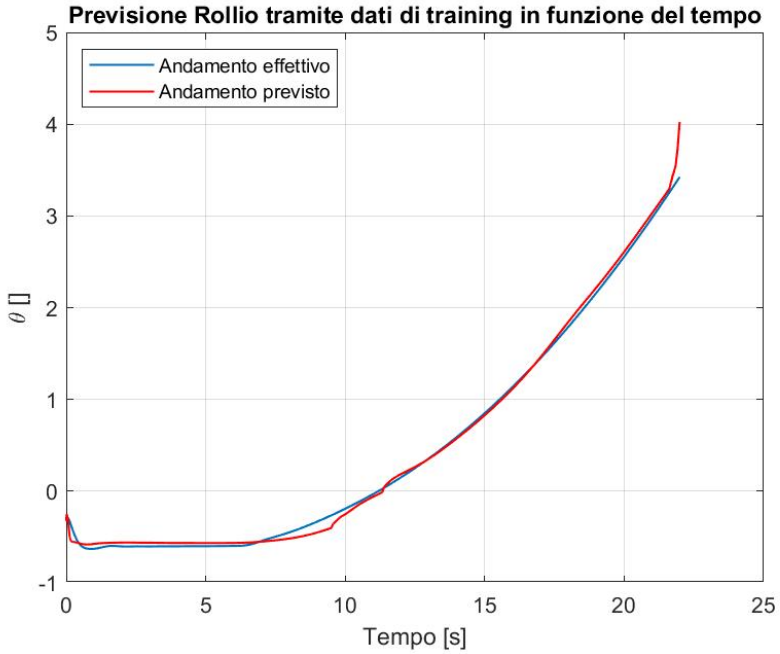


Figura 50: Stima rollio su intero dataset tramite *net_1*

Come valutabile da Figura 50, l'andamento simulato approssima molto bene quello rappresentante un caso reale (nel nostro caso, come al solito, questo deriva dalla prima parte della trattazione). Si fa notare che, rispetto a un caso analogo di manovra a chiocciola, ma su superfice piana, senza particolari variazioni, chiaramente

il risultato in termini di valore di angolo di rollio sarebbe stato più elevato. Altri esiti soddisfacenti sono stati poi ottenuti anche per i trasferimenti di carico verticale.

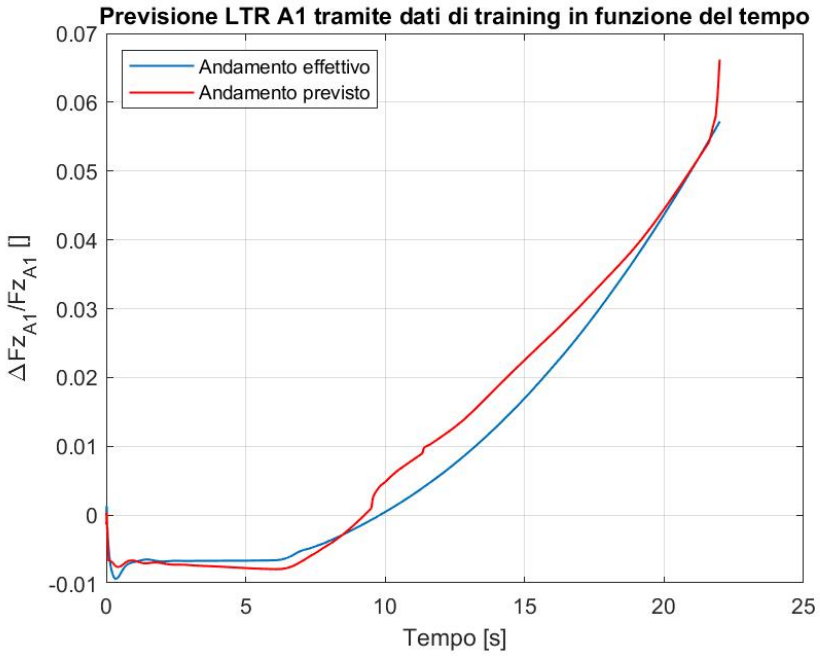


Figura 51: Stima LTR A1 su intero dataset *net_1*

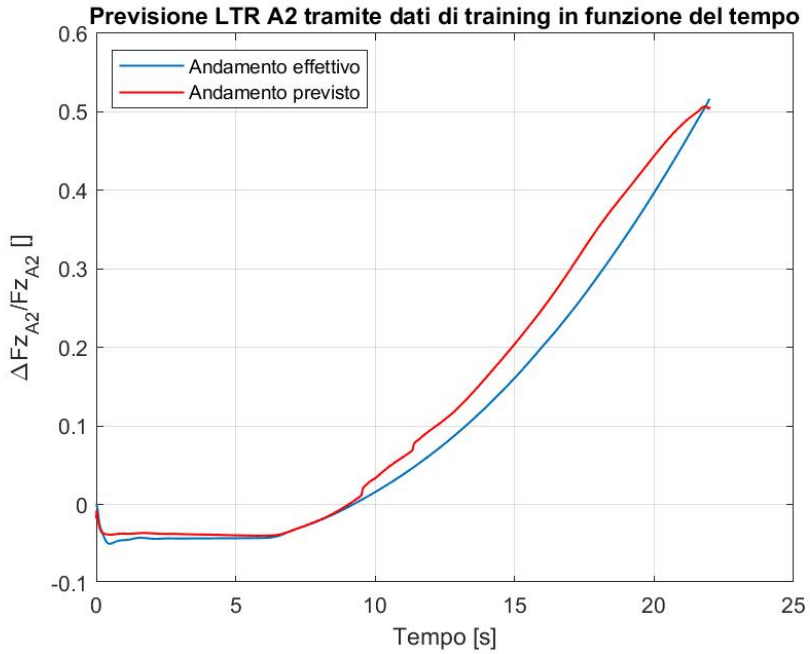


Figura 52: Stima LTR A2 su intero dataset *net_1*

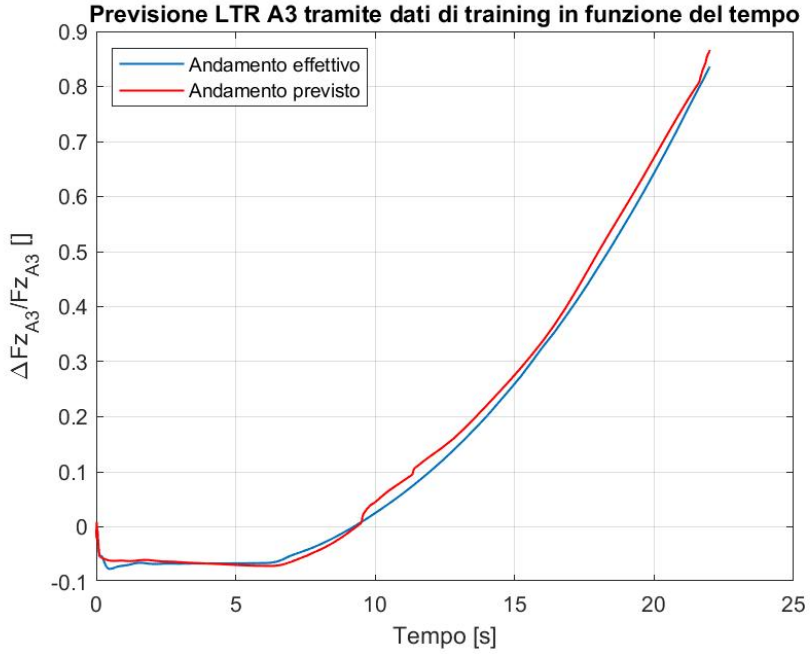


Figura 53: Stima LTR A3 su intero dataset *net_1*

È importante osservare che, per quanto concerne questa manovra, la stessa è caratterizzata dall'essere piuttosto breve e dall'avere un dataset relativamente contenuto. Da questo si capisce come, nonostante la rete approntata (*net_1*) abbia in memoria solo una manovra a chiocciola, e sia stata addestrata quasi totalmente con manovre svolte su superfici piane, la stessa abbia un'ottima efficacia. Viene fatto notare che è stata utilizzare la diciannovesima versione di training della rete neurale. Dato che questo è il primo caso applicativo considerato, si presentano anche gli effetti della rete ottenuta dalla prima manovra del processo di training, in modo da evidenziare ancora alcuni aspetti. Per evitare di appesantire troppo la trattazione, però, viene riportato solo l'andamento dell'angolo di rollio. È comunque possibile, se si desidera, osservare tutti gli altri diagrammi caricando *'network_1'*, *'layers_1'*, contenente le effettive informazioni, e *'options_1'*, il tutto al posto dei nominativi con 19 all'interno (procedura molto semplice, basta entrare e agire all'inizio del codice *Ice_Patch*).

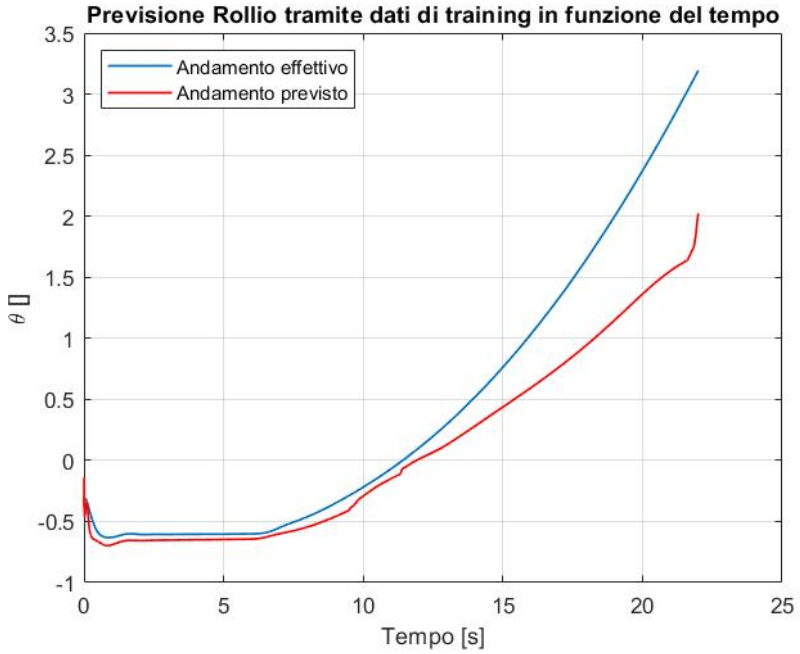


Figura 54: Stima rollio su intero dataset tramite *net_1*, ma nella versione ricavata dalla prima manovra di training

Riferendoci a Figura 54, appare evidente come la stima risulti notevolmente peggiorata. Risultati simili è possibile osservarli anche sugli LTR, ed in generale su ognuna delle manovre di testing su cui si decidesse di usare una rete *net_1* salvata da una delle manovre iniziali o di intermezzo del training (ancora in una fase di completamento a quel punto). Questo conferma che il training, progressivamente, permette di ottenere delle reti neurali più performanti in caso generale. Chiaramente, è possibile anche che una delle manovre introdotte in fase di addestramento non apporti sostanziali benefici, ma di questo si parlerà meglio nella sezione *Considerazioni e possibili sviluppi futuri: IA*.

Proseguendo nella presentazione dei risultati, passiamo ora agli aspetti previsionali, ovvero alla rete *net_1_1*. Questa, come detto, si basa su quanto ricavato nella parte precedente, ovvero dal network *net_1*. Nell'ottica di realizzare un confronto con quanto accade effettivamente (dati standardizzati del *Post Processor*), è quindi necessario utilizzare solo una parte del dataset a disposizione per ricavare rollio ed LTR. Qui il ridotto contenuto dello stesso entra in gioco in maniera fortemente negativa. Con meno dati a disposizione, peggiorano infatti tutti gli andamenti simulati nelle figure 50, 51, 52 e 53 derivanti dalla rete *net_1*. Questo però comporta anche che la rete *net_1_1*, già non ottimale anche per dataset molto grandi (si vedano gli esiti per la manovra *CRG_Suzuka*), porti, anche per uno spazio temporale ragionevole per una previsione (circa 2 secondi), dei risultati insoddisfacenti perché, come detto, le grandezze in uscita dalla prima rete altro non sono che gli input della seconda. Sempre nell'ottica di non appesantire inutilmente il lavoro di Tesi, vengono presentati i risultati per il trasferimento di carico dell'assale posteriore in questo caso specifico (l'andamento migliore insieme a quello del rollio se facciamo riferimento alle figure 53 e 50, oltre a quello più indicativo, visti i valori, di un eventuale rischio di ribaltamento del mezzo).

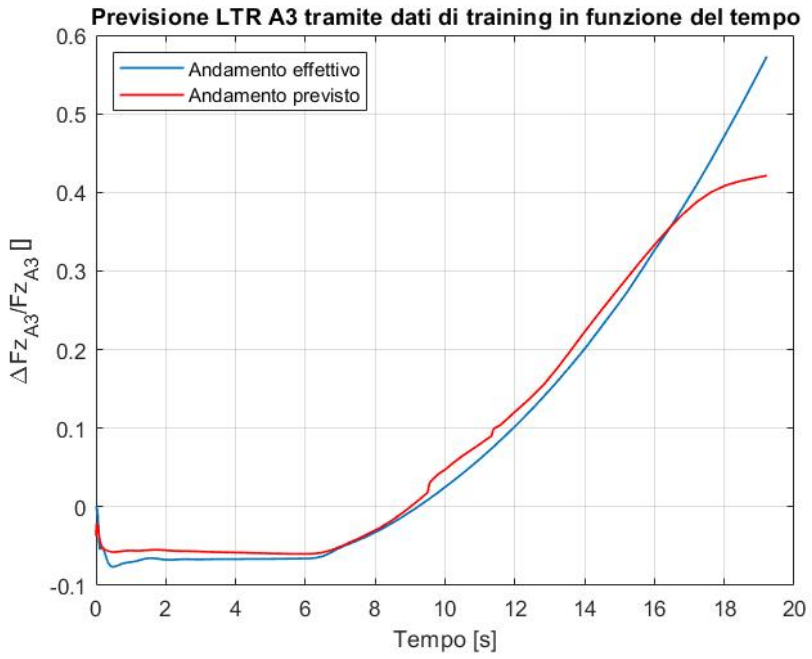


Figura 55: Stima LTR A3 su dataset parziale tramite *net_1*

Come anticipato, il valore simulato risulta essere meno preciso di quello elaborato con tutto il dataset a disposizione. Questo si riflette anche nei diagrammi successivi, dove possiamo osservare prima (Figura 56) l'andamento previsto dalla rete *net_1_1* per la parte di dataset usato a scopo predittivo (confrontato chiaramente con il corrispettivo reale) e, subito dopo (Figura 57), l'andamento complessivo simulato (ovvero LTR A3 stimato da *net_1* e previsto da *net_1_1* in un unico diagramma) a riscontro con quello reale. Come evidente, la rete programmata per scopo predittivo presenta una scarsa efficacia. Si era cercato di porre parzialmente rimedio, in tal senso, anche se a scopo puramente teorico (essendo nella realtà i valori reali ottenuti da *net_1* e non forniti) provando ad aggiungere allo script una parte che permettesse l'aggiornamento dei dati in ingresso a questa seconda reti con quelli considerati reali (ovviamente standardizzati) provenienti da Simscape. Questo ha permesso di ottenere degli esiti migliori, ma non tali da potersi considerare rilevanti in un'ottica di presentazione dell'efficacia o di eventuali sviluppi futuri della rete.

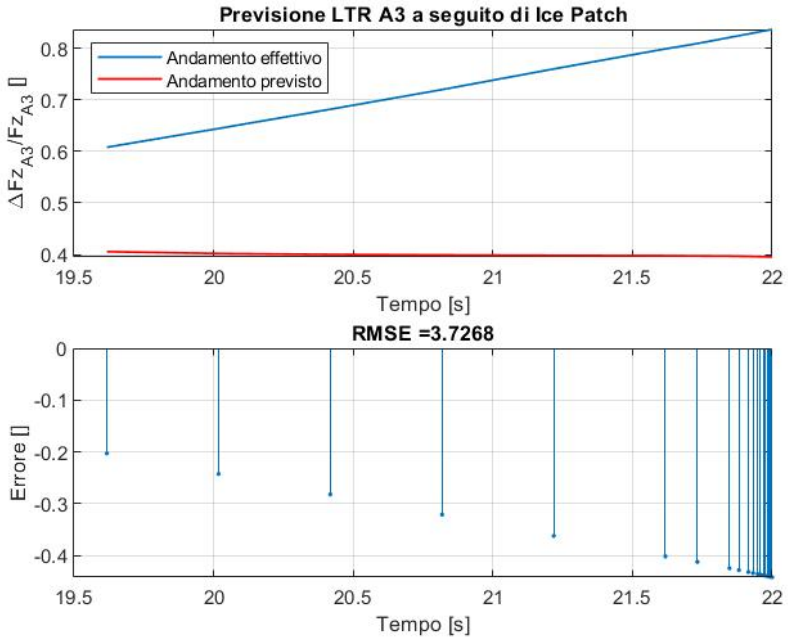


Figura 56: Previsione tramite *net_1_1* di LTR A3

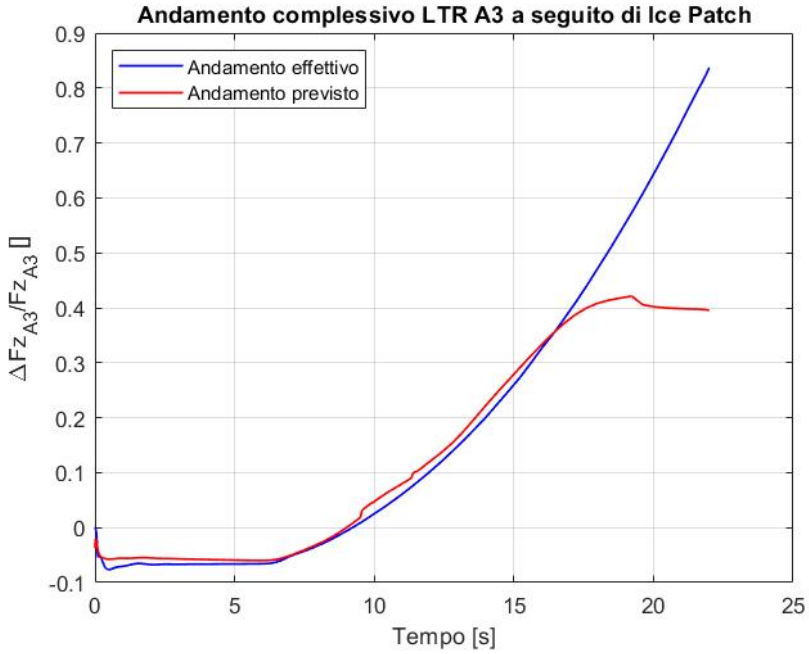


Figura 57: Confronto tra andamento di LTR A3 complessivo simulato e previsto con quello reale

Non si presentano, per quanto affermato in precedenza, gli esiti associati al codice *Ice_Patch_noise* e si ricorda che questi verranno riportati in parte unicamente per la manovra *CRG_Suzuka*, dove i grafici risultano essere ancora sufficientemente leggibili.

CRG_Suzuka

La manovra che si va ora a presentare è quella principe della dimostrazione dell'efficacia della scelta delle reti neurali per poter stimare gli andamenti degli indici di rollover. Essa infatti è una manovra completamente ignota alla rete, ma sviluppata su una superficie piana, che è quella della quasi totalità delle manovre di addestramento e con un dataset molto numeroso, che permetterà di ottenere risultati migliori non tanto dal punto di vista della *net_1*, quanto da quello della *net_1_1*. Si fa notare che, per entrambe, si è utilizzata la diciannovesima versione dei network, che si è confermata essere quella relativamente più efficace per i fini di questa trattazione (sarà così anche per le altre due manovre di testing ancora da presentare). Questa manovra è quindi, come detto, rappresentativa di una condizione in cui non si conosce una traiettoria molto complessa, ma è nota la superficie su cui la stessa si stia svolgendo.

Presentando i risultati ottenuti per dataset intero con la rete *net_1*, è possibile osservare delle stime ancor più efficaci di quelle presentate per il caso *Ice_Patch*.

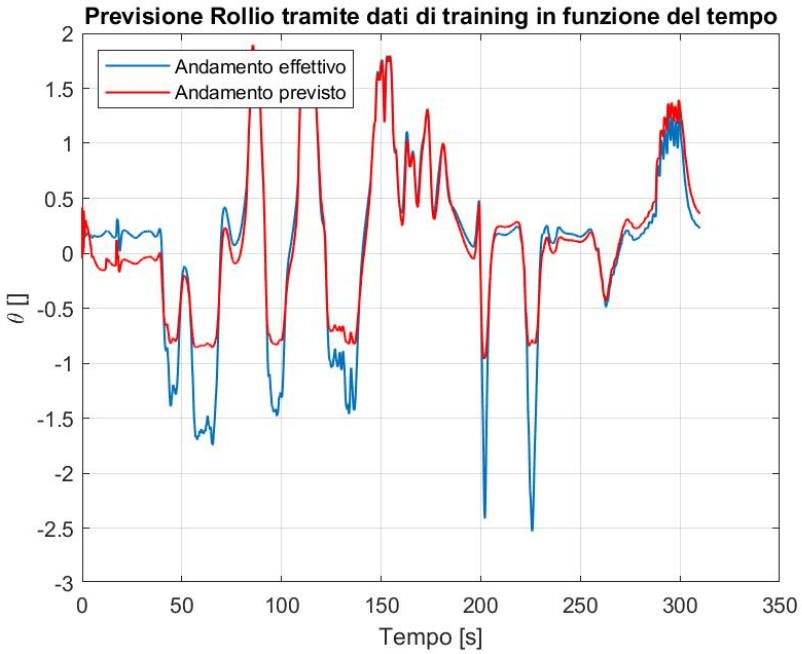


Figura 58: Stima rollio su intero dataset tramite *net_1*

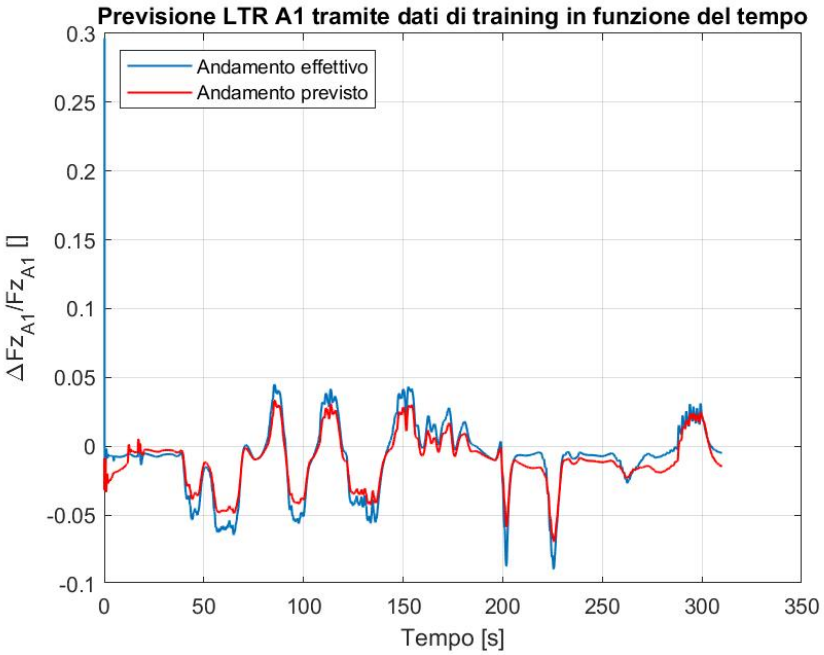


Figura 59: Stima LTR A1 su intero dataset *net_1*

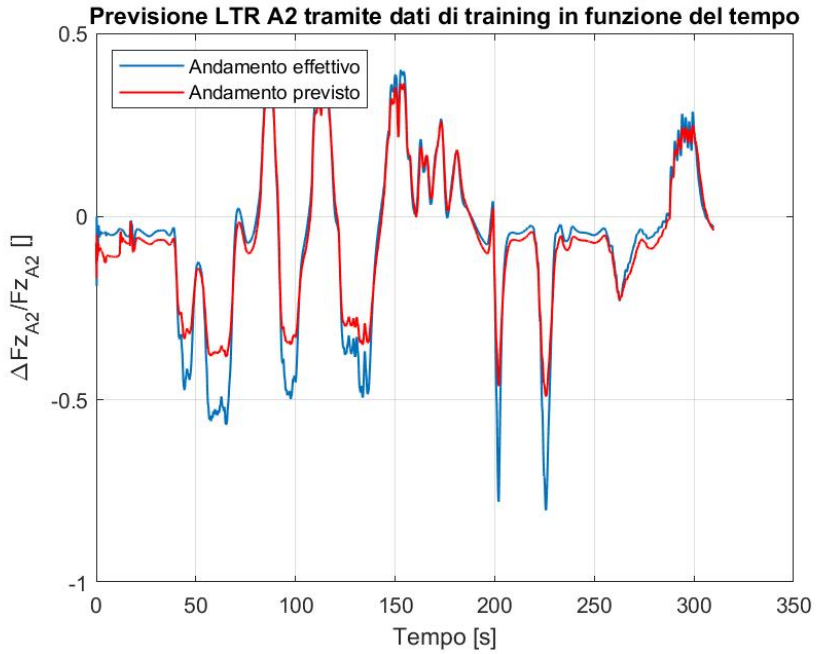


Figura 60: Stima LTR A2 su intero dataset *net_1*

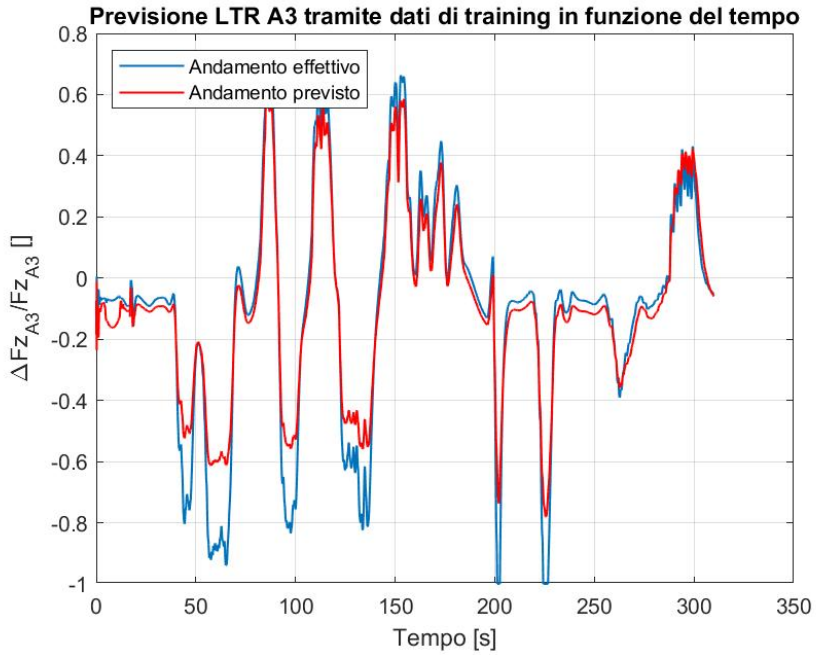


Figura 61: Stima LTR A3 su intero dataset *net_1*

Risulta particolarmente rilevante sottilineare come, nonostante gli andamenti reali siano anche notevolmente complessi (dato che ci sta comunque muovendo su un circuito di Formula 1), gli stessi vengano approssimati efficacemente dalla rete *net_1*, che si ripete non essere stata addestrata minimamente alla manovra in questione, risultando particolarmente valida per l'effettuazione di una stima dei valori di LTR e rollio.

Per quanto concerne la rete neurale implementata con scopi predittivi, anche questa, forte di un maggiore numero di dati a disposizione, permette di ottenere risultati non ottimali, ma comunque interessanti. Come nel caso precedente, per evitare di riportare un numero eccessivo di figure, vengono presentati gli andamenti per rollio ed LTR A3, con il primo come discontinuità rispetto alla presentazione degli esiti sull'*Ice_Patch*, dato che, in questo caso, pur provando a predire l'andamento della manovra andando 8 secondi avanti nel tempo, la stima realizzata da *net_1* rimane pressoché invariata.

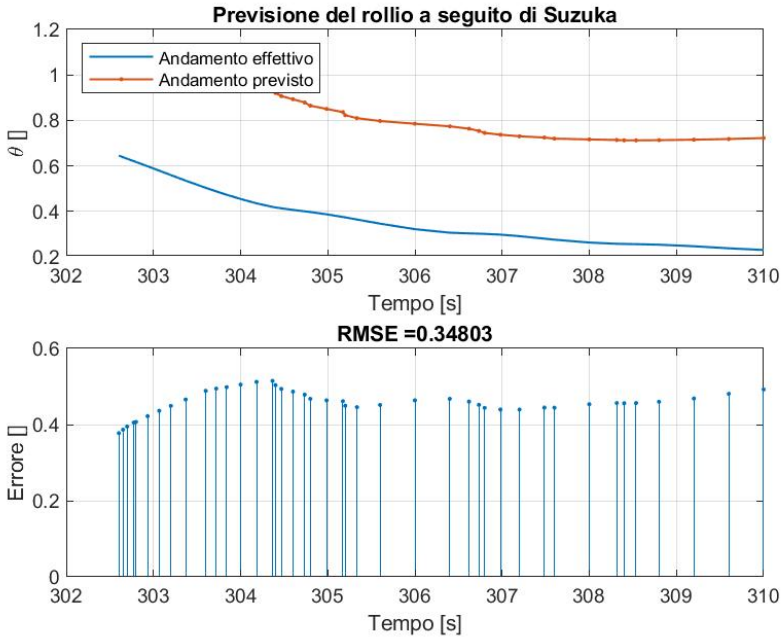


Figura 62: Previsione tramite *net_1_1* del rollio

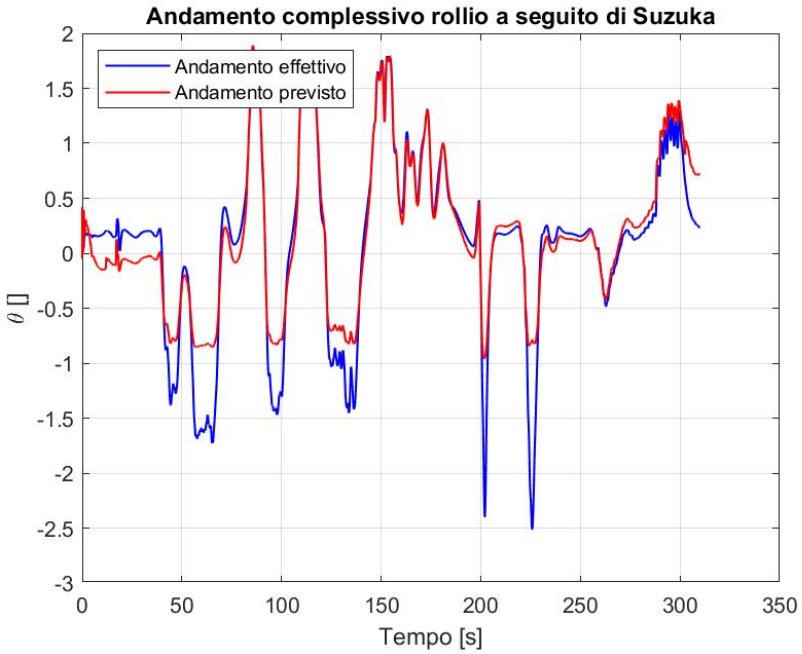


Figura 63: Confronto tra andamento del rollio complessivo simulato e previsto con quello reale

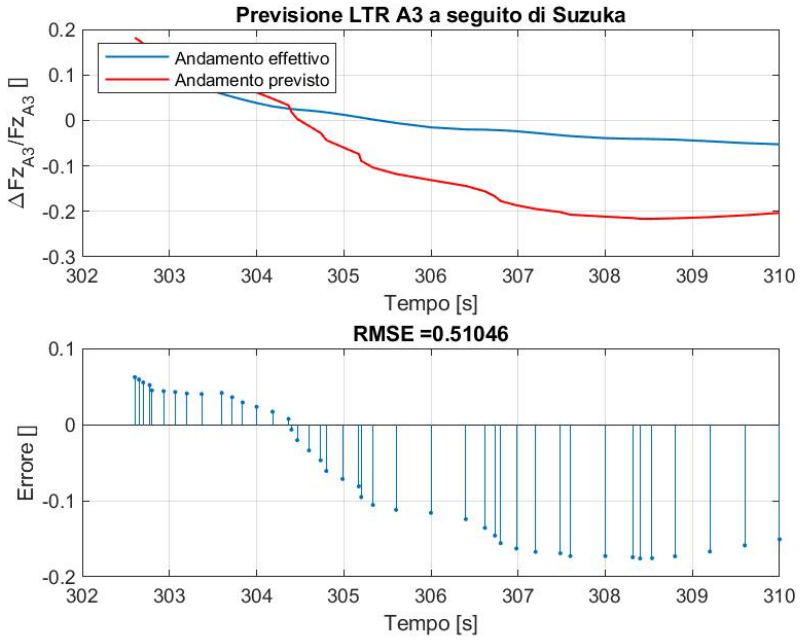


Figura 64: Previsione tramite *net_1_1* di LTR A3

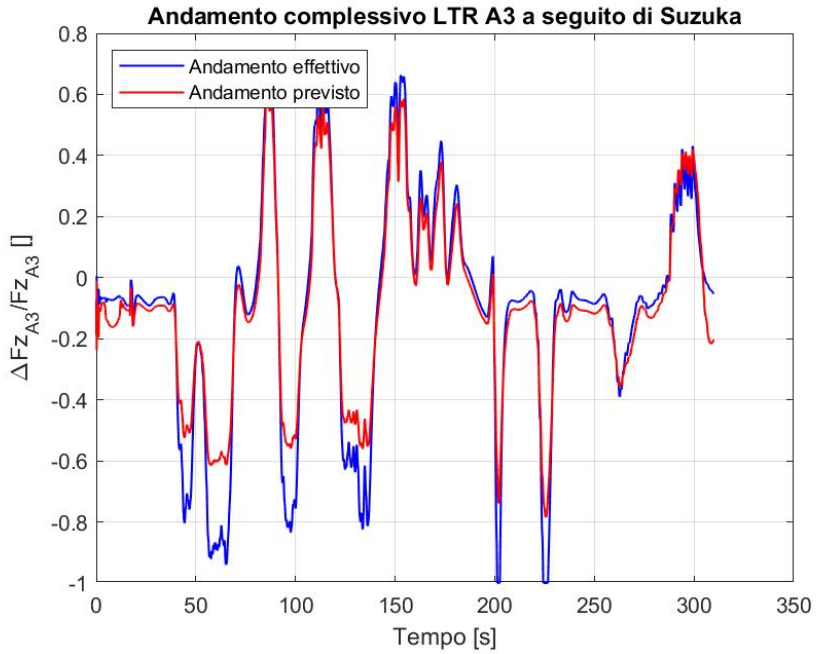


Figura 65: Confronto tra andamento di LTR A3 complessivo simulato e previsto con quello reale

Osservando i risultati sopra presentati, è possibile affermare che, in un'eventuale prosecuzione del lavoro, per migliorare *net_1_1*, si possa provare ad allenare e utilizzare la stessa solo dopo un certo intervallo di tempo (da individuare con maggiore precisione).

Si presenta, infine, per questa manovra, l'andamento dell'angolo di rollio ricavato con segnali in ingresso nella rete *net_1* con rumore sovrapposto. Come visibile, la rete riesce, in quelle che sono condizioni critiche da elaborare per la natura dei segnali di input, ad ottenere comunque un buon risultato (tenuto conto proprio di quanto siano disturbati i segnali delle grandezze di interesse).

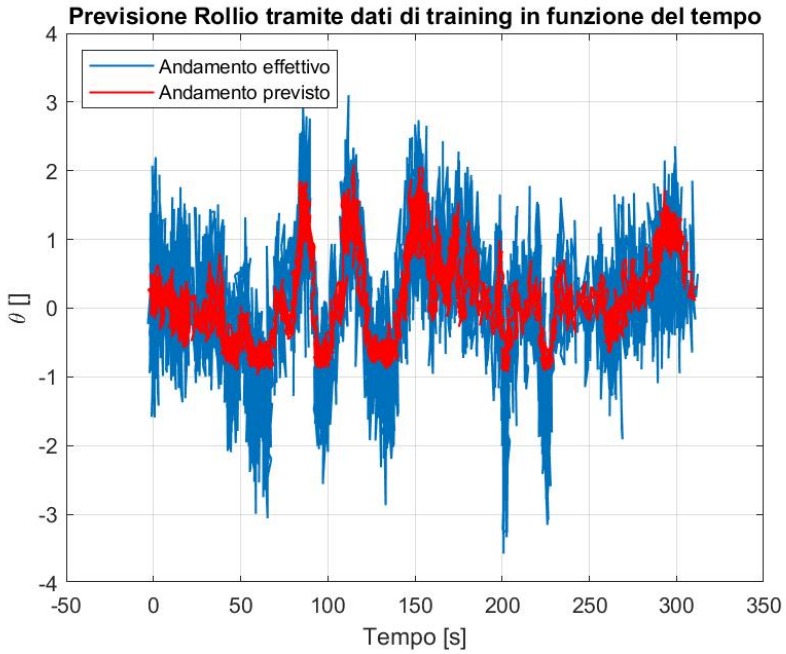


Figura 66: Andamento dell'angolo di rollio stimato da *net_1* in condizioni di massimo rumore

Rough_Road

La manovra esaminata in questa sezione è caratterizzata dallo svolgersi su una superficie non piana, caratterizzata dalla presenza di diversi dossi di altezza differente. Questa manovra è rappresentativa di un caso in cui sia la stessa che la superficie risultano entrambe non note. Si fa notare che la rete, come già detto più volte, è stata allenata quasi completamente su manovre che si svolgevano su superfici, invece, piane. Questo ha comportato la presenza di una traiettoria nuova, cui la rete non era abituata e, contemporaneamente, la presenza di una nuova superficie di svolgimento del moto.

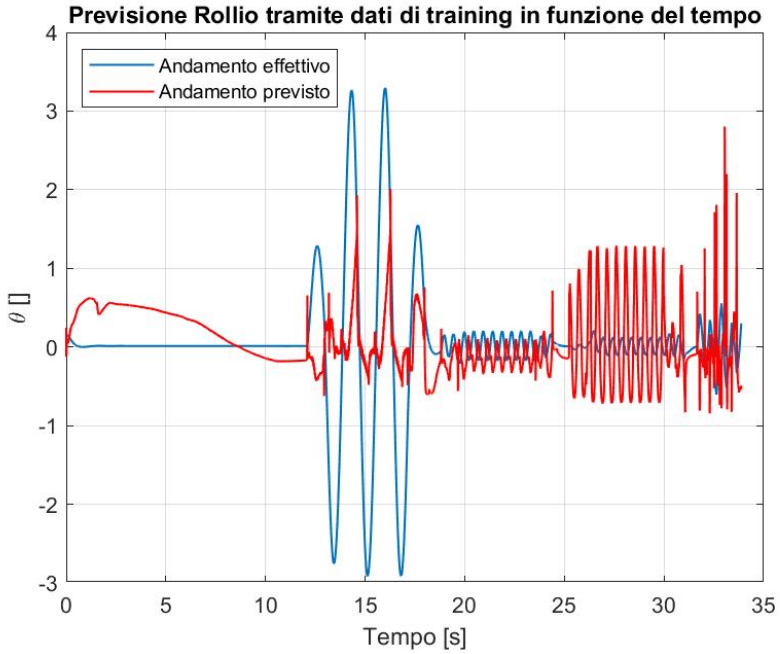


Figura 67: Stima rollio su intero dataset tramite *net_1*

Osservando l'andamento ottenuto per il rollio, lo stesso si presenta chiaramente erroneo, ma molto interassante considerate le premesse fatte per questa manovra. Viene evidenziato che, in questo caso, la relativamente breve durata della stessa non coincideva però con un basso quantitativo di dati. Considerazioni analoghe possono poi essere fatte per quanto concerne gli esiti ricavati per gli LTR sui tre assali.

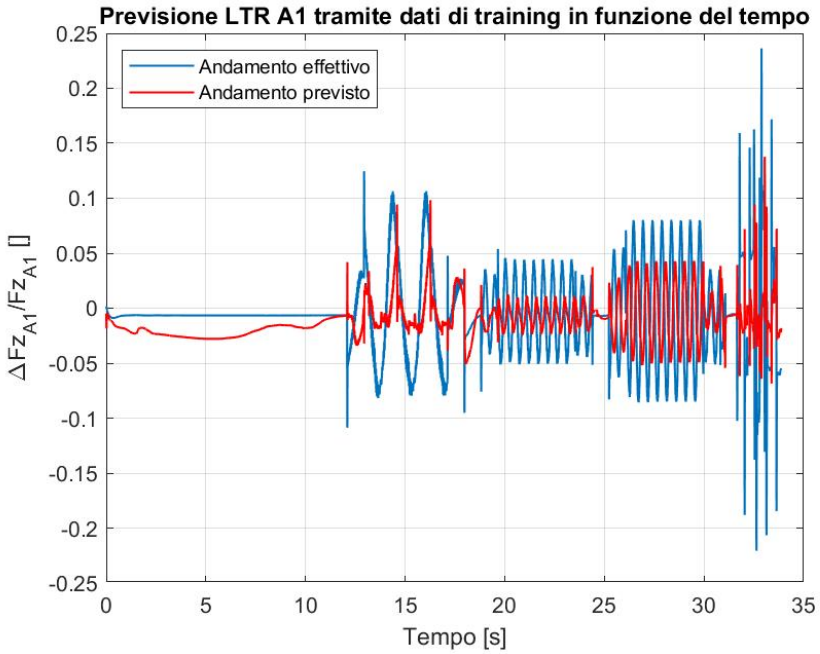


Figura 68: Stima LTR A1 su intero dataset *net_1*

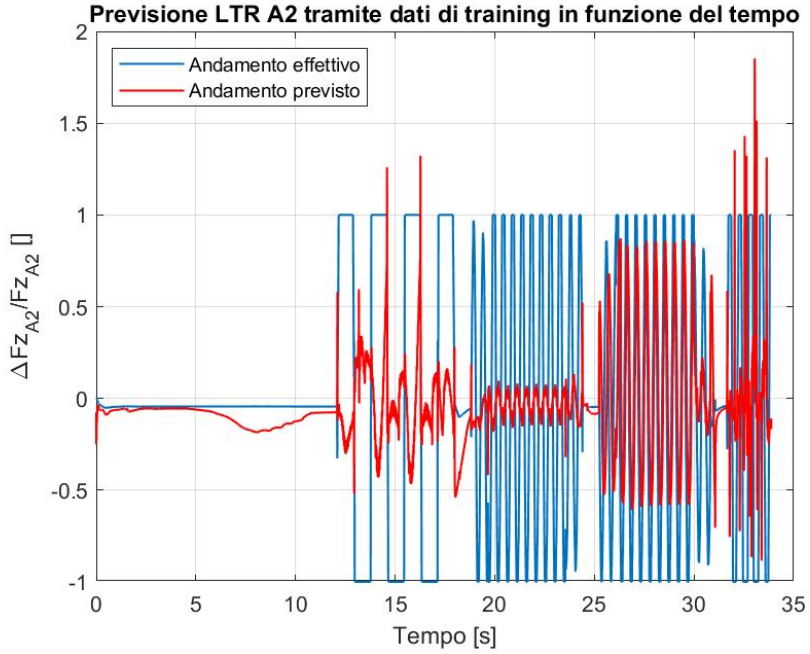


Figura 69: Stima LTR A2 su intero dataset *net_1*

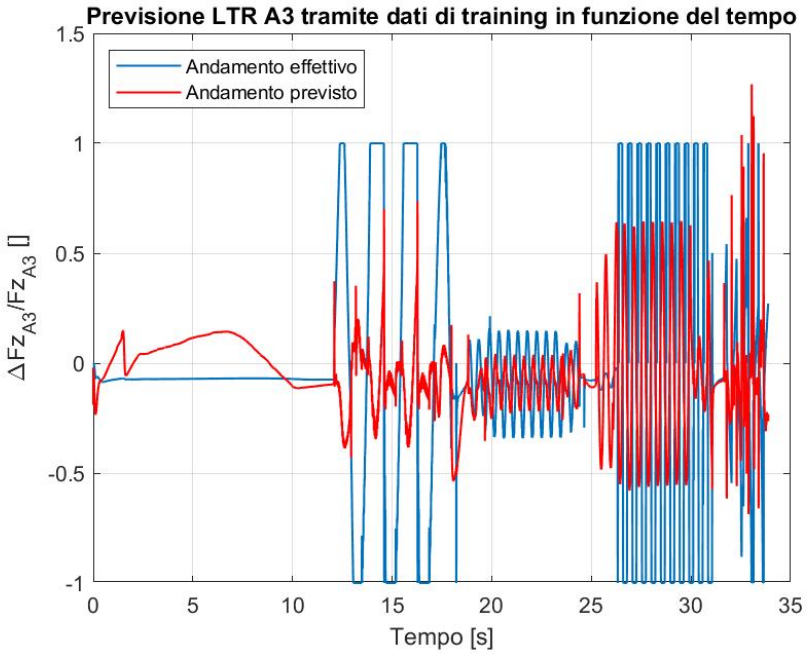


Figura 70: Stima LTR A3 su intero dataset *net_1*

Si evidenzia come, riferendoci in particolare alle figure 68, 69 e 70, questa manovra risulti non essere caratterizzata, nonostante il mezzo (che si ricorda essere un truck) di cui si sta analizzando il moto, da una condizione di incipiente rollover, che era invece possibile ravvisare nei casi di *Ice_Patch* e *CRG_Suzuka* (dove era stato necessario intervenire manualmente per cambiare i parametri di velocità del veicolo, per evitarne il ribaltamento su un circuito obiettivamente non pensato per lo stesso). Questo è semplicemente legato al fatto che la traiettoria richiesta al camion, in questo caso, non presentava curve, ma era rettilinea, per scelta di analisi di una condizione differente.

Per quanto concerne, invece, gli aspetti previsionali, questi lasciano molto a desiderare, come è possibile vedere riferendoci alle figure di esempio 71 e 72.

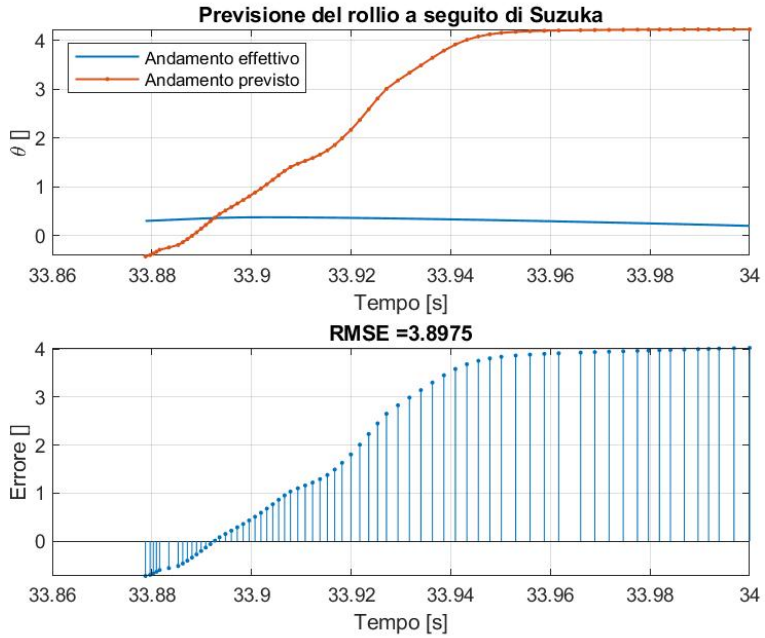


Figura 71: Previsione tramite *net_1_1* del rollio

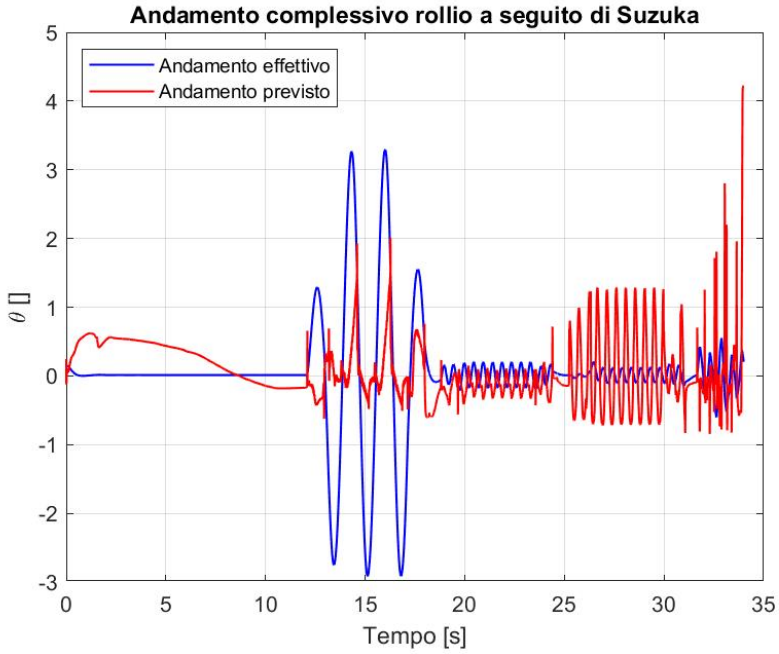


Figura 72: Confronto tra andamento del rollio complessivo simulato e previsto con quello reale

Plateau

L'ultima manovra presentata ricalca i concetti seguiti per la *Rough Road*. Anch'essa, infatti, rappresenta una manovra complessa (più di quanto visto sul percorso con i dossi) su una superficie non nota alla rete, caratterizzata da una fase di salita seguita da un plateau. In questo caso, gli esiti sono risultati insufficienti sia considerando gli aspetti previsionali (*net_1_1*), sia quelli di stima *net_1*, che riescono a seguire solo parzialmente gli andamenti reali, con l'unica eccezione rappresentata da Figura 74.

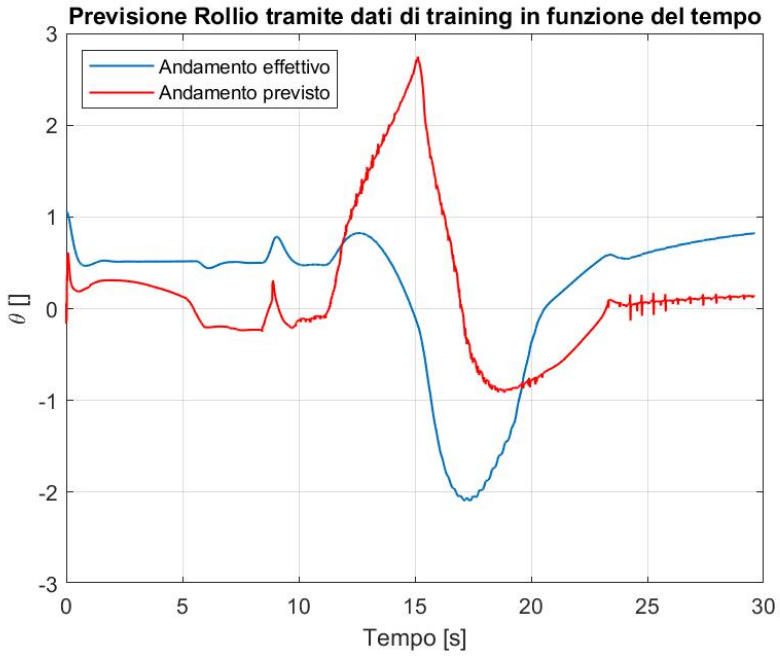


Figura 73: Stima rollio su intero dataset tramite *net_1*

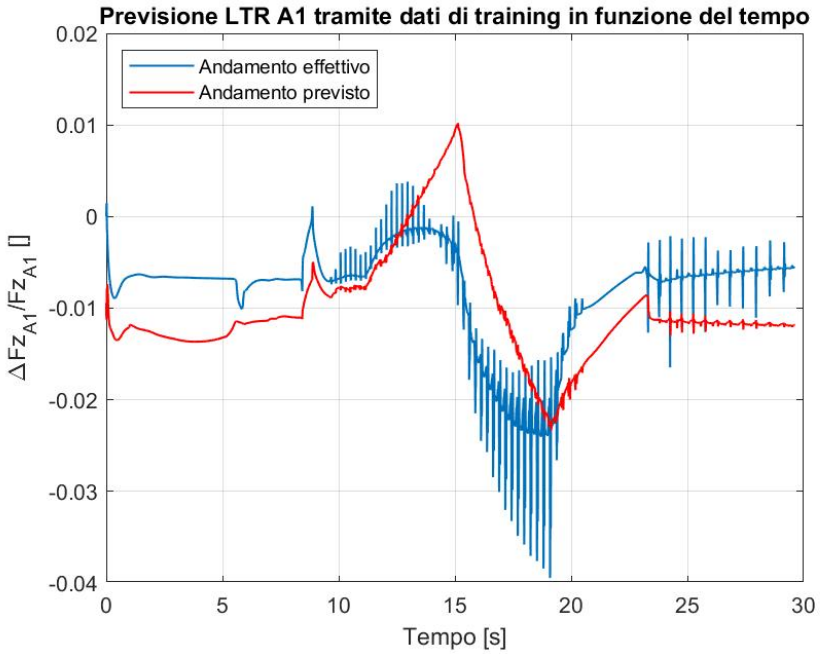


Figura 74: Stima LTR A1 su intero dataset *net_1*

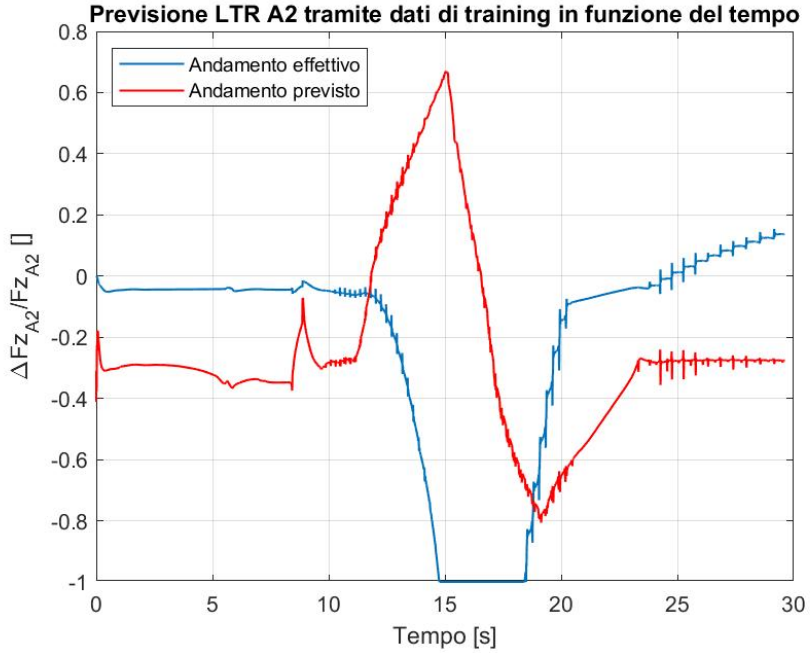


Figura 75: Stima LTR A2 su intero dataset *net_1*

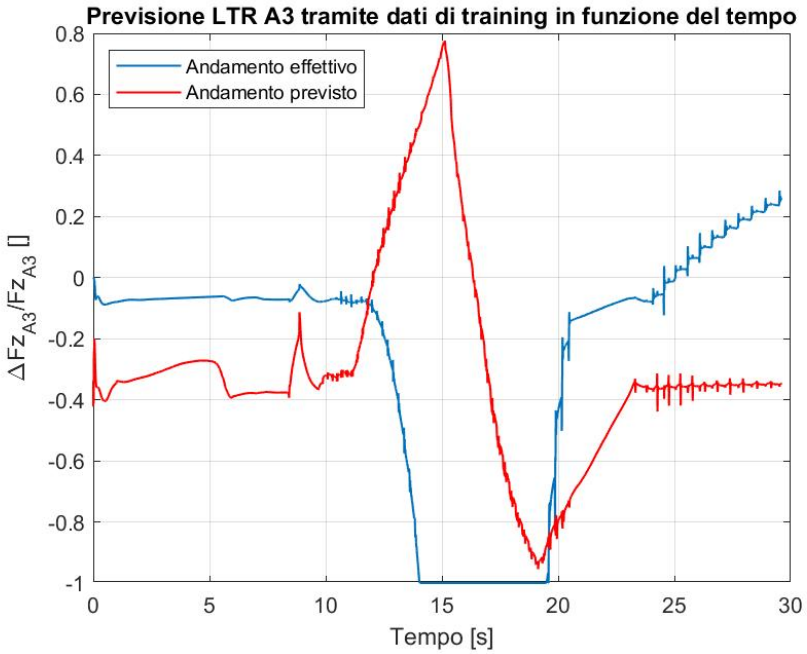


Figura 76: Stima LTR A3 su intero dataset *net_1*

Visti i risultati ancora da perfezionare anche sulla fase di stima, non vengono presentati, in questo caso, quelli legati agli aspetti di predizione di questa trattazione. Risulta però rilevante entrare più nel merito della manovra. La stessa, infatti, è stata caratterizzata dall'avere un tratto curvilineo, che porta ad una fase protratta di rischio di ribaltamento del mezzo (figure 75 e 76). Questo movimento, correlato alla superficie in salita, non essendo mai stato visto dalla rete neurale in queste condizioni in fase di addestramento, ha fatto sì che la stima degli indici di ribaltamento risultasse molto approssimativa, soprattutto dal punto di vista degli andamenti (che, invece, pur presentando valori erronei, venivano rispettati nel caso della *Rough_Road*, caratterizzata da una manovra molto semplice).

Considerazioni e possibili sviluppi futuri: IA

Al termine di questo capitolo è possibile affermare che la fase di stima e previsione da parte delle reti neurali fornisca risultati divergenti. La prima, infatti, consente di ottenere ottimi esiti per manovre anche molto complesse non conosciute dalla rete (*CRG_Suzuka*) o per superfici non note di natura però molto semplice (*Ice_Patch*), la seconda, invece, è relativamente inaffidabile se non riferita a tratti molto brevi di tempo e facendola basare su un numero rilevante di dati pregressi (*CRG_Suzuka*). Per quanto concerne invece, manovre non note svolte su superfici molto complesse (*Rough Road* e *Plateau*), entrambe le reti, e in particolare la prima (*net_1*, da cui dipende *net_1_1* risultano avere ancora necessità di addestramento. Si può dire con ragionevole certezza, infatti, che allenando la rete su manovre aventi superfici simili o uguali a quelle complesse citate in precedenza (aumentandone quindi il numero in fase di training), i risultati migliorerebbero. In ottica di possibili sviluppi futuri si suggerisce quindi di andare in questa direzione o, in alternativa, dato in questo momento le reti, e in particolare quella di stima, sono addestrate *senza* i parametri della strada (ad esempio i valori di attrito, che si possono supporre, o l'altitudine, che è di per sé nota a priori), è possibile provare ad aggiungere questi valori in fase di addestramento (passando da 6 a 13 input, per via dell'aggiunta di z e μ per ciascuna delle ruote), con certezza ancora maggiore della precedente che gli esiti migliorino in seguito a una scelta del genere. Chiaramente, l'ideale sarebbe implementare entrambi gli aspetti contemporaneamente.

Si evidenzia, inoltre, un'ulteriore particolarità di interesse. Infatti, come detto, tendenzialmente il training porta a reti con risultati migliori più si va avanti nell'addestramento (come detto più volte). Questo, tuttavia, è valido per manovre (e conseguentemente superfici) sconosciute alla rete neurale. Nel caso in cui si dovesse andare a studiare una manovra già nota e contenuta all'interno della procedura di addestramento, chiaramente è bene studiare esattamente quel caso specifico, sfruttando non l'ultima rete del training, ma quella associata alla precisa manovra

in questione. Discorso analogo può anche essere fatto per le superfici, suggerendo che, un'ulteriore tipologia di approccio, potrebbe essere semplicemente quella di studiare specifiche tipologie di reti neurali a seconda del terreno su cui dovrà poi muoversi il veicolo.

Infine, altra tipologia di approccio, complementare a quelle sopra presentate, è occuparsi di una definizione più puntuale dei dataset, cercando di averli tutti della stessa dimensione e che le manovre contenute al loro interno possano essere tutte significative una volta avviate (in questo lavoro è già stato avviato un processo simile, ma portarlo a termine avrebbe richiesto un tempo eccessivo). È possibile, comunque, affermare che le reti neurali siano una strada efficace per ottenere la stima (e, in futuro, magari la previsione) delle variabili indici di ribaltamento dei mezzi pesanti, anche se, chiaramente, risulta necessario approfondire alcuni aspetti per migliorare le performance su superfici non note. Risulta particolarmente importante evidenziare che le stesse provvedono alla creazione di un modello dinamico, facilmente aggiornabile e che garantisce tempi di risposta immediati, motivo per cui è effettivamente interessante esplorare questo campo. Nell'ottica di sviluppi futuri, riprendendo quanto già anticipato nelle sezioni precedenti, può poi essere interessante tarare differentemente la presenza di rumore e introdurre lo studio di appositi filtri, oltre che, chiaramente, evidenziare che è possibile aggiungere nuove manovre alla fase di training e sfruttare, in questo senso, ad esempio quelle di testing una volta che si è terminato di analizzarle.

Ultimo aspetto di interesse è poi fare notare, ancora una volta, che la rete di stima delle grandezze si basa su sei variabili derivabili sì da una ECU, ma utilizzabili anche a priori, dato che sono parametri caratteristici di una qualunque pianificazione della traiettoria. Questo permette, eventualmente, di poter studiare in anticipo, senza andare sul campo, quello che dovrebbe essere il moto di un mezzo che deve seguire un certo percorso, riuscendo a sapere già, grazie alla rete neurale, cosa aspettarsi nei punti più critici. Chiaramente, perché una stima del genere sia sempre

attendibile (abbia quindi errori minimi), è necessario ampliare notevolmente la fase di addestramento (passando, con ogni probabilità, da decine di minuti a decine di ore), ma questo è forse il risvolto pratico più interessante di questo lavoro dato che, senza sapere le caratteristiche fisiche del mezzo (se non in fase di addestramento) e nient'altro che una conoscenza di traiettoria, è possibile ottenere in tempi immediati quello che è il rischio di ribaltamento di quel mezzo (pesante, ma non necessariamente). Questo andrebbe, con ogni probabilità, a rendere definitivamente obsoleta la rete *net_1_1*, per la quale si possono comunque provare numerose soluzioni (dato che è legata alla prima).

Il lavoro qui presentato chiaramente ha validità per un veicolo, ma si evidenzia, chiaramente, il fatto che questa tipologia di IA sia utilizzabile nei campi più disparati, visto il potenziale a disposizione.

Conclusioni

Al termine di questo lavoro di Tesi è possibile affermare che si è riusciti a correlare efficacemente l'ambiente *Simscape Vehicle Templates* con quello legato al **Deep Learning Toolbox**. Attraverso il primo si è generata, infatti, una grande quantità di dati legati a specifiche manovre e superfici su cui è stato fatto muovere un mezzo pesante, un truck di cui si voleva studiare il rischio di ribaltamento mentre, attraverso il secondo, raccogliere questi dati ed esaminarli (in una fase di training e in una di testing) tramite reti neurali (IA) in modo da poter estrarre delle grandezze indici del pericolo di rollover corso dal veicolo stesso. In quest'ottica, rimandando alle considerazioni effettuate per ognuna delle due macro parti di questa trattazione al termine di ciascuno dei rispettivi capitoli per maggiori dettagli (si vedano le pagine 65 e 103), si può dire di essere riusciti ad ottenere una rete neurale adatta alla stima delle grandezze cercate. Questa è caratterizzata da tempi di risposta immediati, un modello dinamico in grado di essere aggiornato molto facilmente ed appositi codici, per quanto presenti ancora necessità di implementazioni, specie in fase di addestramento, di superfici non piane (al momento usate quasi esclusivamente nella fase di testing). Molto importante è poi evidenziare che, come approfondito meglio a pagina 103, l'attuale modello è basato esclusivamente su parametri caratteristici di una traiettoria. Ampliando quindi la fase di addestramento della rete, è possibile utilizzarla non più solo per una semplice stima, ma per una vera e propria predizione delle variabili di interesse per un qualunque veicolo (non solo pesante) cui viene assegnato quel determinato percorso (con parametri di velocità etc.), oltre che,

chiaramente, essere utilizzabile, variando *input* ed *output*, per gli ambiti più disparati. Si sottolinea, peraltro, che è stata codificata un'ulteriore rete neurale con scopo predittivo, ma questa non ha, al momento, fornito i risultati cercati o mostrato un potenziale simile a quello della precedente. Viene evidenziato, infine, che ogni aspetto principale di questa trattazione è stato implementato nell'interfaccia utente (UI) di *Simscape Vehicle Templates* (che è stato soggetto a numerose modifiche, creandone una versione personalizzata), in modo da rendere il più possibile accessibile quanto fatto ad utenti esterni, per cui sono stati previsti anche diversi commenti esplicativi.

Bibliografia

L. Güvenç H. Yu and Ü. Özgüner. Heavy duty vehicle rollover detection and active roll control. *IEE Access*, 46(6):451–470, 2008.

Enrico Galvagno Mauro Velardocchia, Alessandro Vigliani. *Appunti di Meccanica del Veicolo*. 2020.

Jon Foster Varun Bollapragada Jeff R. Crandall Mark Clauser Jason R. Kerrigan Taewung Kim, Dipan Bose. Identification of characteristics and frequent scenarios of single-vehicle rollover crashes during pre-ballistic phase; part 1 – a descriptive study. *Elsevier*, 107:31–39, 2017.

Bin Li Tianjun Zhu, Xiaoxuan Yin and Wei Ma. A reliability approach to development of rollover prediction for heavy vehicles based on svm empirical model with multiple observed variables. *IEE Access*, 8:89367–89380, 2020.

Xiaoxiang Na Tianjun Zhu, Xiaoxuan Yin and Bin Li. Research on a novel vehicle rollover risk warning algorithm based on support vector machine model. *IEE Access*, 8:108324–108334, 2020.

Konghui Guo Jiming Lu Xinjie Zhang, Yi Yang and Tao Peng. Contour line of load transfer ratio for vehicle rollover prediction. *Vehicle System Dynamics*, 55(11): 1748–1763, 2017.

Liang Hou Huosheng Hu Xiangjian Bu Xuanwei Chen, Wei Chen and Qingyuan Zhu. A novel data-driven rollover risk assessment for articulated steering vehicles using rnn. *Journal of Mechanical Science and Technology*, 34(5):2161 2170, 2020.

Abderrahmane Boubezoul Yamine Sellami, Hocine Imine and Jean-Charles Cadiou. Rollover risk prediction of heavy vehicles by reliability index and empirical modelling. *Vehicle System Dynamics*, 56(3):385–405, 2018.