

POLITECNICO DI TORINO

Laurea Magistrale in Ingegneria Meccanica

Tesi di Laurea Magistrale



**Politecnico
di Torino**

Sviluppo logiche di path planning e tracking per veicoli passeggeri a guida autonoma

Relatori:

Mauro Velardocchia

Antonio Tota

Luca Dimauro

Candidato:

Gianluca Frison

A.A. 2022/2023

*A tutti coloro
che credono in me*

Indice

Indice delle figure	7
Sommario	11
1. Introduzione	15
1.1 Veicoli autonomi	15
1.2 Metodi di path planning	19
1.2.1 Caricamento della mappa di simulazione	20
1.2.2 Definizione dei punti di partenza e di arrivo	20
1.2.3 Modelli di path planning	21
1.2.4 Simulazione e Visualizzazione dei risultati.....	30
1.2.5 Ottimizzazione del percorso.....	31
2. Progettazione Path Planning	33
2.1 Introduzione funzione.....	34
2.2 Utilizzo e funzionalità	37
2.3 Studio di sensitività	39
2.3.1 Introduzione più ostacoli (con NumSegments = 4).....	41
2.3.2 Più punti di passaggio (Waypoints)	51
2.4 Variazione funzione trajectoryOptimalFrenet.m	54
2.5 Analisi dei risultati e utilizzo	54
3. Progettazione Path tracking	63
3.1 Modello dinamico di veicolo	63
3.2 Parametri fisici del modello di veicolo impiegato	68
3.2.1 Massa totale.....	72
3.2.2 Semipassi anteriore e posteriore.....	72
3.2.3 Momento di inerzia di imbardata	73
3.2.4 Coefficienti di rigidità in deriva	74
3.2.5 Angoli di sterzo e angolo volante in funzione dell'accelerazione laterale	77
3.3 Confronto modello a bicicletta e modello di veicolo IPG-CM.....	78
3.3.1 Manovra su IPG-CM	79
3.3.2 Manovra su Matlab	82
3.3.3 Risultati e confronto	83
3.4 Sistema di controllo.....	86
3.4.1 Modello dinamico di veicolo.....	87
3.4.2 Validazione modello stati errore.....	90
3.4.3 Modello Simulink	93

3.4.4 Risultati validazione modello controller su Simulink	101
3.4.5 Calibrazione parametri ottimizzatore LQR	107
4. Simulazioni CarMaker	117
4.1 Passaggio mappa Matlab a IPG-CM formato CRG con ASAM.CRG	117
4.2 Configurazione simulazione finale su IPG CarMaker	124
4.3 Implementazione controllo sterzo.....	129
4.4 Implementazione controllo velocità.....	132
4.4.1 Controllo di velocità con riferimento off-line	132
4.4.2 Controllo di velocità con riferimento on-line	134
4.5 Risultati simulazioni su modello di veicolo CarMaker.....	137
4.5.1 Risultati controllo di velocità con riferimento off-line	138
4.5.2 Risultati controllo di velocità con riferimento on-line.....	143
4.5.3 Confronto velocità di riferimento off-line vs on-line.....	147
5. Conclusioni.....	151
Ringraziamenti	153
Bibliografia	155

Indice delle figure

Figura 1 – Processo per la generazione della traiettoria di riferimento	20
Figura 2 - Logica di controllo veicolo autonomo [10].....	22
Figura 3 – Path planner con struttura multilivello [10].....	22
Figura 4 - A* Planner Matlab [10].....	24
Figura 5 - Hybrid A* Planner Matlab [10]	24
Figura 6 – PRM: Mappa inspessita per tener conto delle dimensioni del veicolo Matlab [10].....	25
Figura 7 -PRM: Creazione nodi, segmenti e traiettoria Matlab [10]	26
Figura 8 – RRT: Inserimento randomico di un nuovo nodo/stato [10].....	26
Figura 9 – RRT: Inserimento di ulteriori nodi, anche intermedi e generazione traiettoria [10].....	26
Figura 10 – RRT*: Generazione della traiettoria più breve con RRT* [10]	27
Figura 11 - Esempio mappa con campo potenziale [18].....	28
Figura 12 - Tipi di State Space, Segmenti, State Validator in Matlab [10]	30
Figura 13 - Possibile personalizzazione degli State Space e state Validator in Matlab [10]	30
Figura 14 - Esempio utilizzo algoritmo RRT in Matlab [10]	30
Figura 15 - Cinematica massa puntiforme	33
Figura 16 – NumSegments=1 con singolo ostacolo	37
Figura 17 – NumSegments=2 con singolo ostacolo	38
Figura 18 – NumSegments=3 con singolo ostacolo	38
Figura 19 – NumSegments=4 con singolo ostacolo	39
Figura 20 – Prima prova superamento ostacolo percorso diagonale	40
Figura 21 – Seconda prova: aumento di segmenti.....	40
Figura 22 – Terza prova: allargamento traiettorie su cui ricercare	41
Figura 23 – 1° Introduzione più ostacoli.....	41
Figura 24 – Introduzione nuovo ostacolo: nessuna traiettoria trovata	42
Figura 25 – Allargamento traiettorie su cui eseguire la verifica.....	42
Figura 26 – Prova con elevato tempo di calcolo.....	43
Figura 27 – Riduzione numero traiettorie calcolate.....	43
Figura 28 – Spostamento del nuovo ostacolo	44
Figura 29 – Ingrandimento nuovo ostacolo	44
Figura 30 – Introduzione maggior numero di segmenti – nessuna traiettoria trovata	45
Figura 31 – Riduzione del numero di segmenti può portare a una soluzione migliore	45
Figura 32 – Introduzione di ostacoli random	46
Figura 33 – Ingrandimento ostacoli	47
Figura 34 – Prova di blocco di una traiettoria precedentemente validata	47
Figura 35 – Prova senza traiettorie di output.....	48
Figura 36 – Aumento accelerazione laterale limite (soft constraint)	48
Figura 37 – Aumento del tempo di simulazione	49
Figura 38 – Aggiunta ostacolo alla traiettoria di riferimento precedente.....	50
Figura 39 – Allargamento traiettorie da valutare	50
Figura 40 – Ampliamento in trasversale delle traiettorie campione	51

Figura 41 – Inserimento terzo ‘waypoint’	52
Figura 42 – Prova fallimentare nell’introduzione di un terzo ‘waypoint’	53
Figura 43 – Raggiungimento del punto finale in presenza di un terzo waypoint	53
Figura 44 – Mappa ad ostacoli per validazione del path planner	55
Figura 45 – Prima traiettoria valida trovata dal path planner	55
Figura 46 – Traiettorie valide trovate	56
Figura 47 – Traiettorie valide che arrivano al punto esatto di destinazione	57
Figura 48 - Messaggio di avviso di informazioni sulle traiettorie trovate valide	57
Figura 49- Velocità in funzione delle coordinate [X,Y]	58
Figura 50 – Tempo di percorrenza in funzione delle coordinate [X,Y].....	58
Figura 51 – Accelerazione restituita dal planner in funzione delle coordinate [X,Y].....	58
Figura 52 - Accelerazione trasversale in funzione delle coordinate [X,Y].....	59
Figura 53 - Accelerazione longitudinale in funzione delle coordinate [X,Y].....	59
Figura 54 - Differenza Acc.pianificatore - Acc.longitudinale.....	60
Figura 55 - Accelerazione longitudinale vs Tempo	61
Figura 56 - Modello a bicicletta	63
Figura 57 - GUI CarMaker con Tesla-Model-S	69
Figura 58 - Caratteristiche di default veicolo.....	69
Figura 59 - Informazioni veicolo	70
Figura 60 - Riconfigurazione nuovo modello veicolo	70
Figura 61 - Modello di veicolo reso rigido con caratteristiche	71
Figura 62 - Informazioni nuovo modello veicolo	71
Figura 63 - Sistemi di riferimento veicolo CarMaker	72
Figura 64 - Posizione assali veicolo	73
Figura 65 - Impostazione manovra Ramp Steer.....	74
Figura 66 - Istantaneo veicolo durante simulazione.....	74
Figura 67 - Istantanea veicolo reso rigido durante simulazione	75
Figura 68 – Ramp steer a velocità costante	75
Figura 69 - Accelerazione laterale manovra RAMP STEER.....	75
Figura 70 - Angolo volante manovra RAMP STEER	76
Figura 71 - Forze laterali - Angoli di scorrimento - Manovra RAMP STEER.....	76
Figura 72 - Angolo sterzo ruote - Accelerazione laterale - RAMP STEER.....	77
Figura 73 - Angolo volante - Accelerazione laterale - RAMP STEER	78
Figura 74 - Impostazione COLPO DI STERZO- 2m/s^2	79
Figura 75 - Velocità COLPO STERZO 2m/s^2	80
Figura 76 - Accelerazione laterale COLPO STERZO 2m/s^2	80
Figura 77 - Angolo volante COLPO STERZO 2m/s^2	80
Figura 78 - Angolo di sterzo COLPO DI STERZO 2m/s^2	80
Figura 79 - Impostazione COLPO DI STERZO- 7m/s^2	81
Figura 80 - Velocità COLPO DI STERZO 7m/s^2	81
Figura 81 - Accelerazione laterale COLPO DI STERZO 7m/s^2	81
Figura 82 - Angolo volante COLPO STERZO 7m/s^2	81
Figura 83 - Angolo sterzo COLPO DI STERZO 7m/s^2	82
Figura 84- Script per manovra di COLPO DI STERZO su Matlab	82
Figura 85 - Modello di veicolo con errori di posizione e imbardata.....	87
Figura 86 - Risultati validazione modello 4 stati su traiettoria rettilinea	92
Figura 87 - Risultati validazione modello 4 stati su traiettoria rettilinea inclinata	93

Figura 88 - Modello Simulink per creazione e validazione modello path tracking.....	94
Figura 89 - Subsystem 'Calcolo_riferimenti'	94
Figura 90 - Subsystem 'Modello_bicicletta'	95
Figura 91 - Guadagno ottimo LQR.....	95
Figura 92 - Subsystem 'State_errori'	96
Figura 93 - Subsystem 'Calcolo_traiettoria'	96
Figura 94 - Subsystem 'Traiettoria'	97
Figura 95 - Subsystem 'Errori_reali_locali'	98
Figura 96 - Rappresentazione significato coordinata curvilinea di riferimento	99
Figura 97 - Funzione 'Stato_vicino' per calcolo coordinata curvilinea di riferimento.....	99
Figura 98 - Modello Simulink con calcolo traiettoria di riferimento continua ma approssimata	100
Figura 99 - Subsystem 'Calcolo_riferimenti'	100
Figura 100 - Funzione approssimata per calcolo della coordinata curvilinea di riferimento	101
Figura 101 - Risultato perseguimento traiettoria rettilinea	102
Figura 102 - Stati errore perseguimento traiettoria rettilinea	102
Figura 103 - Risultato perseguimento traiettoria rettilinea inclinata	103
Figura 104 - Stati errore perseguimento traiettoria rettilinea inclinata	103
Figura 105 - Risultato perseguimento traiettoria circolare.....	104
Figura 106 - Stati errore perseguimento traiettoria circolare	105
Figura 107 - Risultato perseguimento traiettoria path planning	106
Figura 108 - Stati errore perseguimento traiettoria path planning.....	106
Figura 109 - Systema generico closed-loop	107
Figura 110 - Bilanciamento Performance - Actuation efforts	108
Figura 111 - Sistema di controllo solo feedback.....	109
Figura 112 - Oprimal Gain da LQR	111
Figura 113 - Risultato inseguimento traiettoria 1° prova	112
Figura 114 - Stati errore 1° prova	112
Figura 115 - Risultato inseguimento traiettoria 2° prova	113
Figura 116 - Stati errore 2° prova	114
Figura 117 - Risultato inseguimento traiettoria 3° prova	115
Figura 118 - Stati errore 3° prova	115
Figura 119 - Esempio mappa generata in Matlab	118
Figura 120 - Esempio utilizzo formato OpenCRG	119
Figura 121 - Creazione griglia di base scenario	120
Figura 122 - Creazione dato formato CRG.....	120
Figura 123 - Creazione ostacoli scenario	120
Figura 124 - Adattamento formati	121
Figura 125 - Salvataggio file con relativo nome	121
Figura 126 - Interfaccia Scenario Editor.....	121
Figura 127 - Direzione asse stradale	122
Figura 128 - Scelta tracciato di simulazione	122
Figura 129 - Profilo stradale riportato nello Scenario Editor	123
Figura 130 - Object List dello Scenario Editor.....	123
Figura 131 - Allargamento carreggiata.....	123
Figura 132 - Scenario definitivo	124

Figura 133 - Rappresentazione dello scenario di prova	124
Figura 134 - Interfaccia modello Simulink-CarMaker	125
Figura 135 - Interfaccia Simulink CarMaker for Simulink	125
Figura 136 - GUI CarMaker for Simulink	126
Figura 137 - Deselezionare Active Route	126
Figura 138 - Condizioni iniziali manovra su CarMaker	127
Figura 139- - Impostazioni Driver dinamica longitudinale e laterale	127
Figura 140 - Vehicle Data Set CarMaker	128
Figura 141 - Steering model	128
Figura 142 - Interfaccia generale Powertrain CarMaker	129
Figura 143 - VehicleControl Subsystem	129
Figura 144 - Controllo angolo volante e derivate	130
Figura 145 – Simulink generazione angolo volante di controllo	130
Figura 146 - Calcolo segno errore posizione laterale locale	131
Figura 147 - Guadagno correttivi matrice dei guadagni ottimi LQR	132
Figura 148 - Controllo proporzionale acceleratore e freno – off-line di velocità	133
Figura 149 - Calcolo velocità longitudinale di riferimento	133
Figura 150 - Nuovo modello controllo off-line/on-line di velocità	134
Figura 151 – Calcolo off-line e on-line velocità	135
Figura 152 - Calcolo on-line velocità di riferimento	135
Figura 153 - Risultato persecuzione traiettoria di riferimento su CarMaker	138
Figura 154 - Errori di posizione laterale e di imbardata manovra CarMaker	139
Figura 155 - Angolo volante manovra CarMaker	139
Figura 156 - Controllo velocità longitudinale CarMaker off-line	140
Figura 157 - Risposte dinamiche controllo velocità off-line 1	141
Figura 158 - Risposte dinamiche controllo velocità off-line 2	142
Figura 159 - Risposte dinamiche controllo velocità off-line 3	142
Figura 160 - Risultato persecuzione traiettoria di riferimento su CarMaker	143
Figura 161 - Errori di posizione laterale e di imbardata manovra CarMaker	144
Figura 162 - Angolo volante manovra CarMaker	144
Figura 163 - Controllo velocità longitudinale CarMaker on-line	145
Figura 164 - Risposte dinamiche controllo velocità on-line 1	146
Figura 165 - Accelerazioni tangenziali e centripete con controllo on-line	146
Figura 166 - Risposte dinamiche controllo velocità on-line 2	147
Figura 167 - Risposte dinamiche controllo velocità on-line 3	147
Figura 168 - Confronto velocità longitudinale di riferimento off-line vs on-line	148
Figura 169 - Confronto velocità di riferimento on-line con coefficienti aggiunti	148
Figura 170 - Confronto velocità di riferimento on-line e velocità eseguita dal veicolo di simulazione	149

Sommario

L'incremento costante dell'interesse, da parte dell'uomo, per la tecnologia, spinge gli animi degli appassionati a una ricerca sempre più intensa e incessante verso forme di integrazione uomo-macchina via via più profonde. Una tra le tecnologie ancora in fase di sviluppo e ottimizzazione sono i veicoli a guida autonoma. Come negare l'inevitabile impatto positivo sull'abbassamento dello stress per i guidatori e sulla riduzione degli incidenti stradali, la diminuzione degli sprechi causati da una guida imprudente e poco frugale, il calo dei consumi di carburante e delle emissioni nell'ambiente di inquinanti. Tutto questo dovrebbe bastare per giustificare il largo interesse del mondo ingegneristico, e non solo, a tale tema di ricerca.

Il presente lavoro di tesi, si pone l'obiettivo di passare in rassegna le fasi di ricerca, di studio, di utilizzo e manipolazione di logiche di base per la generazione e il perseguimento di traiettorie per veicoli a guida autonoma. A seguito di una prima fase di ricerca, documentazione e catalogazione generale dei sistemi a guida autonoma, si passa allo studio di alcune logiche per la generazione della traiettoria ('path planning') che rappresentano il cuore fondamentale di un sistema autonomamente decisionale, mettendo in evidenza le differenze e le somiglianze che esistono tra di esse. Ci si concentra, dunque, sullo studio e ottimizzazione di un metodo 'Curves Motion Planning' in ambiente Matlab denominato 'trajectory Optimal Frenet', basato sulla generazione di traiettorie curvilinee cinematicamente e dinamicamente eseguibili da un veicolo reale, andando ad agire su alcune funzioni di costo che garantiscano l'ottenimento di traiettorie conformi alle richieste di progetto (e.g. riduzione distanza percorsa, limitazione accelerazioni laterali). L'obiettivo originario dell'ottimizzazione è quello di rendere un modello di path planning fruibile in un ambiente off-road, in cui gli ostacoli non sono necessariamente corpi fisici perfettamente definiti, come avviene in un ambiente strutturato quale la città, ma entità da evitare completamente o parzialmente, quali buche o avvallamenti. La soluzione, qui proposta, risiede nella modellazione degli ostacoli più o meno voluminosi in funzione dell'entità di un determinato ostacolo naturale. Si passa a una seconda fase di creazione e validazione in ambiente Matlab di un modello dinamico lineare di veicolo: single-track, che possa rappresentare, entro certi limiti, il modello a più gradi di libertà scelto su IPG-CarMaker che verrà successivamente usato nella fase di validazione dell'intero algoritmo proposto. Si esegue un upgrade del single-track model ottenendo un modello a quattro stati (invece che due) per la rappresentazione degli errori di posizione e orientamento del veicolo rispetto alla traiettoria di riferimento. Anche in questo caso il modello viene validato rispetto al veicolo di simulazione di CarMaker, per poter essere usato nello sviluppo di un controllo basato su un ottimizzatore LQR (in Matlab e Simulink), atto all'inseguimento della traiettoria (path tracking) tramite la riduzione dei suddetti errori di posizione e di orientamento. Si tiene, ovviamente, conto dei limiti (in parte già valutati nel path planning) dinamici caratterizzanti il mezzo scelto come campione. Tale logica di controllo viene utilizzata per permettere l'esecuzione automatica della traiettoria in ambiente CarMaker, importando il blocco di controllo dell'angolo volante nell'interfaccia 'CarMaker4Simulink' in cui è possibile modificare la logica di controllo Simulink che sta sotto il governo dei veicoli IPG-CarMaker. Parallelamente viene importato la mappa (contenente gli opportuni ostacoli), su cui eseguire la simulazione, da Matlab a CarMaker

tramite l'utilizzo della libreria ASAM OpenCRG reperibile online. Infine, al termine delle simulazioni, vengono riportate le risposte dinamiche del veicolo come prova della validità dell'intero algoritmo di path planning e path tracking qui presentato.

Abstract

The constant increase of human interest in technology drives enthusiasts to a progressively deeper and more restless quest for forms of human-machine integration. New technologies are spreading, many of which are still in the development phase, including driving automation systems. These can undeniably bring positive impact in terms of reduction of the stress levels for drivers as well as road accidents decrease, reduction of wastes caused by incautious and fuel-inefficient driving, leading to decreased fuel consumption and pollutant emissions into the environment.

The aim of this thesis work is to review, study, and implement basic algorithms for generating and pursuing desirable trajectories in autonomous vehicles. After a preliminary literature review about autonomous driving systems, the study has been focused on trajectory generation logics ('path planning'), which represent the fundamental core of an autonomously decision-making system, highlighting the differences and similarities among them. The activity is then shifted on studying and optimizing a 'Curves Motion planning' method implemented in Matlab, called 'trajectory Optimal Frenet'. It is based on the generation of curvilinear trajectories that are kinematically and dynamically executable by a real vehicle, acting on certain cost functions that ensure the generation of trajectories in line with projects requirements (e.g., reducing distance covered, limiting lateral accelerations). The ideal optimization goal is to adapt a path planning model for an off-road environment, where obstacles are not necessarily well-defined physical bodies as it is in a structured environment such as a city, but entities to be completely or partially avoided such as potholes or bumps. The proposed solution lies in modelling the size of obstacles based on the dimensions of specific natural obstacles. Then a second phase is followed with the creation and validation of a linear dynamic model of a single-track vehicle in Matlab. This model represents, within certain limits, the higher multi-degree of freedom model selected in IPG-CarMaker for the validation of the entire proposed algorithm. An upgrade is performed on the single-track model to obtain a four-state model (rather than two) for representing position and orientation errors of the vehicle with respect to the reference trajectory. This model is also validated against the CarMaker vehicle to be used in the development of control strategy based on a LQR optimizer (in Matlab and Simulink) for trajectory tracking purposes. This control logic is used to enable the automatic execution of the trajectory with a co-simulation between Simulink and CarMaker environments, by modifying the vehicle steering system. The path-planning based map (with the appropriate obstacles) is imported from Matlab to CarMaker, by using the online available ASAM OpenCRG[®] library. Finally, the dynamic response of the vehicle is presented and critically analyzed, as proof of the validity of the entire algorithm.

1.Introduzione

1.1 Veicoli autonomi

I veicoli a guida autonoma, di cui si parla in moltissimi testi tra cui [1] [2], sono dei sistemi complessi in cui varie componenti hardware e software interagiscono al fine di percepire ciò che è presente nell'ambiente circostante tramite sensoristica, decidere quali azioni adottare per mezzo di software e/o dell'intelligenza artificiale (IA) ed effettuare le manovre tramite opportuni attuatori. Tutte queste funzioni, eseguite da un veicolo autonomo, possono essere suddivise in una gerarchia generale di livelli:

- Livello Strategico (Percezione): è l'insieme delle azioni legate all'acquisizione di informazioni sull'ambiente circostante, ovvero l'identificazione di ostacoli fissi e dinamici, pedoni, linee e segnali stradali, conformazione del terreno e molto altro. Questa fase avviene grazie all'utilizzo di sensori quali GPS, IMU (Inertia Measurement Unit), Camera, LiDar, Radar, sistemi V2X....
- Livello Tattico (Pianificazione): è l'insieme delle operazioni volte alla determinazione degli input di riferimento da inviare ai sistemi di attuazione del veicolo. Fanno parte di questa categoria i sistemi di path planning per la determinazione della traiettoria e della velocità di riferimento da mantenere durante il percorso.
- Livello di Controllo (Controllo): è l'insieme dei sistemi (elettronici e meccanici) atti all'esecuzione delle manovre di riferimento che il livello tattico predetermina. Fanno parte di questa categoria sia gli algoritmi per il calcolo delle azioni di controllo da eseguire per perseguire le traiettorie desiderate (i.e. path tracking), sia gli attuatori che eseguono fisicamente le varie azioni.

Questa è la struttura 'multy-layer' classica per il controllo di veicoli più o meno autonomi. Essa si basa sulla suddivisione dei vari processi strategici, tattici e di controllo e l'identificazione degli attori hardware e software che le compongono e determinano. Uno schema riassuntivo è il seguente:

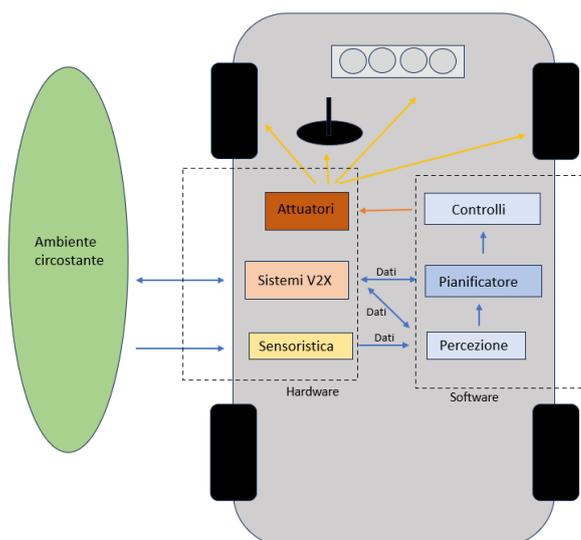


Figure 1 - Architettura classica 'multi-layer' per veicoli automatizzati

Oltre a questa configurazione classica del controllo di un veicolo, esistono delle architetture dette 'End-to-end' basate sull'IA (Neural-Network e Deep Learning/Machine Learning) che permettono di insegnare al veicolo, tramite varie prove sperimentali ('trial and error'), quali azioni di controllo compiere a seguito di specifici segnali che i sensori acquisiscono [3]. Questo argomento non verrà trattato all'interno del presente lavoro, poiché prevede lo sviluppo di tematiche di ricerca attuale, congeniali ad uno studio successivo più avanzato.

Tornando alla struttura multi-layer, basata sulla distinzione dei vari attori che concorrono al funzionamento di un veicolo autonomo 'classico', fanno parte della 'Sensoristica' una svariata quantità di sensori, di cui i principali e più conosciuti sono:

- GPS: 'Global Positioning System' è un sistema, ben noto ormai, di posizionamento e navigazione basato sull'utilizzo di satelliti.
- IMU: 'Inertial Measurement Unit' è un componente elettronico che serve per la misurazione delle forze agenti su un corpo dotato di inerzia, quindi delle accelerazioni angolari nelle tre direzioni principali nello spazio. Al suo interno contiene degli accelerometri, dei giroscopi e alcune volte dei magnetometri.
- Camera: corrisponde a una classica telecamera che prende immagini dall'ambiente circostante che poi verranno processate attraverso opportuni software di decodifica ed elaborazione.
- LiDar: 'Light Detection and Ranging' è un Sistema di scansione che utilizza la luce laser (solitamente nelle frequenze comprese tra l'ultravioletto e l'infrarosso) per acquisire informazioni di distanza (quindi di volume) sull'ambiente circostante (ostacoli, pedoni, altri mezzi...). Esso emette tramite una sorgente un fascio laser che rimbalza sugli oggetti da scansionare e viene assorbito da un ricevitore posto in corrispondenza dell'emettitore. In base al tempo che intercorre tra l'emissione e il riassorbimento della luce si riesce a calcolare la distanza da un determinato oggetto/superficie. Il LiDar genera una nuvola di punti di colore diverso in funzione della distanza che poi viene elaborata digitalmente per ricostruire l'immagine dell'ambiente esterno.

- Radar: ‘Radio Detection and Ranging’, è il parallelo del LiDar, che al posto di sfruttare un raggio laser come mezzo per il rilevamento delle superfici, utilizza onde elettromagnetiche (sulle frequenze di onde radio o microonde) per il rilevamento della posizione, velocità di oggetti fissi o mobili.

I sistemi di comunicazione di informazioni tra il veicolo e le altre entità dell’ambiente circostante che possano interferire con esso passano sotto la sigla generale V2X (Vehicle-to-everything) e si suddividono nelle seguenti categorie:

- V2I: Veicolo verso l’infrastruttura (segnaletica stradale, edifici, ...)
- V2N: Veicolo verso la rete (interconnessione basata sulle reti internet cellulare, migliorata a seguito dell’introduzione del 5G)
- V2V: Veicolo verso Veicolo (con veicoli che possiedono questo sistema di comunicazione)
- V2P: Veicolo verso Pedone (sistemi di avvertimento pedoni e altri utenti deboli della strada quali ciclisti)
- V2D: Veicolo verso dispositivo elettronico (comunicazione con ogni strumento elettronico che possa interfacciarsi con il veicolo)
- V2G: Veicolo verso Griglia (sistema di connessione tra veicoli elettrici e la rete di distribuzione elettrica)

Tutti questi sistemi di comunicazione sfruttano la rete WLAN (ovvero senza fili) per mettere in contatto il veicolo e gli altri interlocutori.

Le componenti software per l’elaborazione (Percezione, Pianificazione e Controllo) dei dati acquisiti tramite la sensoristica e i sistemi V2X prevedono l’utilizzo di software appositi che lavorino in tempo reale, al fine di processare i dati in ingresso e calcolare le opportune risposte in uscita da far eseguire agli Attuatori. Lo sviluppo di questi algoritmi avviene per mezzo di software quali Matlab, Simulink, Python e moltissimi altri strumenti di programmazione e validazione quali IPG-CarMaker, CarSim, che verranno utilizzati nel proseguo, quindi non oltre descritti in questa sezione.

I sistemi di attuazione principali in un veicolo, qualunque sia il suo livello di automazione, sono:

- Angolo di sterzo
- Pedale impianto frenante
- Pedale motore (termico o/e elettrico in base alla soluzione adottata)
- Sistema di trasmissione e cambio marcia (manuale o automatico)
- Luci di svolta o di frenata e ulteriore segnaletica

Maggiore è la connessione tra questi dispositivi e più il veicolo sarà in grado di eseguire le manovre desiderate con precisione e sicurezza. Ad esempio, un sistema TV (‘Torque Vectoring’) può aumentare la ‘capacità sterzante’ di un dato veicolo, aiutando concretamente l’azione dell’impianto sterzante.

Classificazione livelli guida autonoma

I veicoli a guida autonoma si differenziano per il livello di autonomia/automazione che posseggono. Esistono sistemi più o meno evoluti che demandano in varia misura le decisioni e le azioni che si devono prendere durante le differenti condizioni di guida. La

normativa SAE J3016_202104 [4] (ultimo aggiornamento del 2021) ha regolamentato e definito quali siano i livelli di automazione che un dato veicolo può assumere. Essa, dunque, suddivide i veicoli autonomi in sei categorie basate sul livello di autonomia con cui il veicolo è in grado di eseguire le DDT ('Dynamic Driving Task'):

Livello 0: 'No Driving Automation': Il conducente è sempre al comando di tutte le DDT; quindi, non è presente un sistema di controllo che possa governare alcuna azione del veicolo, se non un AEB ('Autonomous Emergency Braking') che attiva la frenata in caso di imminente collisione e sistemi di allarme anticollisione che emettono un segnale acustico.

Livello 1: 'Driver Assistance': Anche in questo caso il conducente deve sempre avere il controllo del veicolo, ma il sistema di controllo può modificare la velocità e la sterzata del mezzo. Fanno parte di questi sistemi il 'Lane departure warning' che genera un avviso sonoro o visivo qualora il guidatore stia cambiando di corsia senza aver attivato la freccia di svolta, oppure il 'Lane keeping' di prima generazione che agisce sullo sterzo per mantenere il veicolo sulla corsia di marcia quando tende a spostarsi al di fuori senza che gli venga richiesto. Si può occupare della dinamica longitudinale e laterale esclusivamente in modo separato.

Livello 2: 'Partial Driving Automation': Il conducente continua ad avere un ruolo fondamentale nel controllo del veicolo, ma può demandare alcune operazioni, anche adattative, al sistema di controllo. Fanno parte di questo livello l'ACC ('Adaptive Cruise Control') e il Lane Keeping di ultima generazione, in grado di adattare la velocità del veicolo in funzione dell'andatura dei mezzi che lo precedono, cercando di mantenere le distanze di sicurezza opportune finanche all'arresto completo. In questa fase si riesce ad integrare il controllo laterale del veicolo parallelamente al longitudinale; pertanto, il veicolo deve essere in grado di agire autonomamente sullo sterzo mentre agisce sulla 'Driveline' e sul sistema di frenata. Questo sistema si disattiva appena il guidatore decide di riprendere il comando.

Livello 3: 'Conditional Driving Automation': Si tratta di veicoli in grado di eseguire autonomamente un viaggio in autostrada, quindi cambiando di corsia per superare, a parte in casi di difficile decisione in cui il driver deve essere in grado di riprendere all'istante la guida. Questo sistema non è ancora in grado di manovrare in contesti cittadini o di ingressi/uscite da autostrade. Tali modelli di veicoli sono stati abbandonati perché il conducente era troppo distratto quando il veicolo richiedeva il suo intervento non essendo stato ben delineato il confine tra una manovra che deve essere eseguita autonomamente e quale per mezzo del guidatore. Il livello 3 prevede già, in alcuni momenti, una completa autonomia del veicolo, ma il guidatore deve rimanere pronto a riprendere il comando in caso di necessità.

Livello 4: 'High Driving Automation': In questo caso il veicolo è in grado di compiere un intero viaggio autonomamente, anche in condizioni più difficili a livello decisionale. Il conducente del veicolo diventa un vero e proprio passeggero. Tuttavia, questi veicoli hanno ancora dei limiti legati alle zone in cui possono essere sperimentati e alla velocità massima che possono raggiungere. Rimangono ancora delle situazioni residuali (molto complesse) in cui il guidatore deve intervenire per prendere possesso del mezzo. Alcuni

sistemi di illuminazione e segnalazione interni all'abitacolo vengono utilizzate per comunicare al passeggero le manovre che verranno eseguite.

Livello 5: 'Full Driving Automation': Il veicolo è in grado di compiere qualsiasi percorso autonomamente in qualsiasi condizione di guida; pertanto, nessun sistema di comando è presente all'interno dell'abitacolo (no sterzo, pedali o altri simili). Questi veicoli sono in grado di viaggiare in qualsiasi area geografica a qualsiasi velocità in sicurezza, anche grazie a sistemi di interconnessione tra V2X precedentemente citati.



Figure 2 - Livelli di automazione veicoli SAE J3016_202104

Nella seguente figura tratta dalla normativa SAE j3016_202104 si fa una distinzione (cromatica) tra i livelli 0-2 in cui si parla di 'assistenza alla guida' o 'sistema di supporto alla guida' che richiedono un costante controllo del conducente, e i livelli 3-5 designabili come 'sistemi di guida automatizzata' che sono in grado, in determinate situazioni più o meno diffuse, di guidare senza intervento del conducente (al limite supervisione nel livello 3) [5].

1.2 Metodi di path planning

Come è stato descritto in precedenza le componenti che caratterizzano un veicolo autonomo sono molte. Nel presente lavoro si ha l'obiettivo di studiare, sfruttare o sviluppare, e ottimizzare delle logiche di path planning e path tracking, di base, che vanno sotto la categoria (in figura 1) 'Software', Pianificazione e Controllo. Andando ad analizzare nel dettaglio ciò che è presente in letteratura si evince come il materiale riguardante il path tracking sia più vasto e diffuso rispetto ai metodi di path planning;

pertanto, si è deciso di concentrarsi maggiormente, almeno in una prima fase di lavoro, sui metodi di pianificazione della traiettoria a livello locale, ovvero quegli algoritmi che a fronte di uno scenario contenete determinati ostacoli (mobili e fissi) è in grado di generare una traiettoria valida, priva di collisioni e ottimizzarla secondo i requisiti di percorrenza imposti dal conducente (minore distanza percorsa, minor tempo di attraversamento, minor accelerazione laterale/curvatura ...).

Il processo globale/generale che porta alla definizione di una opportuna traiettoria, per un qualsiasi tipo di mezzo in movimento, prevede l'utilizzo di una serie di macro-passaggi riassunti nel seguente schema, preso dalla fonte [6]:

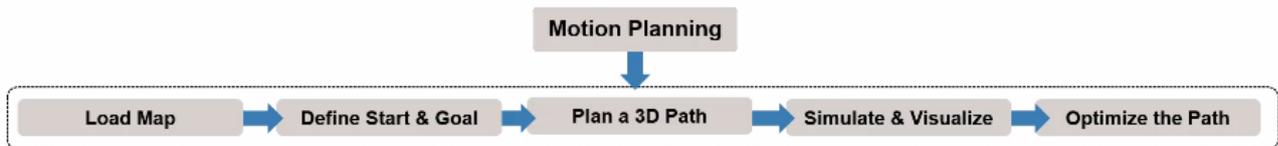


Figura 1 – Processo per la generazione della traiettoria di riferimento

1.2.1 Caricamento della mappa di simulazione

La prima fase ('Load map') richiede la generazione di uno scenario in cui testare o utilizzare il modello di path planning. Tale mappa/ambiente/scenario può essere ottenuta in svariati modi sul software Matlab:

- Ci sono mappe già presenti nella libreria Matlab o in alcuni suoi toolbox scaricabili dal sito della MATHWORKS®.
- Si possono creare scenari a partire da una serie di elementi base (rette, curve, rotonde, incroci, veicoli, ostacoli, ...) e messi assieme tramite appositi toolbox Matlab (RoadRunner-Automated Driving Toolbox).
- Possono essere importate delle mappe da altri software di simulazione quali IPG Carmaker o CarSim che permettono di creare, proprio come in RoadRunner, degli scenari descrittivi dell'ambiente in cui il planner deve operare.
- Possono essere estratti degli scenari direttamente dal mondo reale tramite una serie di sensori (quali Lidar, Infrarossi, Radar, ...) che acquisiscono una nuvola di punti, che rappresentano gli oggetti e le superfici circostanti, per poi essere processati in modo da essere leggibili al programma di pianificazione.

Questa fase di acquisizione della mappa in cui operare fa parte dei compiti che il livello strategico (percettivo) deve eseguire. Non verrà ulteriormente descritta per non uscire dal tema del presente lavoro.

1.2.2 Definizione dei punti di partenza e di arrivo

La seconda fase ('Define Start and Goal') richiede l'inserimento dei punti di partenza e di arrivo della traiettoria che si vuole ottenere nella fase di pianificazione locale della traiettoria. Questi punti vengono inseriti o dall'utente che vuole testare l'algoritmo di programmazione o a partire dai punti di passaggio principali che vengono direttamente estratti dal percorso che un dato veicolo deve seguire (per esempio da un viaggio impostato su Google Maps).

Oltre ai punti di partenza e di arrivo si devono impostare una serie di parametri che devono essere rispettati nella ricerca di una traiettoria valida. Questi parametri sono solitamente legati a vincoli cinematici o dinamici che il veicolo non può superare, oppure sono dei pesi/costi specifici per la penalizzazione di elevati tempi di percorrenza o maggiori distanze percorse.

Altri dati da impostare sono legati al processo di discretizzazione e al tempo di campionamento che si vogliono avere nel processo di path planning. L'orizzonte temporale di programmazione di queste traiettorie locali è dell'ordine dei 10 secondi circa e di una distanza tra lo 'start point' e l'end point' di 10-100 metri. Ecco il perché del termine 'pianificatore locale' e non globale, come Google Maps.

In questo lavoro il punto di partenza e di arrivo sono stati impostati dal sottoscritto, come verrà descritto al capitolo 2.

1.2.3 Modelli di path planning

La terza fase ('Plan a 3D Path') è quella che interessa maggiormente in questo lavoro di tesi, e consiste nello studio e ottimizzazione di un codice per la generazione di traiettorie a livello locale (capitolo 2). Data una serie di ostacoli presenti nell'ambiente di simulazione, i punti di passaggio globali e i vincoli, il codice deve essere in grado di generare almeno una traiettoria valida, priva di collisioni, tra il punto di partenza e il punto di arrivo.

Esistono vari tipi di 'Path Planner'. Una prima distinzione può essere fatta tra quelli che lavorano in uno spazio 2D e quelli in 3D. I pianificatori 2D sono stati sviluppati principalmente per robot o veicoli da movimento su asfalto [7], mentre i planner 3D sono stati utilizzati per applicazioni aeree di UAV-droni [8] oppure per veicoli off-road (UGV) [9] che necessitano di un numero maggiore di gradi di libertà e uno studio più approfondito delle condizioni operative in cui il mezzo deve operare (i.e. terrameccanica). L'obiettivo del presente lavoro è indirizzato all'utilizzo di planner 3D per veicoli movimento terra (UGV), anche se il planner utilizzato è stato sviluppato sul piano 2D. La proposta è quella di considerare avvallamenti e buche come delle entità da evitare completamente o parzialmente; quindi, rappresentate come zone di collisione più o meno grandi su cui negare completamente il passaggio.

Il processo di pianificazione di traiettorie può essere inserito all'interno dello schema seguente, che rispecchia i tre livelli principali per il controllo di un veicolo autonomo:

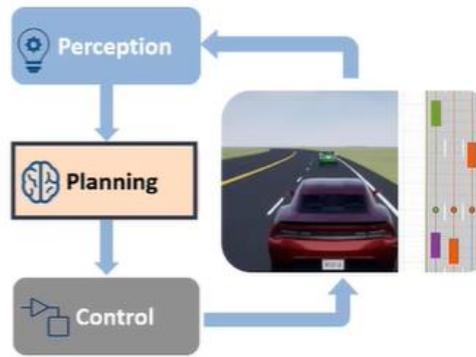


Figura 2 - Logica di controllo veicolo autonomo [10]

Si trovano già disponibili online una serie di pianificatori locali implementati su codici Matlab: ‘Motion Planning Algorithms’ [10] presenti nel ‘**Navigation Toolbox**’ scaricabile dal sito della MathWorks®. Tutti questi algoritmi si basano sulla generazione di una serie di traiettorie o segmenti di cui viene successivamente verificata la non presenza di collisioni con gli ostacoli presenti nella mappa e il rispetto dei vincoli cinematici/dinamici preimpostati. Ovviamente questi algoritmi definiscono la successione degli ‘Stati’ che il veicolo dovrà percorrere lungo la traiettoria. Nella figura che segue si fa riferimento ad una struttura complessa a più livelli di path planning in cui il ‘Global Planning’ è il vero pianificatore del percorso $[X_{ref}, Y_{ref}, \psi_{ref}]$ da perseguire. In realtà esso fa parte dei planner locali rispetto a quelli globali quali Google Maps, ma in questo schema si identificano anche quelle sotto-operazioni di controllo del carattere del veicolo (‘Behavior Planning’) in cui si desidera pianificare, non la traiettoria vera e propria, ma altri fattori quali il mantenimento di corsia tramite il ‘Lane Keeping’ o il rispetto della velocità di riferimento adattativa tramite un ACC (‘Adaptive Cruise Control’). Queste operazioni possono essere ugualmente catalogate sotto la categoria ‘Controlli’, come viene fatto nel presente lavoro. L’ultima categoria detta ‘Local Re-planning’ o ‘Trajectory Optimization’ rappresenta la ricalibrazione della traiettoria, definita tramite i ‘Path planning algorithms’, in fase di esecuzione, per garantire un miglior processo decisionale adattato alle condizioni che il veicolo si trova ad affrontare istante per istante. Anche in questo caso esso si può catalogare come una fase di controllo invece che di pianificazione.

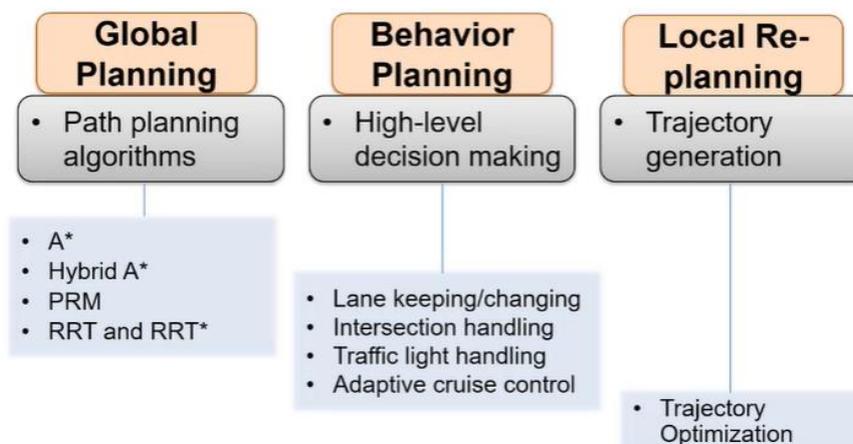


Figura 3 – Path planner con struttura multilivello [10]

I modelli veri e propri di Path Planning, in quanto generatori di Traiettorie di riferimento $[X_{ref}, Y_{ref}, \psi_{ref}]$ si possono classificare secondo quanto indicato in [2] e [11]:

Occupancy Grid/Lattice Planning

Anche detti ‘Grid based Algorithms’ o ‘Graph search based planners’ sono stati utilizzati inizialmente per la programmazione di robot che non dovessero tenere conto della segnaletica stradale e dei limiti dinamici che un veicolo possiede.

Questi modelli di path planning sono basati sulla suddivisione della mappa di simulazione in una griglia o in un reticolo. Per passare dal punto di partenza A a quello di arrivo B si deve generare una traiettoria che attraversi una serie di stati intermedi, verificando la non presenza di altre entità (pedoni, veicoli, infrastrutture) che vengono eliminate dal gruppo di possibili punti di passaggio. Non è sempre garantito l’ottenimento di una traiettoria valida o ottimizzata. Maggiore sarà la finezza della griglia/meshatura e migliore sarà la rappresentazione degli ostacoli all’interno di essa, ma condurrà a un aumento dei tempi di calcolo computazionale. Esistono, dunque, varie logiche più o meno evolute in funzione dell’ottimizzazione che raggiungono nella definizione della traiettoria stessa:

Dijkstra: è il metodo di pianificazione Grid based inventato per primo e si basa sulla ricerca della traiettoria più breve su una mappa a griglia o a reticolo [12]. Ogni ostacolo viene indicato come una cella o un nodo occupato attraverso cui la traiettoria da pianificare non può passare. Questo metodo, se una soluzione esiste la trova, esplorando tutte le possibili traiettorie percorribili, portando, però, a un incremento considerevole dei tempi di calcolo.

A*: Come il metodo precedente suddivide la mappa in una griglia in cui le celle che contengono degli ostacoli vengono segnate come occupate. Pianifica la traiettoria più corta priva di collisioni attraverso questa mappa senza esplorare tutti i nodi che ne fanno parte tramite funzioni euristiche in esso implementate; quindi, presenta una riduzione considerevole dei tempi di generazione del percorso rispetto a Dijkstra. Anche in questo caso, se esiste la soluzione, essa viene trovata.

Si possono imporre dei costi/pesi relativi ai nodi che vengono attraversati (non considera vincoli cinematici o dinamici).

È appropriato per robot olo-nomici dove non è essenziale il controllo dei parametri cinematici e dinamici del mezzo.

Questo metodo, come Dijkstra lavora in una mappa bidimensionale (salvo ulteriori elaborazioni dell’algoritmo).

Gli stati che vengono specificati nella definizione della traiettoria sono $[X_{ref}, Y_{ref}]$.

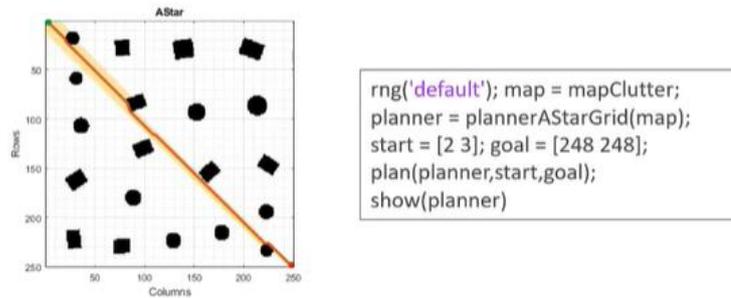


Figura 4 - A* Planner Matlab [10]

Hybrid A*: è una variante dell'A* più idoneo al rispetto dei vincoli cinematici (e parzialmente dinamici) legati ai limiti propri di un veicolo reale (vincoli non-olonomici). Si basa sui cosiddetti 'state lattices', ovvero delle traiettorie curvilinee che permettono una connessione graduale tra i vari nodi/stati del sistema [13]. Permette, quindi, una generazione di traiettorie prive di punti angolosi molto accentuati che si presentavano, invece, nei modelli base Dijkstra e A*.

Utilizza ancora una suddivisione della mappa in una griglia oppure in un reticolo e produce una serie di primitive dallo stato in cui si trova allo stato successivo, poi sceglie la primitiva migliore tra quelle generate.

Ovviamente evita gli ostacoli come nei casi precedenti e lavora sempre in uno spazio 2D.

In questo caso gli stati definiti sono (x,y,θ) , quindi tiene conto anche dell'angolo di imbardata.

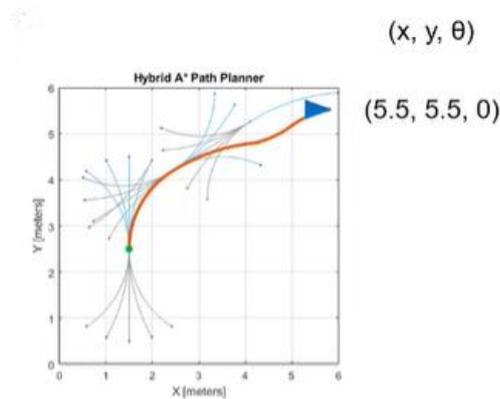


Figura 5 - Hybrid A* Planner Matlab [10]

Con l'Hybrid A* si può impostare la lunghezza delle primitive, il minimo raggio di curvatura delle traiettorie, il moto in avanti e all'indietro e altri parametri di interesse.

I metodi a griglia diventano computazionalmente onerosi quando le mappe diventano di dimensione maggiore e quando i gradi di libertà aumentano.

Esistono altri algoritmi di pianificazione della traiettoria che fanno parte di questa categoria basati sulla griglia o sul reticolo, in cui il funzionamento di base rimane sempre circa lo stesso. Alcuni di essi sono dei metodi di ripianificazione di ottimizzazione dinamica per tenere conto della variazione dello scenario in cui si deve operare istante per istante: dynamic A* (D*), Field D*, Theta*, Anytime Repairing A* (ARA*), Anytime D*

(AD*). Altri modelli misti uniscono i metodi grid based con altri, quali i ‘Potential Field (presentati in seguito) per la rappresentazione degli ostacoli al fine di generare un tracciato ottimale anche dal punto di vista del confort e della sicurezza [14].

Random Sampling Algorithms

Essi vengono anche chiamati ‘Sampling Based Algorithm’ o ‘Sampling based planners’ perché si basano sulla generazione di nodi/punti, in modo randomico all’interno dello spazio degli stati/ configuration space, attraverso cui la traiettoria di riferimento potrà passare; quindi, solo alcuni di questi verranno attraversati dal tracciato di riferimento finale a seconda che essi risultino sovrapposti ad ostacoli. Il vantaggio principale di tali algoritmi, rispetto ai precedenti, è l’elevata rapidità con cui un percorso valido viene generato anche all’interno di spazio a più dimensioni; infatti, essi vengono utilizzati nella pianificazione del percorso di droni/ UAV [6] e dei movimenti di bracci robotici, in tempo reale. Un’altra peculiarità è la loro capacità di considerare vincoli non-olonomici (cinematici/dinamici) nella ricerca della traiettoria migliore. Il difetto principale risiede nella scarsa ottimizzazione del percorso trovato dal momento in cui il metodo di ricerca da essi attuato non è del tutto esaustivo di tutti i possibili stati esistenti all’interno della mappa.

I principali modelli esistenti sono i seguenti:

PRM: ‘Probabilistic RoadMap’: esso si basa sulla generazione di nodi in punti della mappa in cui non sono presenti ostacoli. Questi punti vengono poi connessi tutti tra loro, come si vede in figura 7, attraverso delle linee. Solo quelle prive di collisione e a minore distanza tra due nodi vengono scelte per il tracciato di riferimento [15].

A livello di codice Matlab, esso prende la mappa con gli ostacoli e li rende più ingombranti per tenere conto dell’ingombro del veicolo nelle fasi di manovra:

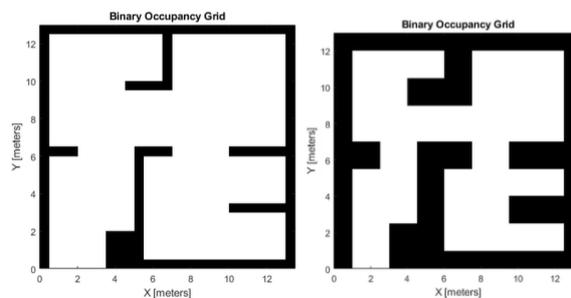


Figura 6 – PRM: Mappa inspessita per tener conto delle dimensioni del veicolo Matlab [10]

Vengono, quindi, inseriti in modo random dei nodi nella mappa e vengono collegati ognuno agli altri tramite una serie di segmenti che non vadano a urtare gli ostacoli.

Si può definire la lunghezza dei segmenti e quanti nodi inserire. Alla fine, vengono scelti i segmenti migliori nel perseguimento della traiettoria tramite altri algoritmi quale l’A*.

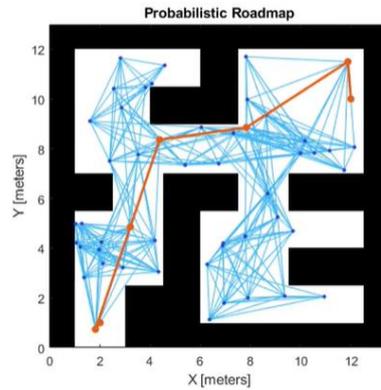


Figura 7 -PRM: Creazione nodi, segmenti e traiettoria Matlab [10]

Questo metodo di path planning è valido solo per robot oloomici; quindi, non considera vincoli cinematici/dinamici caratteristici di un veicolo reale. Va bene quando gli ostacoli della mappa sono noti a priori e statici.

RRT: ‘Rapidly-exploring Random Tree Algorithm’: è sicuramente il modello di path planning più studiato tra tutti i random sampling algorithms perché può tenere conto di vincoli non-olomici del veicolo (massima curvatura, momento imbarante, ...) lavorando in real-time.

Genera randomicamente degli stati/nodi ‘ad albero’ all’interno della mappa di simulazione in zone prive di ostacoli per poi collegarli tramite il segmento più breve che collega due nodi adiacenti.

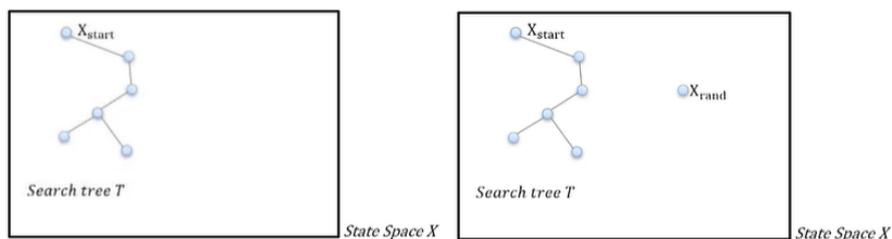


Figura 8 – RRT: Inserimento randomico di un nuovo nodo/stato [10]

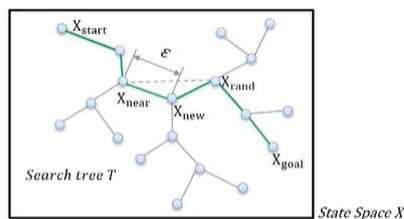


Figura 9 – RRT: Inserimento di ulteriori nodi, anche intermedi e generazione traiettoria [10]

Si possono generare primitive che connettano i nodi con grado del polinomio elevato al fine di ottenere delle curve traiettorie regolari e prive di punti angolosi che porterebbero a forti accelerazioni/oscillazioni del mezzo nella fase di esecuzione.

Il suo modello è stato ottimizzato in moltissimi lavori di ricerca per poterlo rendere fruibile ed efficace in scenari reali. Un modello ottimizzato, ad esempio, viene integrato in

[16], al fine di avere un planner che sia in grado di generare traiettorie che tengano conto di funzioni di ottimizzazione specifiche.

Questo algoritmo restituisce una traiettoria valida ma non necessariamente la più breve.

RRT*: è una variante ottimizzata dell'RRT. Espande l'albero nello stesso modo dell'RRT, ma cerca di collegare due nodi il più vicino possibile attraverso la generazione di un cerchio/ una bolla attorno al nodo da cui deve avanzare l'albero.

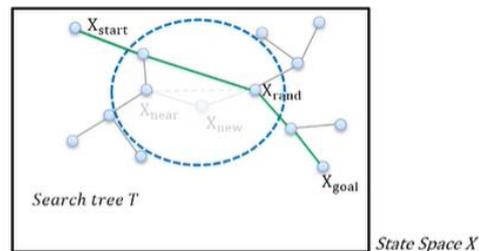


Figura 10 – RRT*: Generazione della traiettoria più breve con RRT* [10]

Quindi, genera meno nodi, alleggerendo il costo computazionale quando i gradi di libertà aumentano e quando la dimensione della mappa cresce, ma i limiti legati ai punti angolosi rimangono. Solo una sovrapposizione di metodi per la generazione di spline curvilinee può garantire percorsi gradualmente e cinematicamente fattibili.

Potential fields methods

Essi sono anche chiamati 'Artificial Potential Field' poiché si basano sulla generazione di un campo potenziale variabile all'interno dello scenario di simulazione. Come si sa dalle leggi della fisica gravitazionale o elettromagnetica, un dato corpo lasciato libero di muoversi all'interno di un campo tenderà a muoversi da zone a più alto verso zone a più basso potenziale. I modelli potential field si basano su questo concetto, pertanto in corrispondenza di ostacoli o di segnaletica stradale vengono assegnati dei livelli alti di potenziale, mentre dove la strada è libera il potenziale sarà il più possibile vicino allo zero. In questo modo si svilupperanno delle forze repulsive nei confronti degli ostacoli, attrattive verso le zone libere e disponibili ad essere attraversate. Questi modelli di path planning sono di largo interesse in ambito di ricerca robotica, veicolistica e aerea, tuttavia presentano alcuni difetti: Nel momento in cui viene generata la traiettoria che segua le zone a più basso potenziale questo pianificatore non tiene conto dei limiti dinamici del veicolo. Esistono delle soluzioni di path planner multilivello che permettono un'ottimizzazione del semplice potential field tramite l'integrazione di controlli ottimi che considerino la dinamica del veicolo. L'altro limite fondamentale legato a questi approcci risiede nel fatto che si possono generare delle mappe potenziali in cui il veicolo raggiunge un punto a basso campo, compreso tra due potenziali molto elevati (relativi ad ostacoli da evitare completamente) e non è più in grado di avanzare verso il punto di minimo che rappresenta l'arrivo desiderato. Anche questo problema è aggirabile agendo sulle funzioni potenziali descrittive dei vari ostacoli.

Molte riferimenti sviluppano delle varianti nella modellazione degli APF (Artificial Potential Field). In [17] viene utilizzato un modello APF che genera funzioni diverse, sia nella forma che nell'entità, dei potenziali in base al tipo di ostacolo considerato: esso suddivide gli ostacoli in attraversabili, non attraversabili e corsie stradali. Gli ostacoli non

attraversabili genereranno un campo più elevato, quindi repulsivo, rispetto agli altri due. A questo algoritmo viene associato un modello predittivo di path planning e path tracking MPC che tenga conto dei vincoli dinamici del veicolo. Un altro modello interessante per la rappresentazione dei campi potenziali si trova in [18] dove si tiene in considerazione la velocità del proprio veicolo e la velocità relativa rispetto agli ostacoli circostanti, al fine di costruire dei potenziali più elevati quando le suddette velocità incrementano. Vengono proposte cinque diverse funzioni potenziali: ‘Target Potential’ posto pari a zero in prossimità del punto di arrivo e ad un valore massimo nel punto di partenza, generando una forza attrattiva del veicolo verso il punto di arrivo. Un ‘Road Potential’ viene caratterizzato al fine di avere una forza repulsiva da parte delle linee di delimitazione della carreggiata per garantire il mantenimento in strada del veicolo. Un ‘Lane Potential’ serve per mantenere nella propria corsia il veicolo, a meno di eventuali sorpassi o per evitare ostacoli. Esso, infatti è molto ridotto per non interferire con ostacoli da evitare assolutamente. Un ‘Vehicle Potential’ viene generato per tenere il proprio veicolo distante dagli altri. Esso viene espresso tramite una funzione di distribuzione gaussiana bidimensionale al fine di generare un campo elevato di ampiezza maggiore in direzione longitudinale di marcia e più stretto sul laterale del veicolo. Si tiene conto anche delle velocità del mezzo e della differenza di velocità rispetto agli ostacoli. Per finire un ‘Velocity Potential’ viene implementato per far inseguire un veicolo di fronte nel caso in cui esso si muova a una velocità superiore all’ego vehicle. Esso è un potenziale negativo, quindi attrattivo, come rappresentazione di un punto di target locale che il veicolo deve inseguire. Al termine del calcolo di queste componenti, i vari potenziali vengono sovrapposti ottenendone uno globale, che permetterà al veicolo di inseguire una traiettoria di riferimento. La rapidità con cui vengono calcolati questi potenziali, rende questi metodi di path planning altamente sfruttabili in tempo reale. Altri testi parlano di APF a livello meno dettagliato facendo una semplice distinzione tra potenziali attrattivi e repulsivi e cercando di sorpassare il problema del minimo locale (Improved Potential Field [19]). Un modello adattativo delle condizioni operative e il relativo sistema strategico, tattico e di controllo viene presentato in [20]. Infine, in [21] viene mostrato un ‘Prediction Artificial Potential Field’ che pone dei potenziali virtuali repulsivi in punti rischiosi di minimo locale in cui il veicolo/robot (AGV) potrebbe rimanere intrappolato senza proseguire oltre. Vengono posti istante dopo istante nuovi punti target da perseguire. I risultati sperimentali (su robot) mostrano una diminuzione del consumo energetico nell’esecuzione della traiettoria, minori oscillazioni di movimenti, minor tempo e distanza percorsa per il raggiungimento della destinazione.

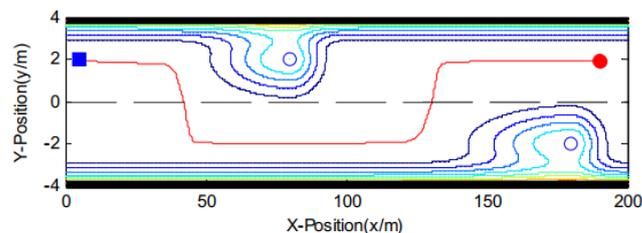


Figura 11 - Esempio mappa con campo potenziale [18]

Curves Motion Planning

Essi sono modelli di path planning che consistono nell'interpolazione di una serie di punti di passaggio attraverso cui passano delle traiettorie campione. Sono spesso stati utilizzati come complemento a modelli Random tree o Occupancy grid al fine di rendere le traiettorie più gradualmente tramite l'utilizzo di polinomi o curvilinee. Quando vengono utilizzate da sole si possono impostare delle funzioni di costo che il pianificatore cercherà di soppesare e rispettare (massima curvatura, accelerazione laterale, velocità, ...) nella scelta del tracciato finale. Le traiettorie che incontrano ostacoli lungo il loro cammino vengono disattivate perché non percorribili.

I tipi di curve utilizzate nella generazione dei possibili percorsi sono: linee rette, circonferenze, clotoidi, curve polinomiali, curve di Bézier, curve Spline.

Non verrà approfondito questo modello di PP poiché ne verrà implementato uno nella fase di progettazione al capitolo 2 (trajectory_Optimal_Frenet) che ha come base un modello di Occupancy grid.

Model Predictive Control

Il MPC è stato utilizzato in moltissimi gruppi di ricerca per la sua capacità di prevedere quale sarà il comportamento del veicolo nei prossimi istanti di tempo e di generare l'azione di controllo sul veicolo a fronte di vincoli imposti dall'utente (velocità, accelerazione, curvatura massima, ...). È anche computazionalmente poco oneroso dal momento che si basa su un modello di veicolo a massa puntiforme (presentato all'inizio del capitolo 2). Per questi motivi è stato molto utilizzato come path tracking (controllo), ma si può implementare anche singolarmente al fine di svolgere sia la funzione di PP che di PT insieme. Infine, è stato applicato anche in modelli di PP all'interno di logiche multilivello in cui acquisisce il ruolo di pianificatore di basso livello accoppiato ad 'high level PP' di varia natura (ad esempio in [22] viene utilizzato nell'alto livello un 'resistance network' basato sugli APF già discussi). Un altro esempio molto valido dell'accoppiamento tra APF e MPC viene mostrato in uno scenario ad alta velocità (validato attraverso simulazione su IPG-CarMaker e Simulink) in manovre di cambio corsia, sorpasso e inseguimento di un veicolo antistante [23]. Infine, un modello non-lineare di pneumatico viene accoppiato a un MPC, usato come PT control, al fine di generare gli angoli volante e le accelerazioni laterali necessarie al inseguimento della traiettoria di riferimento [24].

È consigliato implementare, a livello di controllo, un modello 'single-track' o 'double-track' per tenere conto della dinamica del veicolo reale, e tenere come PP locale il MPC. Ovviamente ogni soluzione che si decida percorrere va opportunamente calibrata per adattare al meglio gli strumenti che vengono utilizzati.

State Space

Tutti questi metodi di pianificazione, alcuni dei quali implementati in Matlab, hanno bisogno di uno State Space a cui fare riferimento. Esso è una serie di variabili (es. posizione, accelerazione, velocità, curvatura, angolo di imbardata, ...) che si vogliono tenere sotto controllo durante le manovre che il veicolo (considerato ancora come una massa puntiforme nei PP) compirà nel inseguimento della traiettoria. Esistono State Space già definiti nelle librerie Matlab, oppure se ne possono creare di personalizzati. Ogni modello di SS ha un maggior interesse nei confronti di specifiche variabili che il

programmatore ha intenzione di tenere sotto controllo nella fase di lancio delle simulazioni PP.

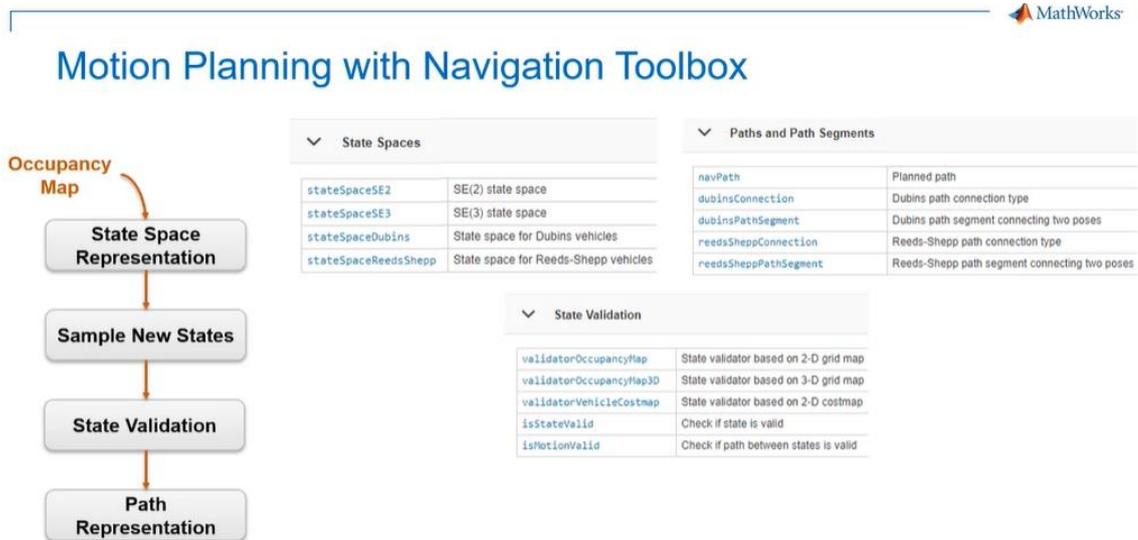


Figura 12 - Tipi di State Space, Segmenti, State Validator in Matlab [10]

Gli 'State Validator' verificano che non ci siano collisioni lungo la traiettoria generata.

The screenshot shows the 'Custom Path Planning Interfaces' section of the MATLAB interface, containing a table with the following entries:

createPlanningTemplate	Create sample implementation for path planning interface
nav.StateSpace	Create state space for path planning
nav.StateValidator	Create state validator for path planning

Figura 13 - Possibile personalizzazione degli State Space e state Validator in Matlab [10]

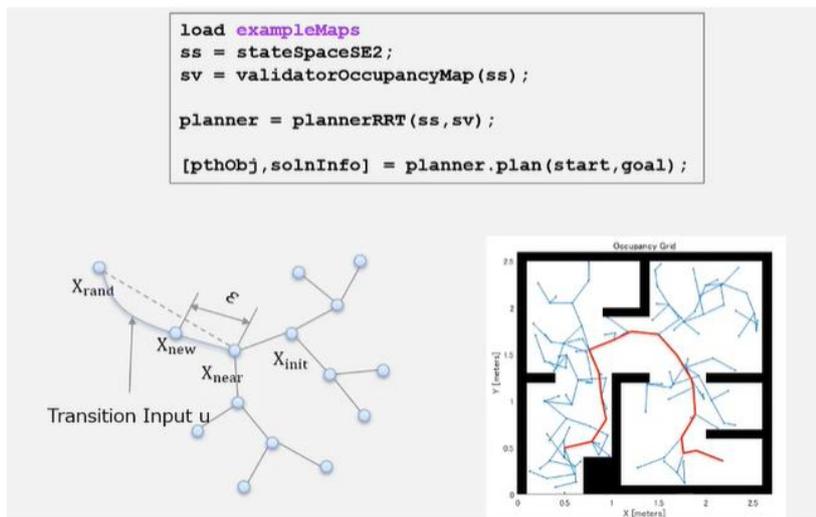


Figura 14 - Esempio utilizzo algoritmo RRT in Matlab [10]

1.2.4 Simulazione e Visualizzazione dei risultati

La quarta parte ('Simulate and Visualize') riguarda il lancio del codice che genera la traiettoria di riferimento, compito per cui è stato programmato; mentre la fase di

visualizzazione dei risultati serve, ovviamente, alla valutazione delle capacità e dei limiti del codice implementato/utilizzato. Successive analisi possono essere condotte per ulteriori ottimizzazioni dell'algoritmo appena sviluppato, ad esempio tramite l'incrocio di più modelli per ottenerne uno più avanzato.

1.2.5 Ottimizzazione del percorso

La quinta fase ('Optimize the path') prevede tutte quelle azioni che si possono fare per migliorare il codice esistente e per sfruttare al massimo le sue potenzialità. Si possono prevedere delle variazioni al codice sorgente, per esempio, per ottenere la traiettoria più breve, quella con minor tempo di percorrenza oppure a minor curvatura massima raggiunta. Oppure si può imporre una velocità di spostamento lungo traiettoria personalizzata, per esempio, per evitare di avere accelerazioni trasversali eccessive. Queste ottimizzazioni possono essere eseguite direttamente nella fase di pianificazione oppure direttamente all'interno della logica di controllo, andando a monitorare il comportamento dinamico del veicolo e degli ostacoli che lo circondano.

Le migliorie che si possono fare sono innumerevoli, e l'obiettivo del presente lavoro riguarda proprio questo. Lo studio e l'ottimizzazione di algoritmi di questo genere (oltre a un successivo di sviluppo di una logica di controllo).

In generale i metodi di PP sono per lo più necessari alla generazione della traiettoria di riferimento $[X_{ref}, Y_{ref}, \psi_{ref}]$, mentre gli altri task possono essere demandati alla fase di più basso livello che è il controllo.

2. Progettazione Path Planning

Tra tutti i metodi di pianificazione di una traiettoria è stato scelto un ‘Curves Motion Planning’: ‘trajectoryOptimalFrenet’ sviluppato dalla MathWorks® che si può trovare al riferimento [25]. Dopo lo studio di alcune pubblicazioni in merito ai ‘Potential fields method’, e presa coscienza dei limiti che questi presentavano, si è deciso di sviluppare tale metodo appartenente alle ‘Curves Motion Planning’ sovrapposto alle ‘Occupancy Grid Planning’, che permette di tenere conto di alcuni limiti cinematici. Anche questo pianificatore, come quelli a campi potenziale, in alcuni casi può non trovare alcun tracciato di riferimento, ma se ricalibrato opportunamente una soluzione viene ricavata.

Le equazioni di base che un modello di path planning utilizza per la stima delle accelerazioni massime sviluppate sono di tipo cinematico. Solitamente questo viene fatto perché dover utilizzare equazioni dinamiche proprie di un veicolo reale e applicarle in ogni traiettoria di cui si fa la valutazione rallenterebbe eccessivamente la ricerca della soluzione, non rendendo applicabili questi algoritmi in real time.

Le equazioni cinematiche di base relative a un modello puntiforme di veicolo che si muove lungo la traiettoria di riferimento:

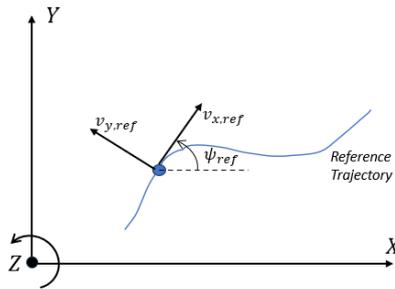


Figura 15 - Cinematica massa puntiforme

$$\dot{v}_{x,ref} = \frac{dv_{x,ref}}{dt} = a_{x,ref} \quad (2.1)$$

$$\dot{v}_{y,ref} = \frac{dv_{y,ref}}{dt} = a_{y,ref} \quad (2.2)$$

$$\dot{\psi}_{ref} = \frac{a_{y,ref}}{v_{x,ref}} \quad (2.3)$$

$$\dot{X}_{ref} = v_{x,ref} \cos \psi_{ref} - v_{y,ref} \sin \psi_{ref} \quad (2.4)$$

$$\dot{Y}_{ref} = v_{x,ref} \sin \psi_{ref} + v_{y,ref} \cos \psi_{ref} \quad (2.5)$$

L'utilizzo del pedice ‘ref’ è stato inserito per indicare che le variabili cinematiche si riferiscono alla traiettoria di riferimento, che andrà perseguita dal veicolo tramite le logiche di path tracking presentate al capitolo 3.

2.1 Introduzione funzione

Il ‘trajectoryOptimalFrenet’ è un metodo di pianificazione della traiettoria, implementato in Matlab, che genera delle traiettorie locali da collegamento dei cosiddetti waypoints che compongono un viaggio intero da un punto di partenza a uno di arrivo. Esso è, dunque, un pianificatore locale della traiettoria in quanto deve garantire lo spostamento di un mezzo da un punto A ad uno B all’interno di un percorso più ampio preimpostato tramite altre logiche (quali Google Maps). Con tale algoritmo, si può identificare la traiettoria più breve e priva di collisioni tra un gruppo numeroso di percorsi che il codice stesso genera in automatico. Il pianificatore identifica degli stati/punti intermedi basati sul percorso principale e su una serie di parametri a scelta del programmatore. Il pianificatore connette gli stati con delle traiettorie curvilinee di polinomi del 4° o 5° ordine. Vengono valutati la fattibilità cinematica (massima accelerazione laterale, massima curvatura), le eventuali collisioni e il costo di ogni singolo percorso. Tale costo deriva da una somma di termini differenti, che verranno descritti in seguito.

Sintassi

[trajectoryOptimalFrenet\(refPath,validator\)](#)

[planner = trajectoryOptimalFrenet\(.,Name,Value\)](#)

Descrizione

‘trajectoryOptimalFrenet’ crea un oggetto che prende in input la traiettoria di riferimento ‘refPath’ nella forma di una matrice nx2 di [x y] che rappresentano le n coordinate dei punti di passaggio della traiettoria globale (si consiglia di impostare solo i punti di partenza e di arrivo della traiettoria, quindi, n=2). ‘validator’ è un oggetto che contiene lo spazio degli stati e altre informazioni:

```
validator = validatorOccupancyMap;
```

valida gli stati e i movimenti che vengono eseguiti all’interno della mappa. Gli spazi occupati da ostacoli nella mappa sono considerati non attraversabili. Crea una mappa di validazione rispetto a una mappa 2D di stati associata SE(2).

Proprietà

Le proprietà che si possono assegnare all’oggetto trajectoryOptimalFrenet (e che questo deve rispettare nella scelta finale della traiettoria di riferimento) sotto forma di “planner.proprietà = ...;” sono le seguenti:

- 1) **Weights**: al suo interno si definiscono i costi delle traiettorie relativi a determinati fattori:

- 1.1) Weights.Time: default = 0; la funzione di costo moltiplica il tempo complessivo di percorrenza della traiettoria per il costo specifico per unità di tempo scelto.
- 1.2) Weights.ArcLength: default = 0; la funzione di costo moltiplica la distanza percorsa nella traiettoria per il suo costo per unità di distanza.
- 1.3) Weights.LateralSmoothness: default = 0; è un peso legato al jerk laterale. La funzione di costo moltiplica il suo peso specifico per l'integrale del quadrato del jerk laterale (derivata nel tempo dell'accelerazione laterale) su tutta la traiettoria.

Il jerk o 'strappo' in italiano si calcola come:

$$j_y = \frac{da_y}{dt} \left[\frac{m}{s^3} \right] \quad (2.6)$$

La stessa definizione si applica al jerk longitudinale.

- 1.4) Weights.LongitudinalSmoothness: default = 0; è un peso legato al jerk longitudinale. La funzione di costo moltiplica il suo peso per l'integrale del quadrato del jerk longitudinale (derivata nel tempo dell'accelerazione longitudinale) su tutta la traiettoria.
 - 1.5) Weights.Deviation: default = 1; la funzione di costo moltiplica questo peso per la distanza laterale al punto di arrivo (B) della traiettoria rispetto al punto effettivamente desiderato.
- 2) **FeasibilityParameters**: è una struttura che contiene parametri di fattibilità sotto forma scalare. Sono delle condizioni che devono essere verificate per poter considerare valida una traiettoria specifica:
 - 2.1) FeasibilityParameters.MaxCurvature: default = 0.1; è la massima curvatura che il veicolo può effettuare. $[m^{-1}]$
 - 2.2) FeasibilityParameters.MaxAcceleration: default = 2.5; accelerazione massima che il veicolo puntiforme può sviluppare nella direzione del moto. $[m/s^2]$
 - 3) **TimeResolution**: default = 0.1 [s]; è il tempo di discretizzazione per determinare gli stati della traiettoria di riferimento che verrà poi scelta. Questi stati discretizzati devono essere validati e soggetti alla funzione di costo.
 - 4) **CostFunction**: è la funzione di costo definita dall'utente. La funzione di costo deve accettare matrici degli stati $n \times 7$ per ogni traiettoria, e restituisce i costi per ogni traiettoria trovata. Questa funzione può anche essere omessa, implementandola successivamente tramite una post-ottimizzazione, che in effetti verrà eseguita.
 - 5) **TrajectoryList**: è una struttura a sola lettura che elenca tutte le traiettorie possibili con tutti i parametri che le caratterizzano:
 - 5.1) TrajectoryList.Trajectory: è una matrice $n \times 7$ $[x, y, theta, kappa, speed, acceleration, time]$, dove n è il numero di punti della traiettoria stimati (in base alla risoluzione temporale scelta).
 - 5.2) TrajectoryList.Cost: costo complessivo delle varie traiettorie
 - 5.3) TrajectoryList.MaxAcceleration: massima accelerazione verificatasi nelle singole traiettorie
 - 5.4) TrajectoryList.MaxCurvature: massima curvatura rilevata durante la traiettoria

- 5.5) **TrajectoryList.Feasible:** contiene righe [*velocity, acceleration, curvature, collision*] di validità rispetto a tutte le traiettorie calcolate. Per ognuna di quelle variabili si assegna un valore pari a 1, 0, -1. 1 significa che quel parametro rientra nei vincoli preimpostati, 0 che quel parametro non è valido, -1 che non è stato proprio valutato il rispetto del vincolo.
- 6) **TerminalStates:** struttura che contiene gli stati di arrivo rispetto alla traiettoria di riferimento. Questo incide sul calcolo di traiettorie alternative tra un punto di partenza e un punto di arrivo.
- 6.1) **TerminalStates.Longitudinal:** lunghezza delle traiettorie dal punto di partenza a quello di arrivo. Default = 30:15:90;
- 6.2) **TerminalStates.Lateral:** vettore di deviazione in direzione trasversale nei punti di arrivo. Default = -2:1:2;
- 6.3) **TerminalStates.Speed:** velocità all'arrivo in direzione del moto (può anche essere un vettore di possibili valori. Default = 10;
- 6.4) **TerminalStates.Acceleration:** accelerazione nel punto di arrivo nella direzione del moto. Default = 0;
- 6.5) **TerminalStates.Time:** vettore o scalare di tempi ammessi per eseguire tutta la traiettoria. Default = 7 [s];
- 7) **WayPoints:** matrice nx2 che contiene I punti di passaggio nella traiettoria generale [x y]. n è il numero di punti di passaggio. I punti di passaggio sono fondamentali per il modo in cui la traiettoria verrà calcolata. Si consiglia di impostare solo un punto di partenza e uno di arrivo.
- 8) **NumSegments:** numero di segmenti longitudinali per ogni traiettoria. Questa proprietà genera degli stati intermedi tra quelli terminali (denominati A e B). Questi generano primitive che tengano conto di più punti intermedi. Maggiore è il numero di segmenti, maggiore sarà la complessità delle curve ottenute e più alto il costo computazionale che il pianificatore deve sopportare. Default = 1;
- 9) **DeviationOffset:** deviazione dalla traiettoria di riferimento in direzione laterale. Un valore negativo indica uno spostamento verso destra durante una manovra che eviti un ostacolo che si trova nella traiettoria di riferimento. Default = 0; Se voglio che il punto di arrivo sia esattamente quello desiderato lascio il suo valore pari a 0.

Funzioni utilizzate

cart2frenet: converte stati cartesiani in stati di Frenet (usata per convertire gli stati dei punti iniziale da coordinate cartesiane in quelle di Frenet: coordinate curvilinee)

copy: crea copie approfondite degli object

frenet2cart: converte stati di Frenet in stati cartesiani

plan: pianifica la traiettoria ottimale

show: funzione per visualizzare la traiettoria

Limiti del metodo:

- Auto-intersezioni nella traiettoria possono dare risultati non sensati
- Il pianificatore non supporta movimenti all'indietro
- L'orientazione iniziale deve essere compresa tra $-\pi/2$ e $\pi/2$, ma il consiglio è di ridurli il più possibile
- Limitare i TerminalStates in applicazioni in tempo reale, perché un loro aumento comporta tempi di calcolo elevato

2.2 Utilizzo e funzionalità

A seguito della comprensione, scrittura e rimaneggiamento del codice relativo alla funzione appena presentata, preso dal sito della MathWorks[®], sono state eseguite una serie di simulazioni per conoscere le sue potenzialità nell'ottenimento di una traiettoria priva di collisioni e che rispetti i vincoli imposti dall'utente. Il codice Matlab utilizzato si trova all'interno di un'apposita cartella denominata 'TrajectoryOptimalFrenet_my.m'.

Traiettoria orizzontale. Un solo ostacolo. Variazione NumSegments da 1 a 4

È sconsigliato utilizzare 5 segmenti (in questi esempi iniziali) perché non aggiungeva alcun risultato significativo rispetto ai casi esaminati, nonostante il suo tempo di calcolo risulti molto maggiore.

NumSegments=1; 50 traiettorie stimate

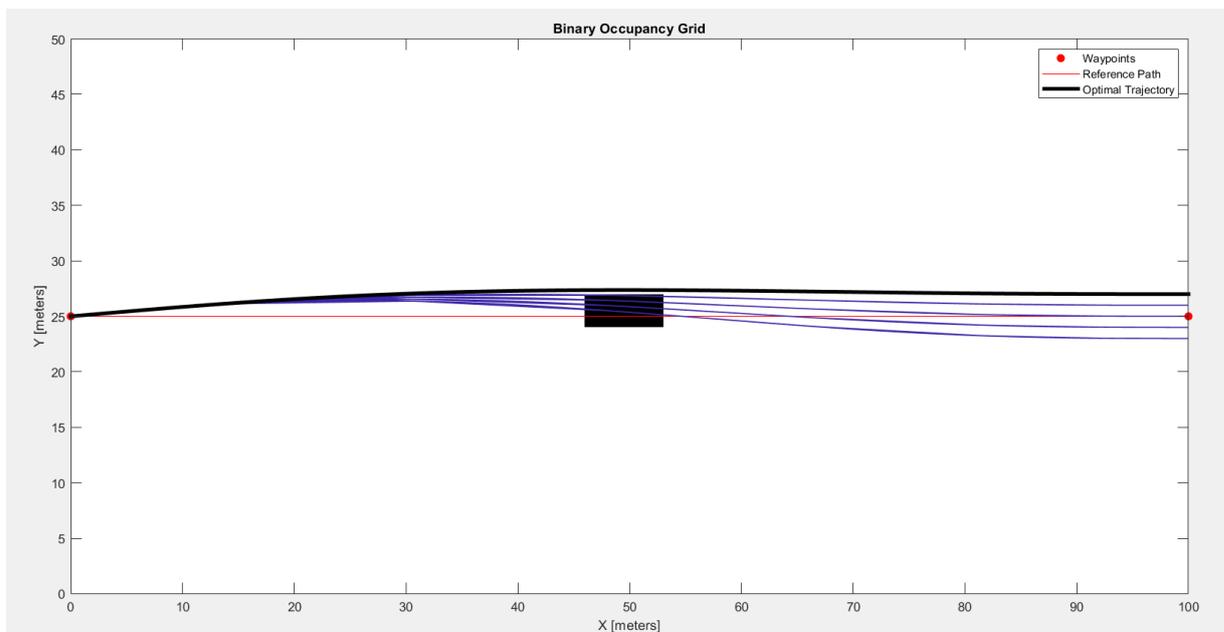


Figura 16 – NumSegments=1 con singolo ostacolo

NumSegments=2; 250 traiettorie stimate

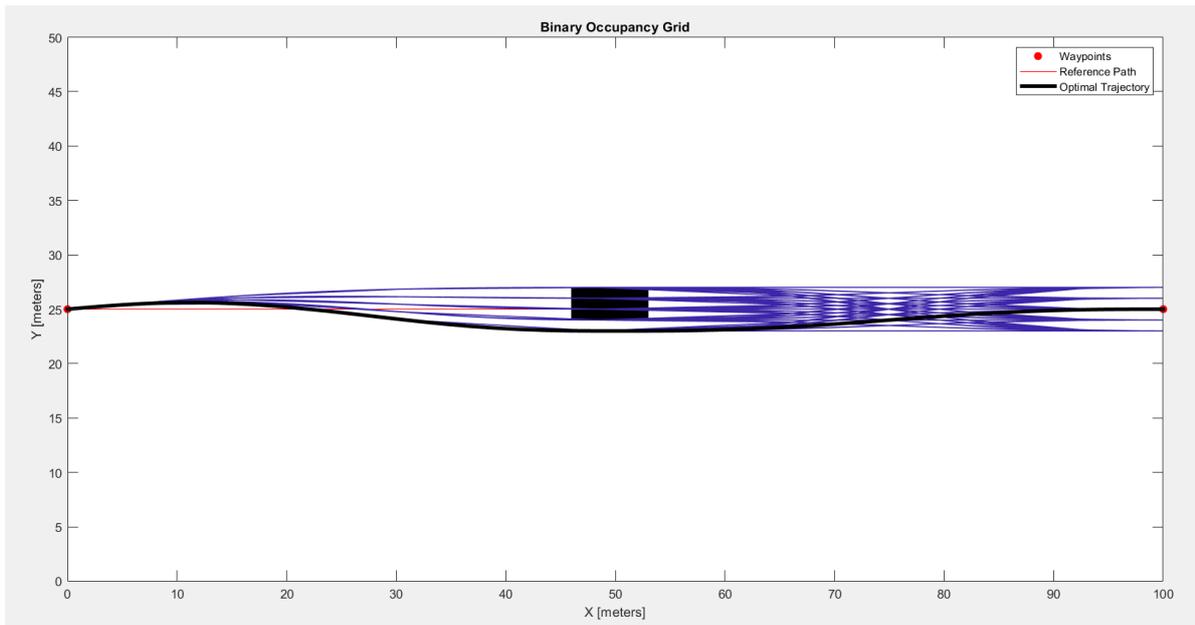


Figura 17 – NumSegments=2 con singolo ostacolo

NumSegments=3; 1250 traiettorie stimate

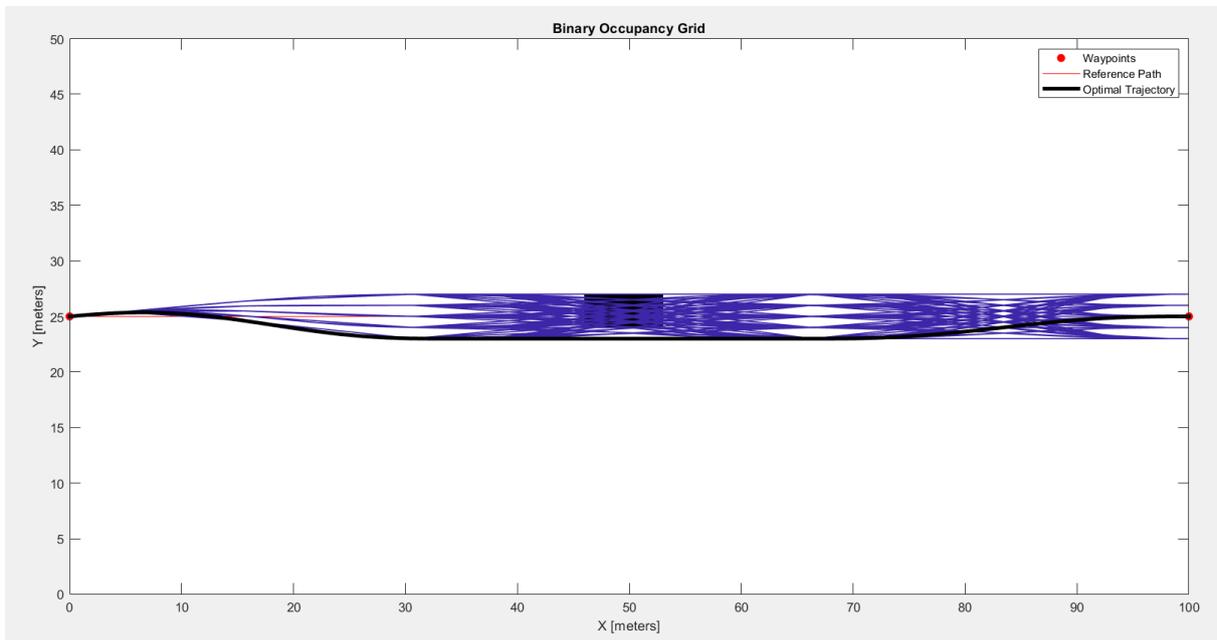


Figura 18 – NumSegments=3 con singolo ostacolo

NumSegments=4; 6250 traiettorie stimate

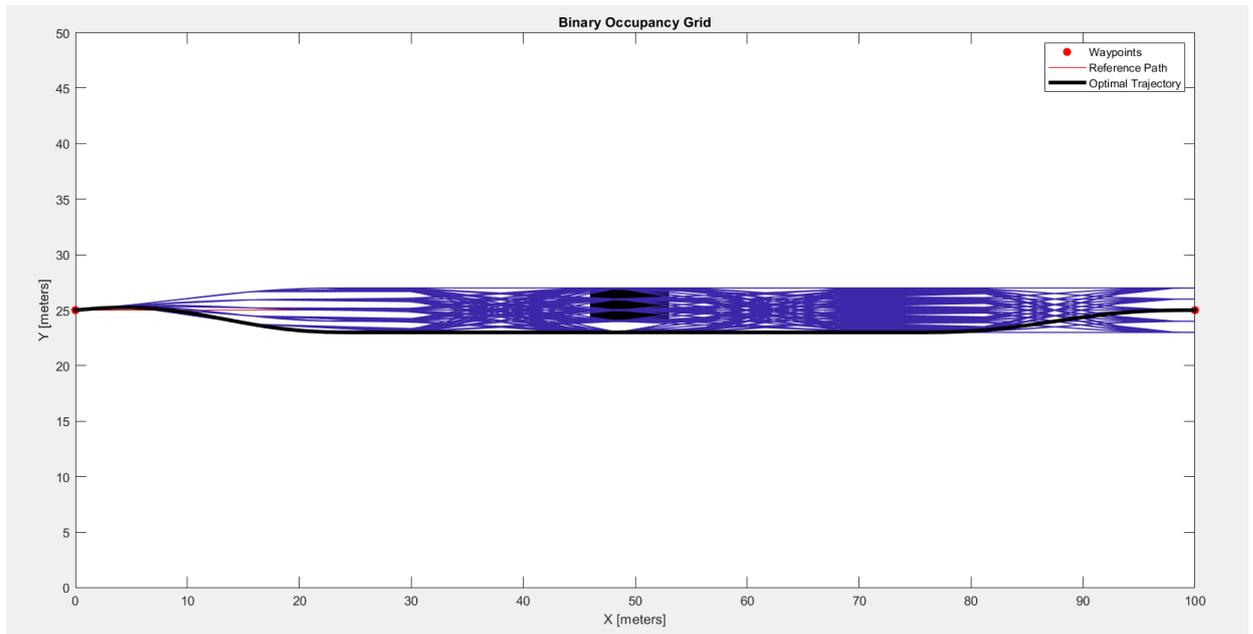


Figura 19 – NumSegments=4 con singolo ostacolo

Essenzialmente si nota un aumento delle traiettorie campione esaminate. I tempi di calcolo crescono di conseguenza, ma la traiettoria ottenuta risulta essere sempre migliore. È bene scegliere, in base alla complessità dello scenario da superare (quantità e posizione ostacoli) la quantità di segmenti ottimale per trovare un percorso valido ma senza sovraccaricare troppo il pianificatore.

Da qui in avanti, salvo diversa indicazione, il valore di NumSegments sarà impostato a 3.

2.3 Studio di sensitività

La traiettoria globale che viene presa da campione in questo studio di sensitività è quella che attraversa la diagonale dello scenario a pianta rettangolare già presentato.

Più precisamente $\text{refPath} = [0,1;100,50] = [x_{in},y_{in};x_{fin},y_{fin}]$;

Queste sono le coordinate dei punti di partenza e di arrivo (A e B)

I parametri imposti in questa prima prova sono:

<code>planner.FeasibilityParameters.MaxCurvature</code>	10
<code>planner.FeasibilityParameters.MaxAcceleration</code>	10
<code>planner.TerminalStates.Lateral</code>	-2:1:2
<code>planner.TerminalStates.Time</code>	[1:1:10]
<code>planner.NumSegments</code>	3
<code>planner.DeviationOffset</code>	0
<code>initCartState</code>	[0 1 pi/8 0 0 0]

Il cui risultato, all'interno di una mappa contenente l'ostacolo visibile in seguito al lancio della simulazione, è:

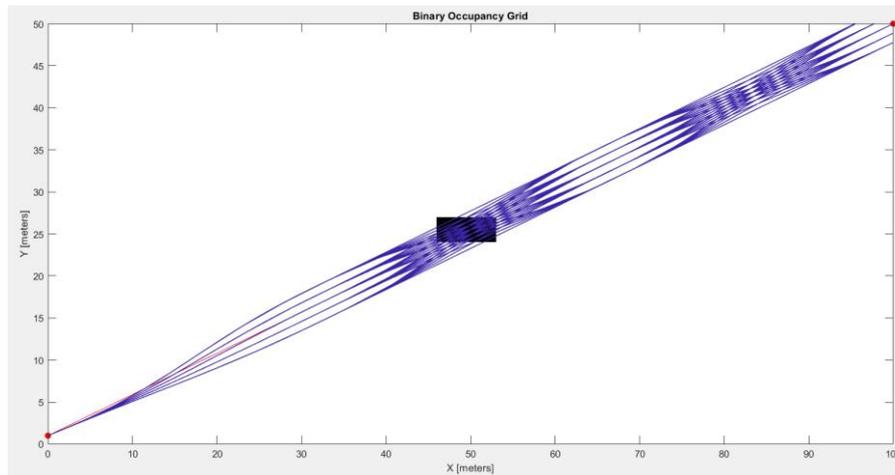


Figura 20 – Prima prova superamento ostacolo percorso diagonale

Nessuna traiettoria è stata trovata in presenza di questo ostacolo. Si valuta il risultato variando NumSegments = 4:

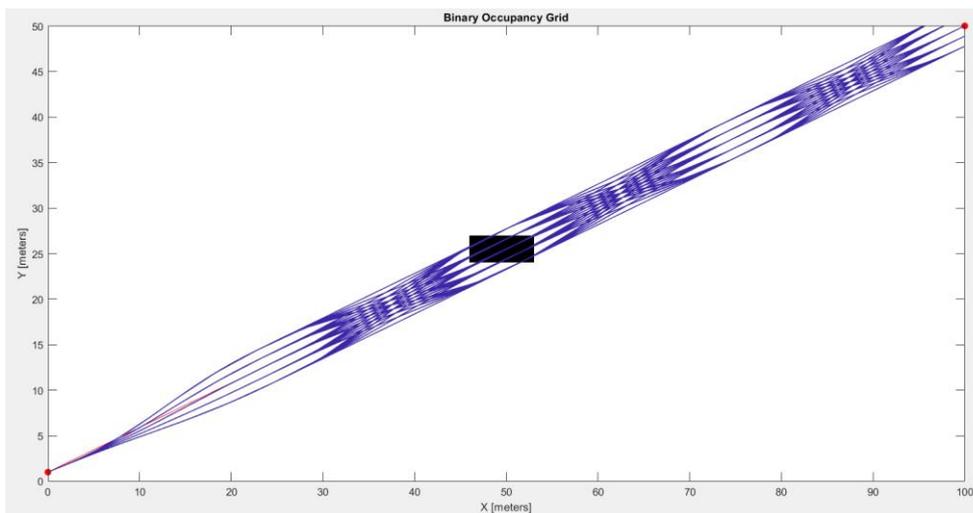


Figura 21 – Seconda prova: aumento di segmenti

Anche in questo caso non è stata trovata una soluzione, pertanto si prova ad allargare il parametro `planner.TerminalStates.Lateral` da `-2:1:2` a `-4:2:4` tenendo `NumSegments=4`:

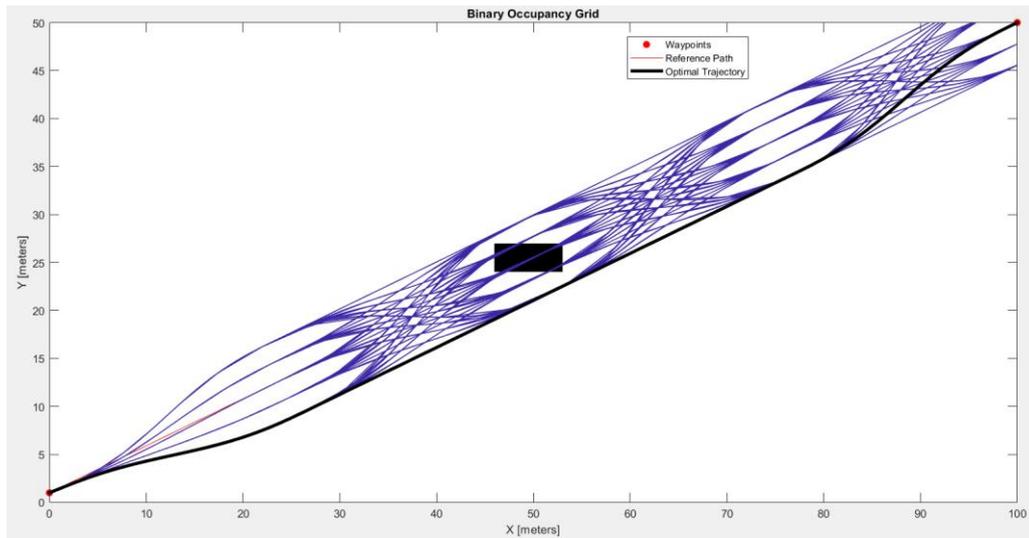


Figura 22 – Terza prova: allargamento traiettorie su cui ricercare

In questo caso una traiettoria adeguata, che eviti le collisioni e arrivi al punto di destinazione esatto è stata trovata. In generale, impostando il parametro `planner.TerminalStates.Lateral` più largo possibile e più sarà alta la probabilità di trovare una traiettoria adeguata. Questo non avviene sempre, dipende dallo scenario con cui ci si deve interfacciare. Ovviamente, un aumento della dimensione del vettore `planner.TerminalStates.Lateral` e un aumento di `NumSegments`, comporta un aumento del tempo richiesto al calcolo della soluzione, pertanto, si cercherà, in scenari contenenti via via sempre più ostacoli, di ottenere il loro valore minimo possibile.

2.3.1 Introduzione più ostacoli (con `NumSegments = 4`)

Con l'introduzione di altri due ostacoli (rappresentati nella figura seguente) la traiettoria priva di collisioni che rispetti tutti i vincoli imposti per il pianificatore viene trovata.

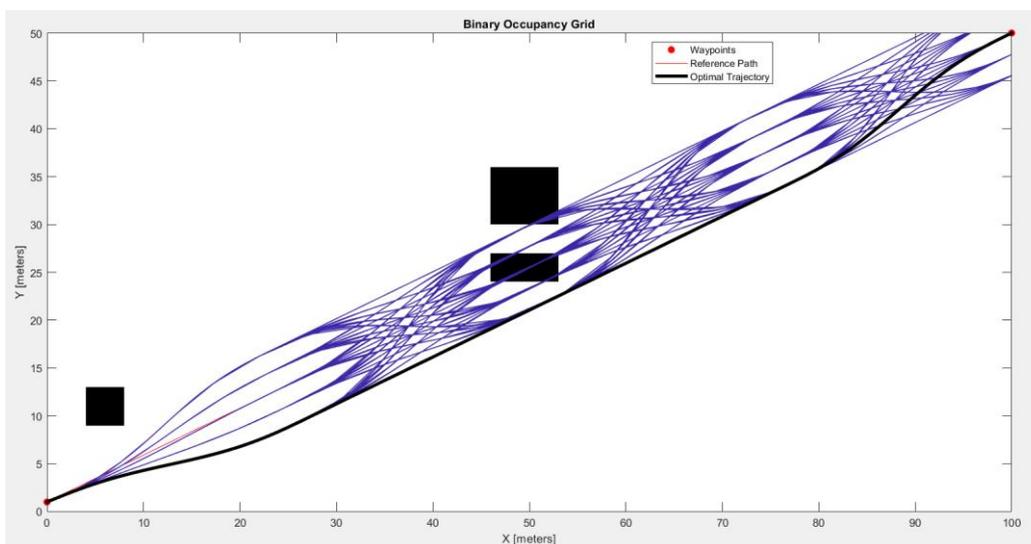


Figura 23 – 1° Introduzione più ostacoli

Introducendo un altro ostacolo che ostruisca la traiettoria nera precedentemente trovata:

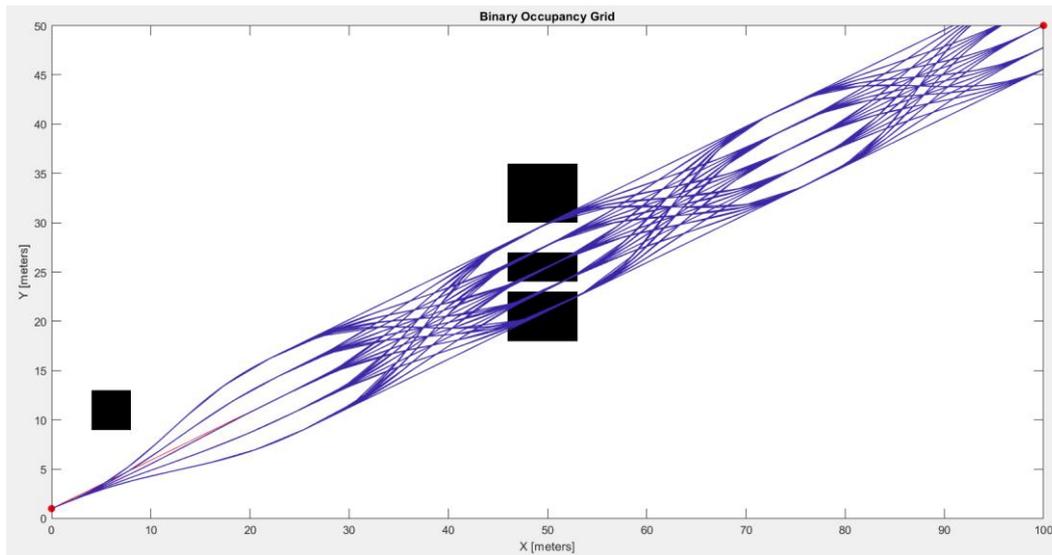


Figura 24 – Introduzione nuovo ostacolo: nessuna traiettoria trovata

Una traiettoria idonea alle nostre richieste e priva di collisioni non viene più identificata. Si prova a introdurre una maggiore libertà sulla deviazione laterale dalla traiettoria principale e un numero di segmenti più elevato per verificare che il metodo continui a funzionare. Si ricorda, però, che l'introduzione di più segmenti e una quantità di traiettorie sul laterale maggiore richiede più tempo di calcolo. `planner.TerminalStates.Lateral = -8:4:8;`

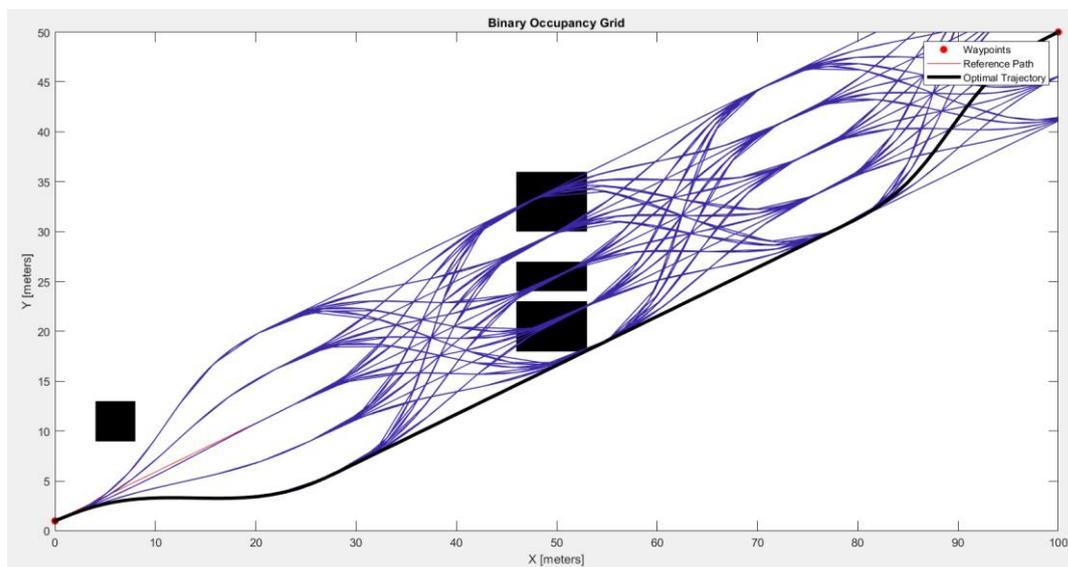


Figura 25 – Allargamento traiettorie su cui eseguire la verifica

A livello computazionale questo calcolo viene eseguito nello stesso tempo dell'esempio precedente perché il numero di traiettorie calcolate è lo stesso (6250). Una configurazione che richiederebbe troppo tempo è la seguente: `planner.TerminalStates.Lateral = -8:2:8;` Richiede 65610 traiettorie calcolate e un tempo di calcolo approssimabile al minuto (per il computer che viene utilizzato nello svolgimento della presente tesi). Il risultato che si ottiene è di carattere scenografico, ma non permette una sua

implementazione in un calcolatore che debba lavorare in real time, a meno di supercomputer dedicati.

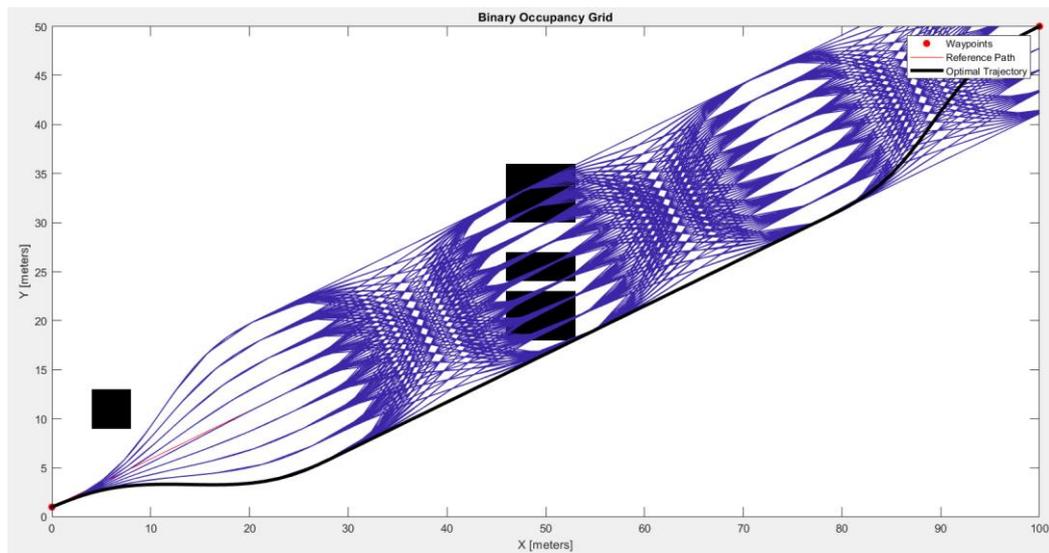


Figura 26 – Prova con elevato tempo di calcolo

Da ora in avanti sarebbe preferibile utilizzare tra le 3 e le 7 traiettorie in laterale (in quest'ultimo esempio erano 9) per mantenere i tempi di calcolo contenuti.

Introduzione ulteriori ostacoli

Si introduce un ulteriore ostacolo che provi a bloccare il passaggio. Si imposta, rispetto alla precedente simulazione: `planner.TerminalStates.Lateral = -8:4:8;`

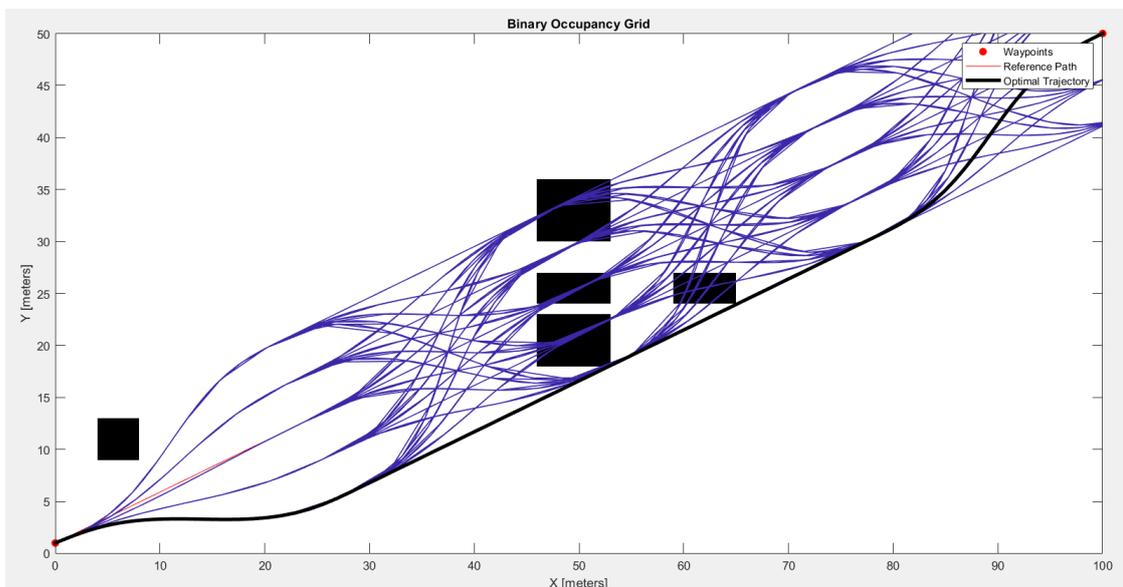


Figura 27 – Riduzione numero traiettorie calcolate

Si inserisce un ostacolo sulla traiettoria trovata sperando di limitare effettivamente il passaggio:

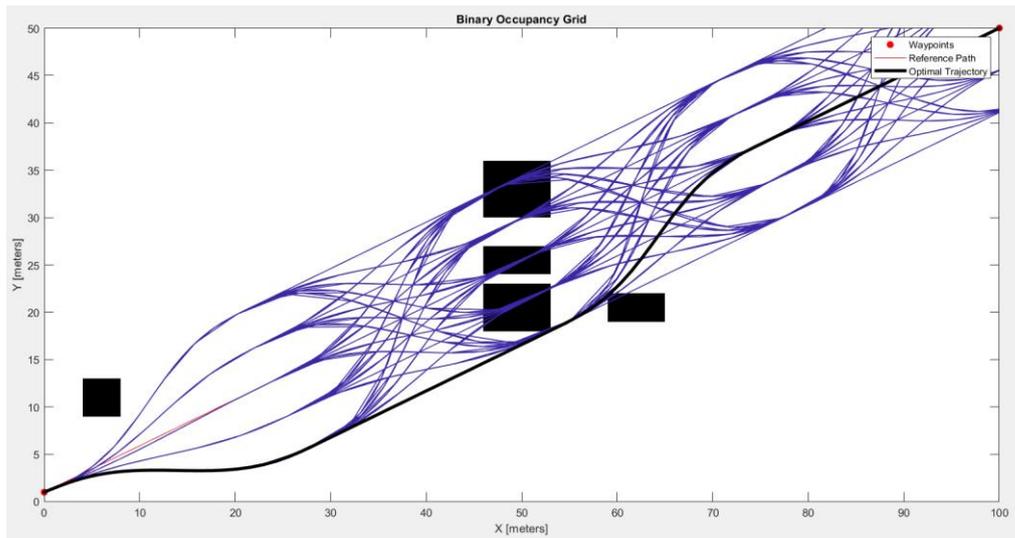


Figura 28 – Spostamento del nuovo ostacolo

Introducendo un ostacolo più largo rispetto al precedente, come ci si aspetta, non si trova più neanche una traiettoria valida di riferimento:

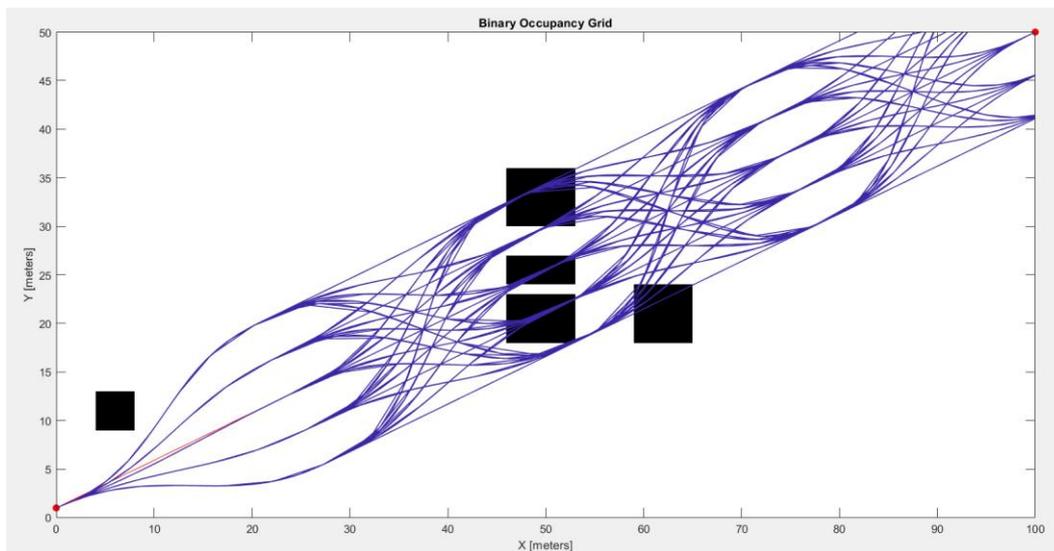


Figura 29 – Ingrandimento nuovo ostacolo

L'ottenimento o meno di una traiettoria adeguata dipende dal numero di traiettorie campione esaminate. Questo non garantisce una soluzione, ma in generale aumenta le probabilità di trovarla. Aumentando NumSegments a 6; Con 156250 traiettorie e tempi di calcolo eccessivamente elevati:

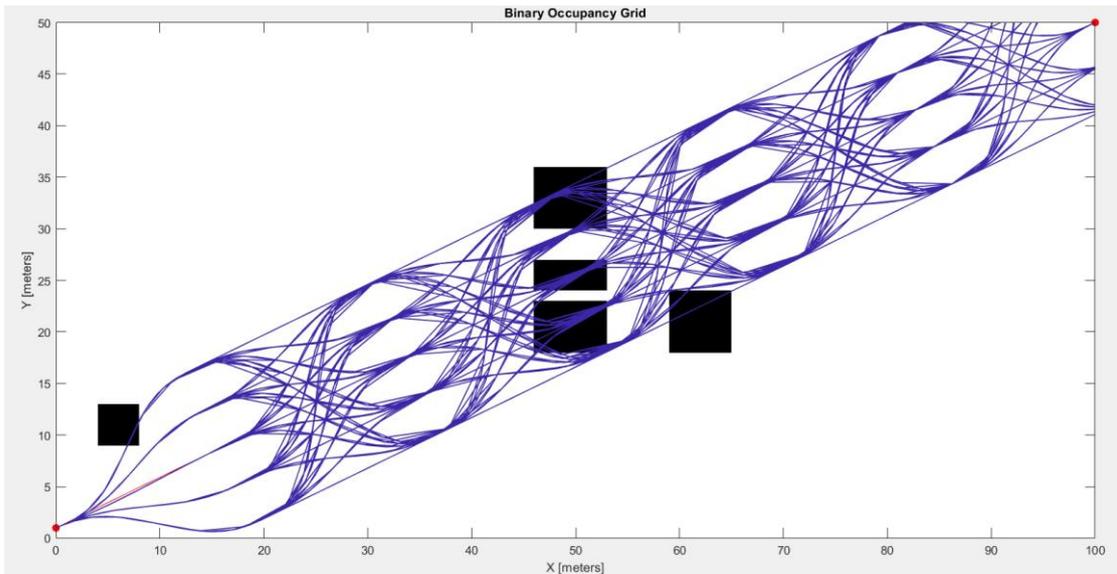


Figura 30 – Introduzione maggior numero di segmenti – nessuna traiettoria trovata

Nonostante siano presenti delle traiettorie adeguate al superamento degli ostacoli, il pianificatore non ha generato una soluzione da lui ritenuta accettabile. Questo risultato è dovuto al fatto che le traiettorie valide in quanto prive di collisioni con ostacoli, non rispettano i vincoli di massima curvatura o accelerazione laterale in un dato punto di esse.

Con NumSegments pari a 5, quindi riducendo il tempo di calcolo:

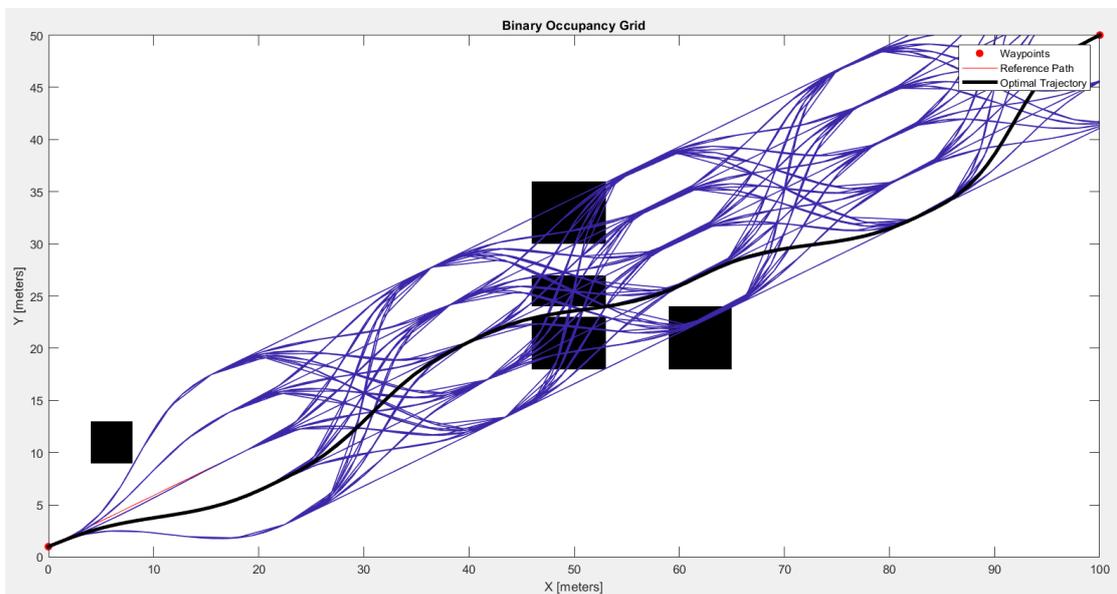


Figura 31 – Riduzione del numero di segmenti può portare a una soluzione migliore

La traiettoria è stata trovata, nonostante il costo computazionale sia sceso decisamente. Infatti, i segmenti considerati in questa prova sono stati 31250 (contro i 156250 della prova a NumSegments=6).

Il rintracciamento di una traiettoria adeguata non dipende solo dal numero di traiettorie considerate ma anche dalla generazione o meno di una traiettoria (tra le tante) che soddisfi **4 condizioni**:

[*velocity, acceleration, curvature, collision*] leggibili in '**planner.TrajectoryList.Feasible**'.
Questi 4 output del sistema possono assumere 3 possibili valori:

- 0 se la condizione non è rispettata
- -1 se la condizione non è stata valutata
- 1 se la condizione è rispettata

Quindi solo in corrispondenza di traiettorie con vettori [1,1,1,1] si ha una traiettoria valida per il pianificatore. Ovviamente, se si volessero allargare questi limiti lo si può fare nella fase di programmazione iniziale.

Mantenendo la stessa quantità di traiettorie utilizzate in quest'ultimo esempio proviamo a inserire una serie di ostacoli random.

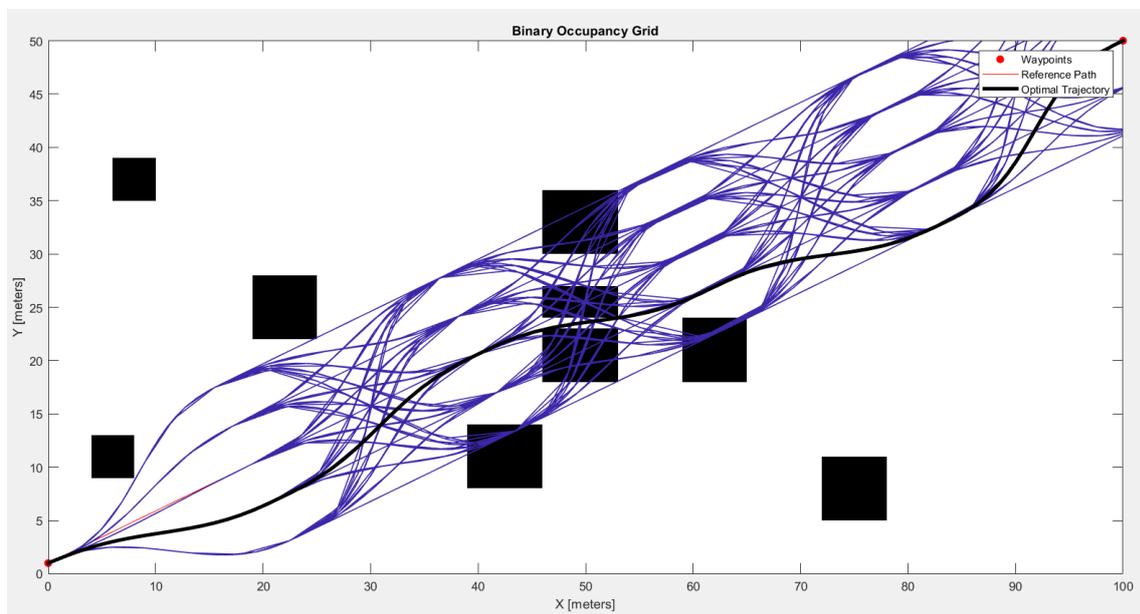


Figura 32 – Introduzione di ostacoli random

Mettendo a più dura prova il pianificatore:

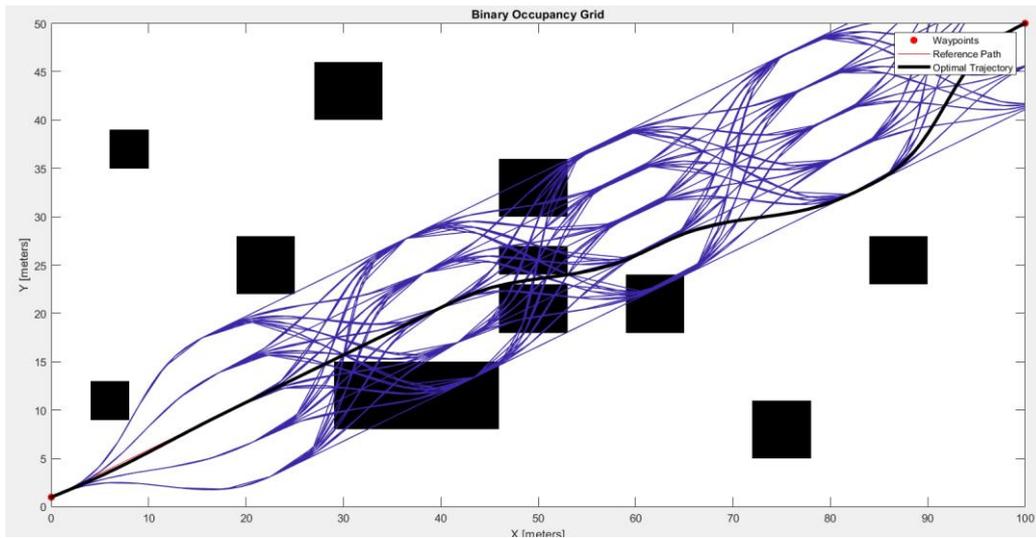


Figura 33 – Ingrandimento ostacoli

Si osservi che la traiettoria trovata, e indicata tramite la linea nera, non è la migliore che si possa ottenere. Questo problema è legato al fatto che il software di path planning dà come output la prima traiettoria che rispetti tutti i requisiti preimpostati. Le traiettorie valide che vengono dopo non vengono più esaminate dal sistema. Questo problema è stato superato tramite la modifica del codice 'trajectoryOptimalFrent' che verrà esaminato nel paragrafo apposito. Quindi, un nuovo codice è stato creato modificando leggermente quello sorgente, ed è stato richiamato 'trajectoryOptimalFrent_303622'.

Si è introdotto un ulteriore nuovo ostacolo che bloccasse la traiettoria precedentemente trovata.

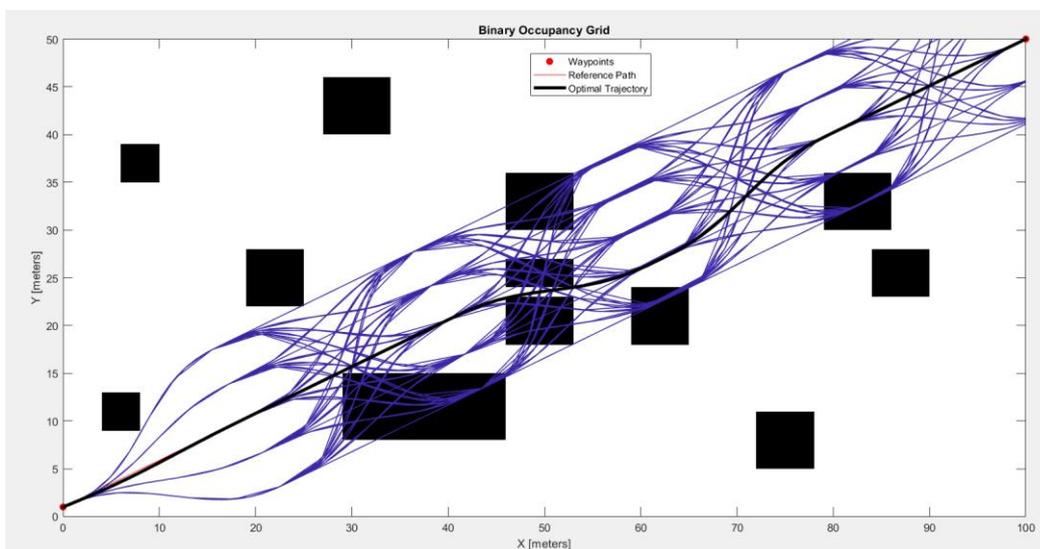


Figura 34 – Prova di blocco di una traiettoria precedentemente validata

Il pianificatore dimostra di essere in grado di trovare traiettorie anche quando lo scenario cambia.

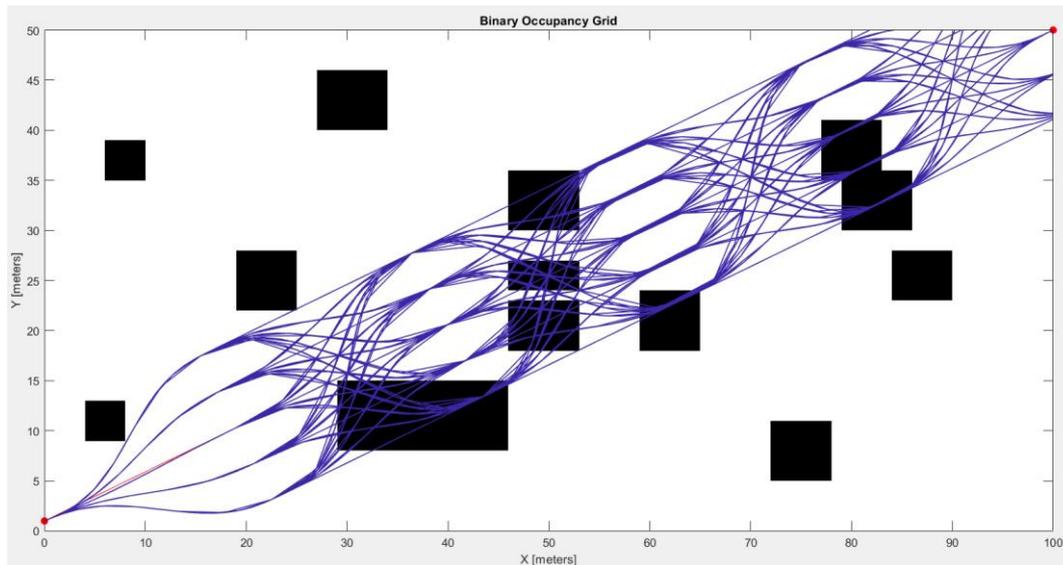


Figura 35 – Prova senza traiettorie di output

In questo caso non si riesce più a trovare una traiettoria che soddisfi le richieste: Si passa a un'analisi delle tabelle dei risultati, per vedere se un **'soft constraints'** variato possa portare all'ottenimento di una traiettoria. Analizzando i dati si nota che il problema risiedeva nell'accelerazione massima di alcune traiettorie che superava il limite di 10 m/s^2 precedentemente impostato. Provando ad alzare questo limite a 50 m/s^2 (non utilizzabile per un veicolo ma introdotta in fase di esplorazione delle funzionalità del codice) si ottiene effettivamente una soluzione:

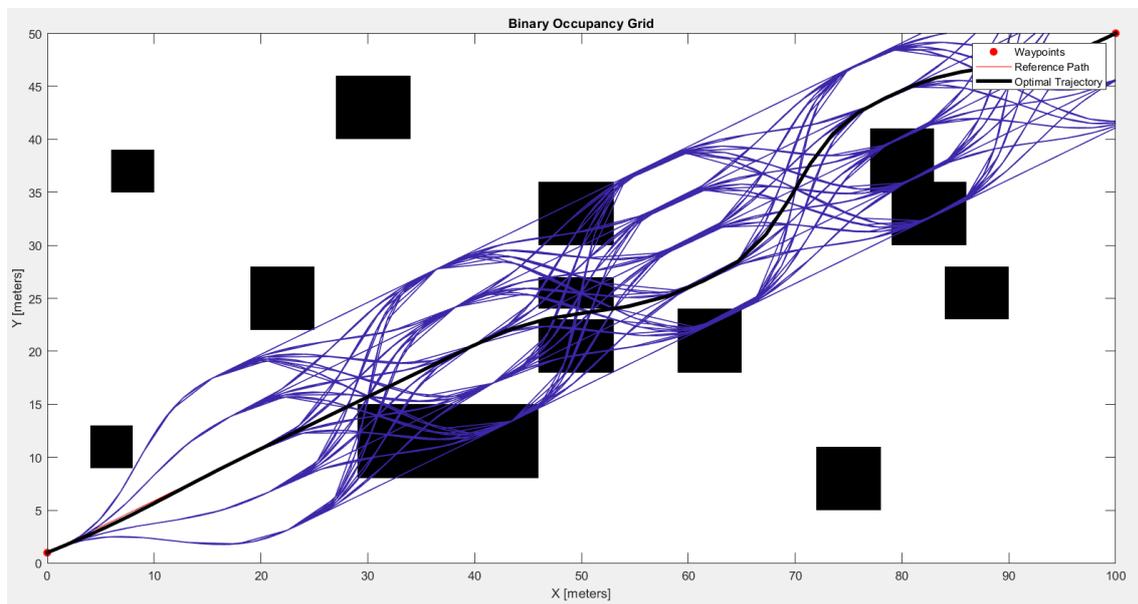


Figura 36 – Aumento accelerazione laterale limite (soft constraint)

La soluzione è stata trovata a scapito di un limite indispensabile per un veicolo terrestre (ovvero l'accelerazione massima di $8-10 \text{ m/s}^2$). In questo caso la traiettoria è stata eseguita in 5 secondi ma l'accelerazione di 10 m/s^2 è stata oltrepassata.

Variazione del tempo di percorrenza della traiettoria

Si è provato ad aumentare il tempo di percorrenza per l'effettuazione della traiettoria:

In questo caso il range di tempo analizzato TerminalStates.Time = [2:2:20] e la massima accelerazione ammissibile è stata lasciata a 50 m/s². Il risultato che si osserva è il seguente:

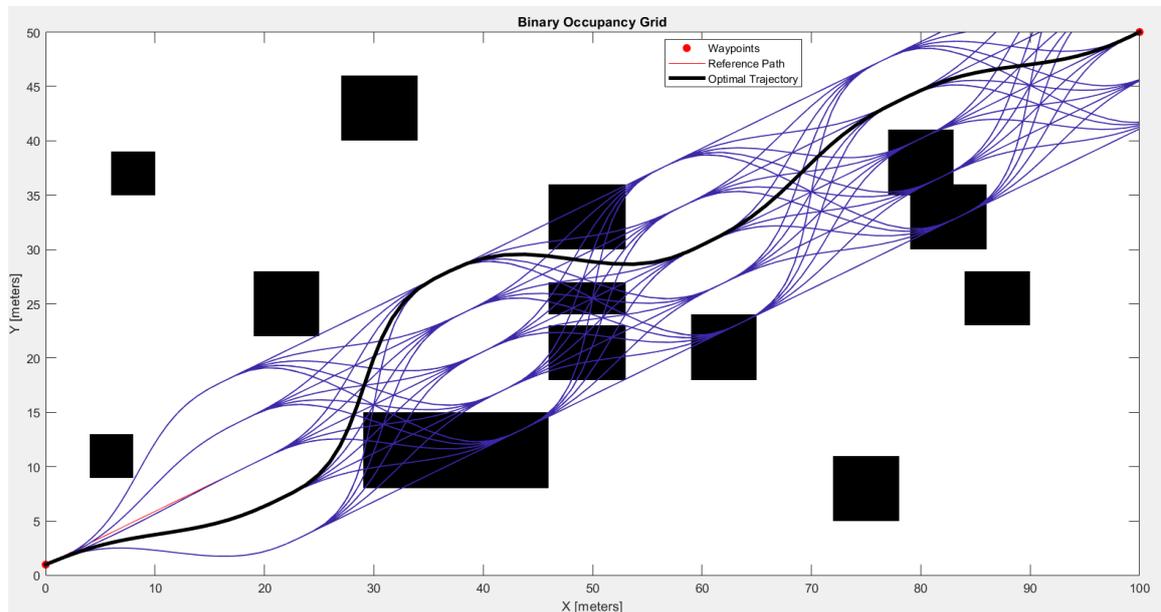


Figura 37 – Aumento del tempo di simulazione

La traiettoria è stata trovata e segue anche un percorso più ondulato e graduale. Il suo tempo di percorrenza è di 10 secondi con un'accelerazione massima di 19.3 m/s², ovviamente troppo elevata per un veicolo comune.

Bisognerebbe implementare una parte di codice che limiti la velocità in corrispondenza delle zone con maggiore curvatura per limitare l'accelerazione laterale del veicolo, mentre, le velocità di percorrenza vengono scelte dal pianificatore con leggi non note all'utente.

Un controllo di velocità che limiti la velocità longitudinale in prossimità di punti in cui la curvatura è maggiore verrà implementato nel capitolo 3 dedicato al path tracking.

Provando a ostacolare la traiettoria appena creata, si valuta se il pianificatore riesca anche in questo caso a trovare una soluzione:

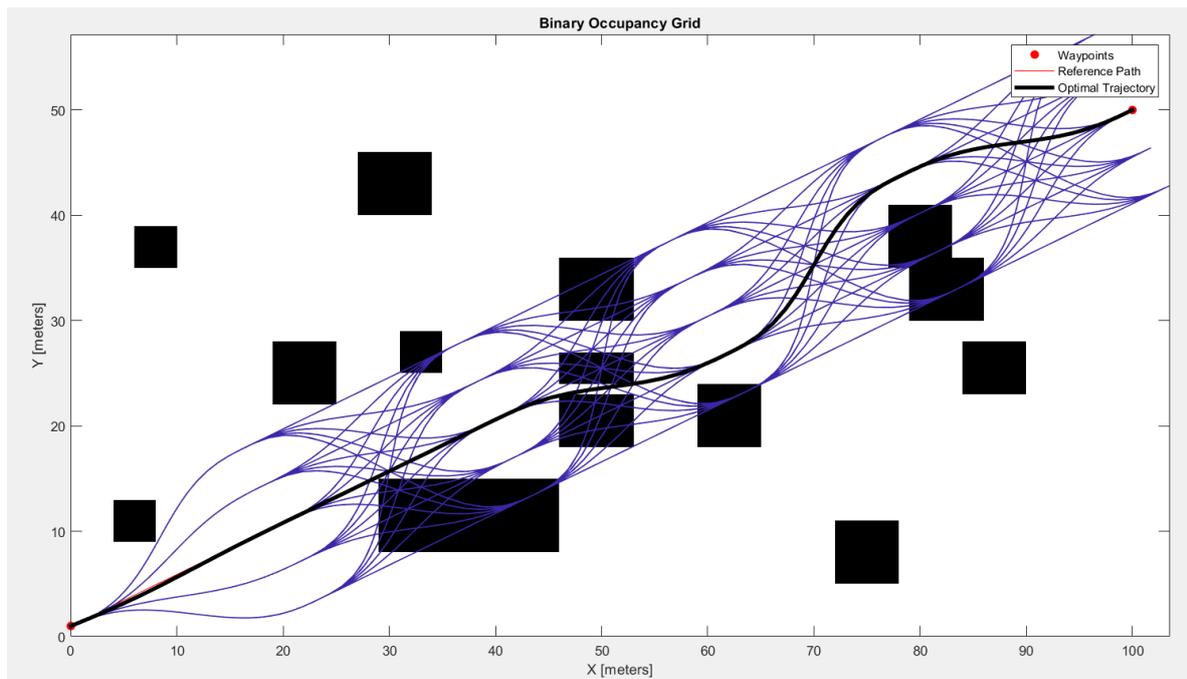


Figura 38 – Aggiunta ostacolo alla traiettoria di riferimento precedente

Una traiettoria viene trovata, ma evita con pochissimo margine le collisioni con ostacoli. Se si tenesse conto delle dimensioni del veicolo, probabilmente ci sarebbe uno scontro.

Provando ancora ad allargare il range in trasversale tra le varie traiettorie si giunge al seguente risultato: `planner.TerminalStates.Lateral = -10:4:10; NumSegments = 5;`

Con 77760 traiettorie valutate il risultato è il seguente:

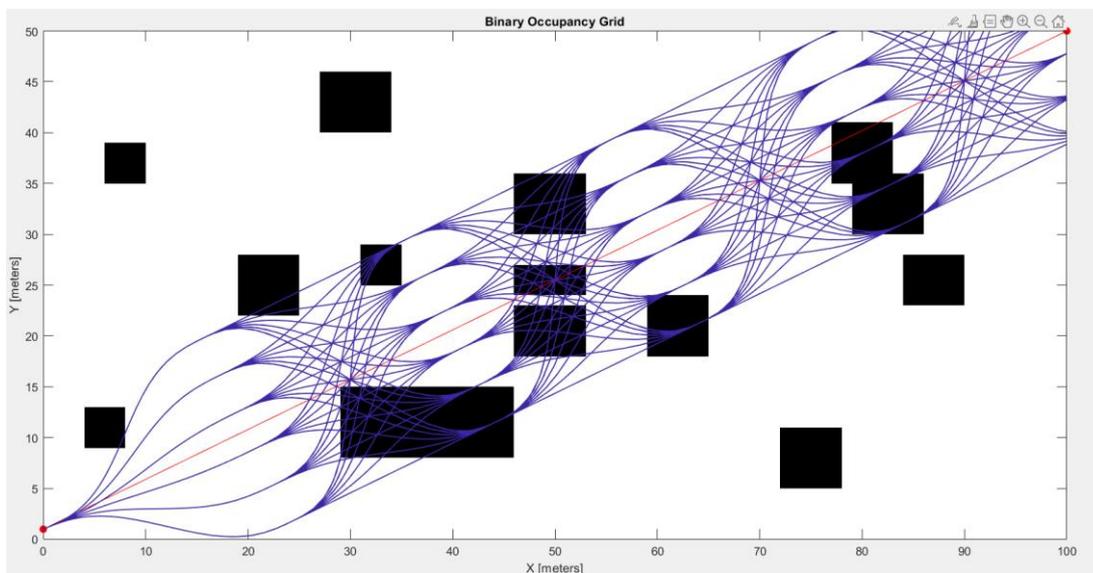


Figura 39 – Allargamento traiettorie da valutare

Non viene identificata una traiettoria valida anche se delle traiettorie prive di collisioni sono presenti. Provando a togliere il vincolo di curvatura massima e di accelerazione massima (posti entrambi pari a 10000): Il risultato non cambia, non viene trovata una traiettoria.

Configurando `planner.TerminalStates.Lateral = -12:4:12`; e `NumSegments = 5`; con tempo di calcolo pari a 10 minuti circa e 168070 traiettorie calcolate, si ottiene:

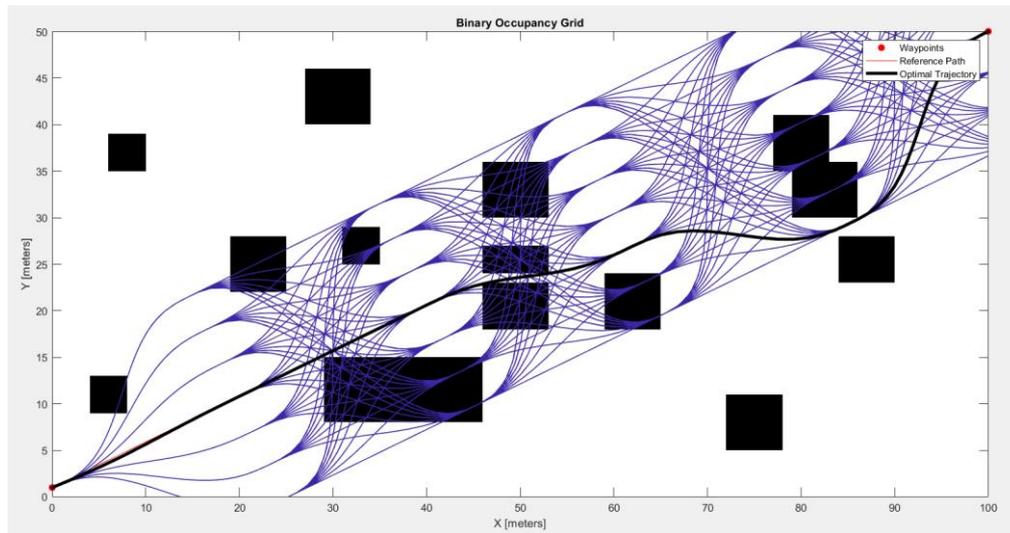


Figura 40 – Ampliamento in trasversale delle traiettorie campione

Si osservi che il pianificatore identifica una traiettoria valida solo se il numero di traiettorie sul laterale è dispari; infatti, se il numero di traiettorie è dispari ci sarà un flusso di traiettorie che converge esattamente al punto di destinazione (B); se, invece, il numero di traiettorie è pari, questa confluenza non ci sarà, non permettendo di avere una traiettoria opportuna. Modificando il parametro `DeviationOffset` da 0 a un valore maggiore, si otterrà, anche nel caso di traiettorie dispari, una traiettoria valida.

2.3.2 Più punti di passaggio (Waypoints)

Esiste la possibilità di impostare più di due punti di passaggio (‘waypoints’) di riferimento attraverso cui le traiettorie devono passare, questo significa avere non solo i punti A e B di partenza e di arrivo, ma anche degli altri punti C,D,...

Impostando i seguenti parametri:

```
refPath=planner.Waypoints = [0,1;95,5;100,50]
```

```
NumSegments = 4
```

```
planner.TerminalStates.Lateral = -8:4:8
```

```
planner.TerminalStates.Time = [2:2:20]
```

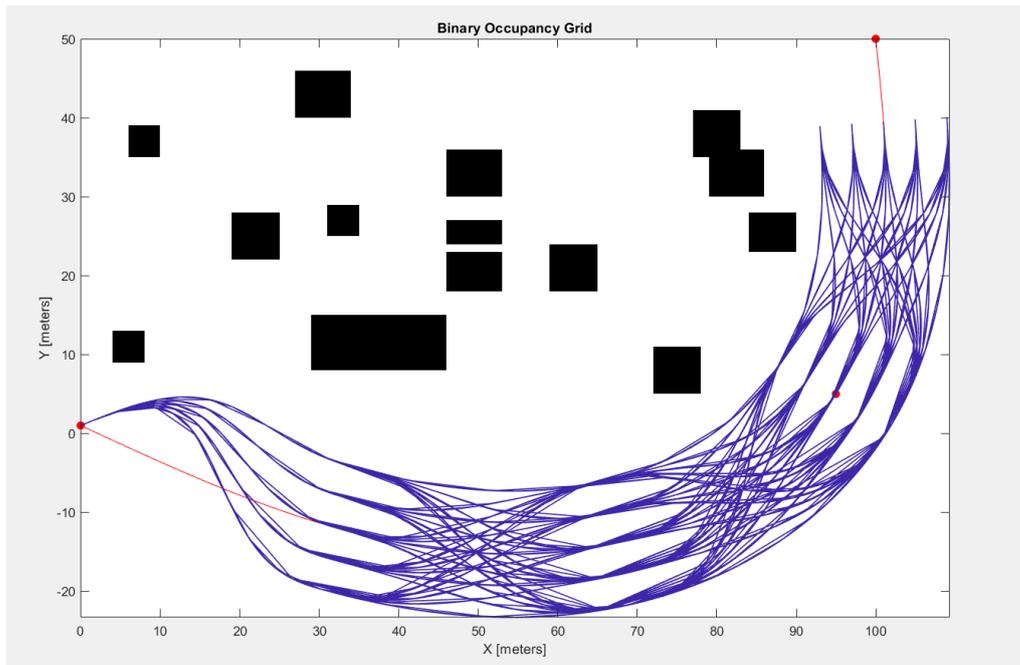


Figura 41 – Inserimento terzo ‘waypoint’

Una generazione di traiettorie di questo tipo non ha alcun senso ingegneristico, se non l’ottenimento di traiettorie curve prive di punti angolosi.

Il grosso limite che di questa configurazione risiede nella specifica della lunghezza della traiettoria. Tramite il parametro `planner.TerminalStates.Longitudinal` si deve impostare la lunghezza totale compiuta dal veicolo, ma questo dipende dalla forma stessa della traiettoria che viene selezionata dal pianificatore, pertanto, si notano queste interruzioni di percorso prima del raggiungimento del punto di arrivo. Si noti questo esempio:

```
planner.TerminalStates.Longitudinal = 100
```

```
planner.TerminalStates.Lateral = -8:4:8
```

```
planner.NumSegments = 4
```

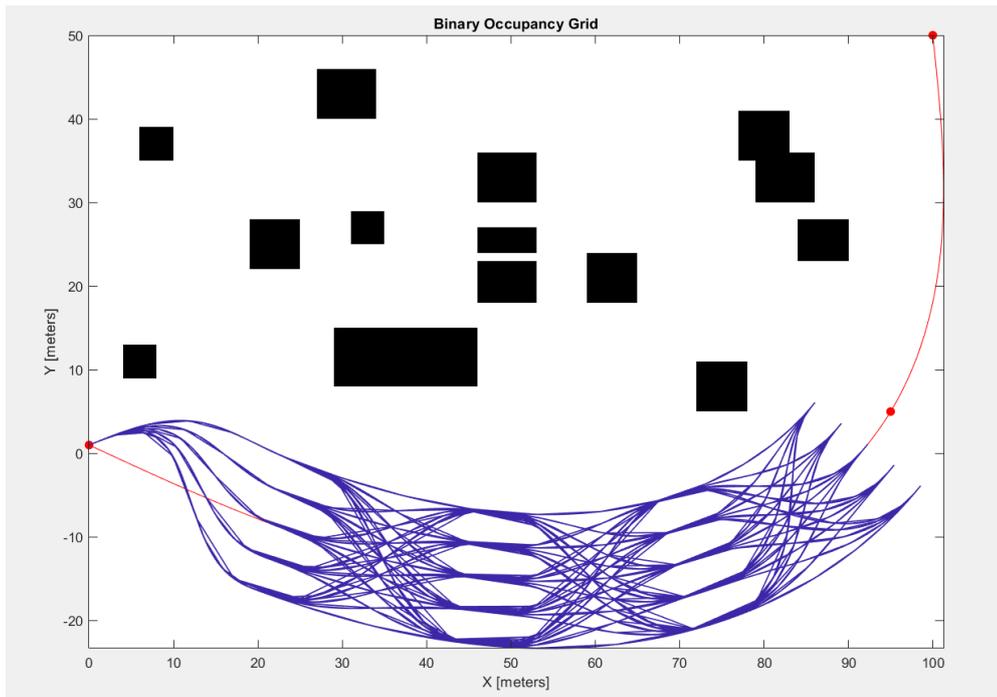


Figura 42 – Prova fallimentare nell'introduzione di un terzo 'waypoint'

Si nota che il punto di arrivo non viene raggiunto perché le traiettorie non sono state impostate sufficientemente lunghe. Proviamo a imporre:

`planner.TerminalStates.Longitudinal = 100:50:150` in modo da lasciare libertà di scelta della lunghezza totale finale del percorso;

`planner.NumSegments = 3;`

Si ottiene una soluzione più valida:

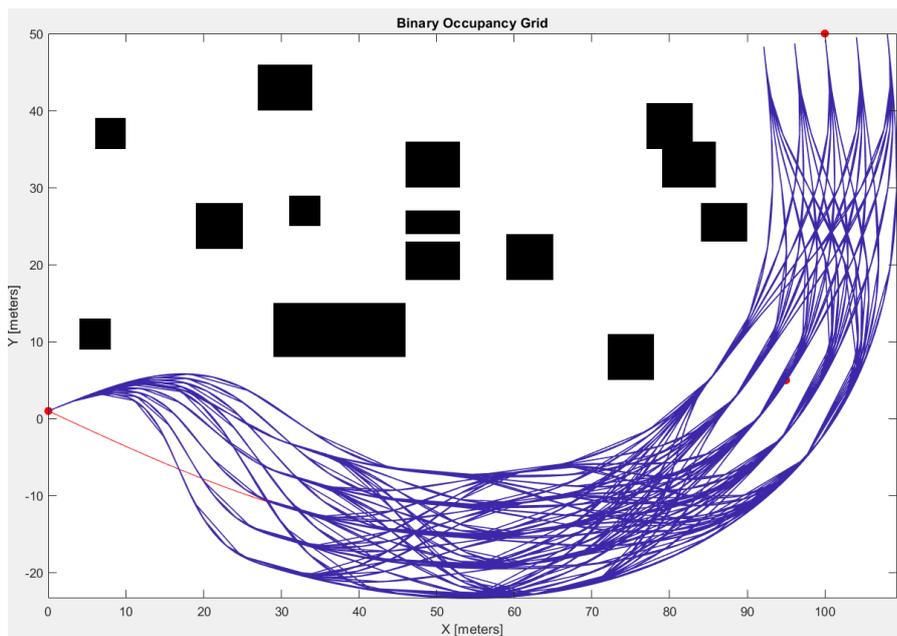


Figura 43 – Raggiungimento del punto finale in presenza di un terzo waypoint

Il motivo della generazione di questa serie di traiettorie con tale curvatura non è noto. Si consiglia di assegnare un calcolo della traiettoria solamente tra due punti (A e B) di partenza e arrivo delle traiettorie generate, limitando, se necessario, il campo di stima della traiettoria a tratti di lunghezza inferiore. Questo può essere fatto agilmente in real time per mezzo di calcolatori a bordo veicoli dedicati a questa operazione.

2.4 Variazione funzione trajectoryOptimalFrenet.m

Durante l'utilizzo del pianificatore/generatore di traiettorie "trajectoryOptimalFrenet.m" è stato riscontrato un problema legato al fatto che una volta trovata una traiettoria valida, dal punto di vista dei limiti e vincoli imposti, la validazione di traiettorie terminava senza dare l'opportunità ad altre traiettorie migliori di essere selezionate. Pertanto, si è deciso di modificare leggermente il codice "trajectoryOptimalFrenet.m" in modo tale da avere una valutazione di tutte le traiettorie possibili, senza escludere tutte quelle che venivano dopo la prima valida rilevata.

Si è, quindi, creato un nuovo codice: "trajectoryOptimalFrenet_303622.m" in cui questo limite viene eliminato interrompendo l'arresto alla ricerca di traiettorie appena descritto.

Viene anche introdotto un peso sulla distanza compiuta in una specifica traiettoria, in modo tale da ottenere la traiettoria priva di collisioni, che rispetti tutti i vincoli imposti dall'utente e a distanza minore. Questa traiettoria verrà successivamente riportata nei modelli di path tracking che renderanno possibile l'esecuzione del percorso.

2.5 Analisi dei risultati e utilizzo

I risultati del codice, così modificato, sono stati analizzati tramite una serie di 'plot' 2D e 3D. Sono stati messi in evidenza un insieme di risultati relativi all'esempio che segue:

Tabella 1 - Coordinate globali traiettoria path planning

planner.FeasibilityParameters.MaxCurvature	10
planner.FeasibilityParameters.MaxAcceleration	10
planner.TerminalStates.Lateral	-4:2:4
planner.TerminalStates.Time	[2:2:20]
planner.NumSegments	3
planner.DeviationOffset	0
initCartState	[0 30 pi/8 0 0 0]
FinalCarState	[100 20 / / / /]
planner.TimeResolution	0.1

Le traiettorie che vengono generate e poi valutate in questo esempio sono state 1250. Il tempo di calcolo non è significativo, pertanto applicabile in tempo reale.

Lo scenario che il generatore di traiettorie doveva affrontare è il seguente, relativo alla mappa già usata nello studio di sensitività del planner:

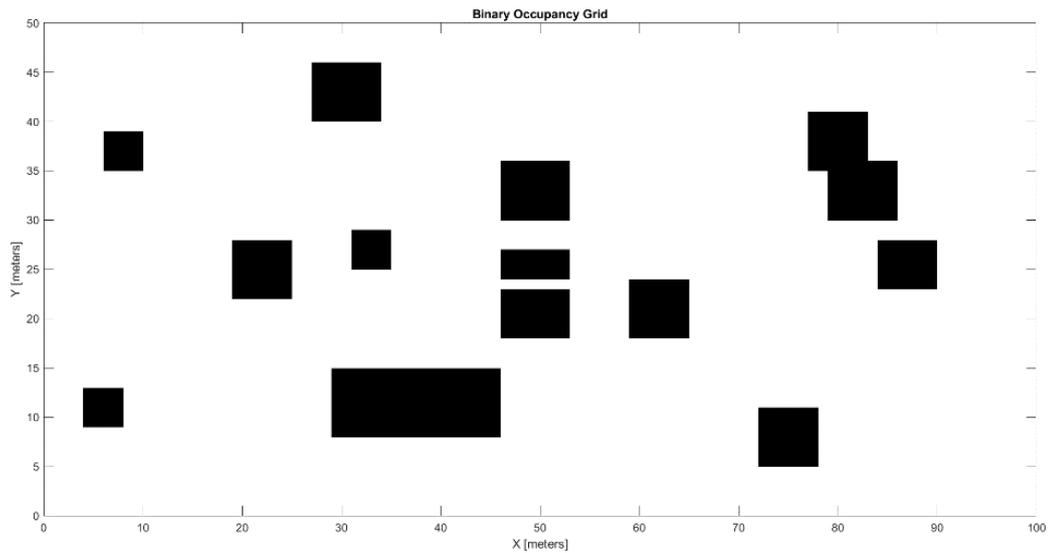


Figura 44 – Mappa ad ostacoli per validazione del path planner

Gli stati/punti di partenza e di arrivo sono:

$$[X_{start}, Y_{start}, X_{end}, Y_{end}] = [0, 30, 100, 20]$$

Le traiettorie che il codice ha generato sono le seguenti (è anche presente una delle traiettorie fattibili, ma che in questo caso non arriva al punto esatto di destinazione, bensì è la prima traiettoria valida trovata):

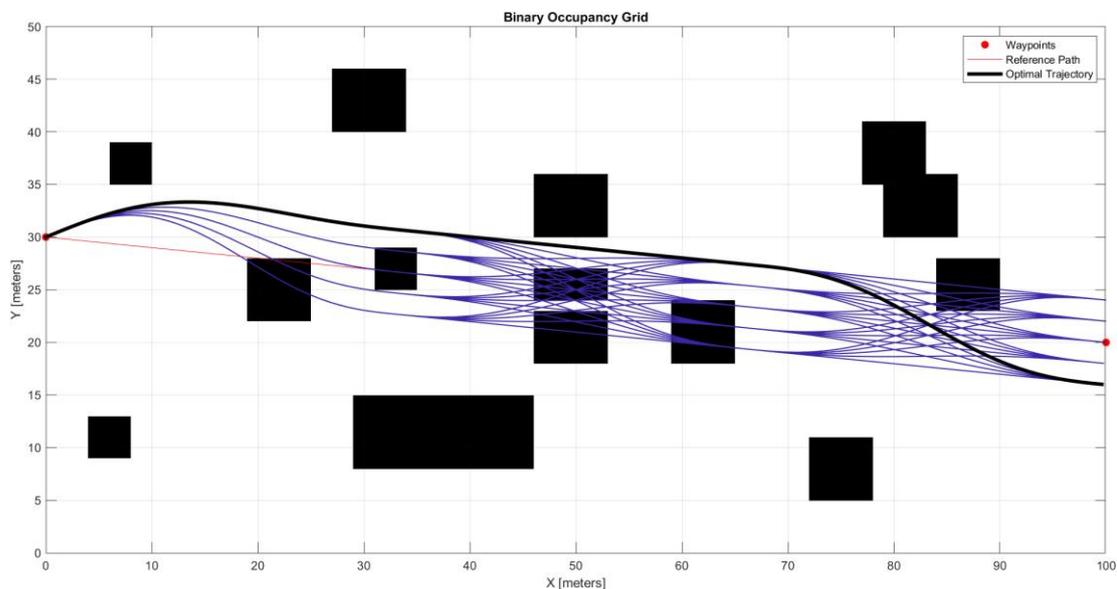


Figura 45 – Prima traiettoria valida trovata dal path planner

Dalle analisi che sono state condotte, tramite l'implementazione di qualche riga di codice aggiuntiva, si è potuta valutare la presenza di 10 traiettorie 'fattibili' (quindi prive di collisioni, che non superassero la curvatura, accelerazione e velocità massime consentite).

```
Le traiettorie valide sono state:10
Tra quelle trovate, la più breve è la:3
La distanza percorsa è di:101.6069
```

Tramite l'erogazione di questo avviso si possono estrapolare una serie di informazioni relative al numero di traiettorie trovate, quale delle traiettorie sia quella più breve e la lunghezza di questa in particolare. Tale lunghezza (curvilinea) viene valutata sia tramite una semplice somma dei tratti compresi tra stati successivi, sia tramite un metodo di approssimazione delle curve con polinomi di terzo grado e successiva somma delle distanze tra gli stati di questi polinomi interpolanti. (guardare il codice per maggiore chiarezza).

Alla fine, sono state trovate queste 10 traiettorie possibili (sovrapposte a gruppi di 2/3/4 traiettorie, differenti per il tempo impiegato nella loro esecuzione, come verrà messo in evidenza successivamente):

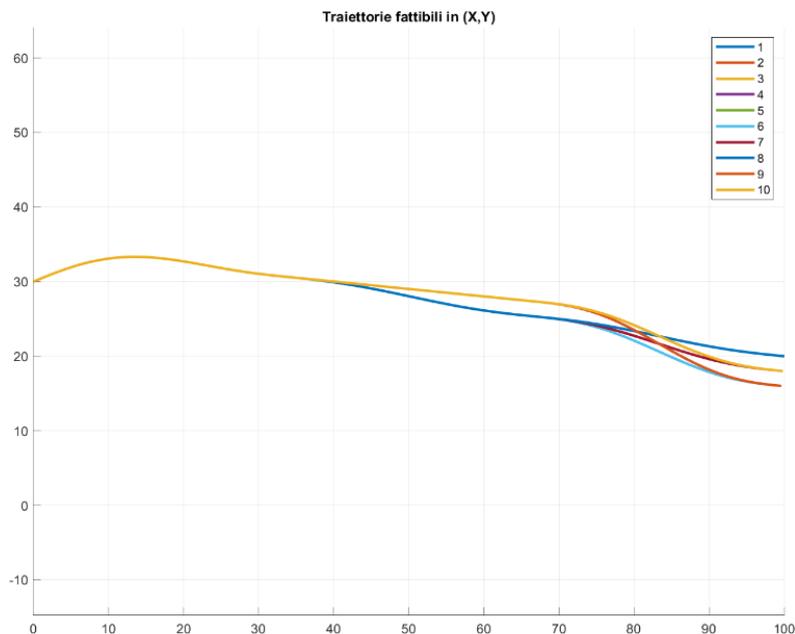


Figura 46 – Traiettorie valide trovate

Da queste traiettorie sono state separate quelle che arrivano proprio al punto di arrivo esatto preimpostato dall'utente:

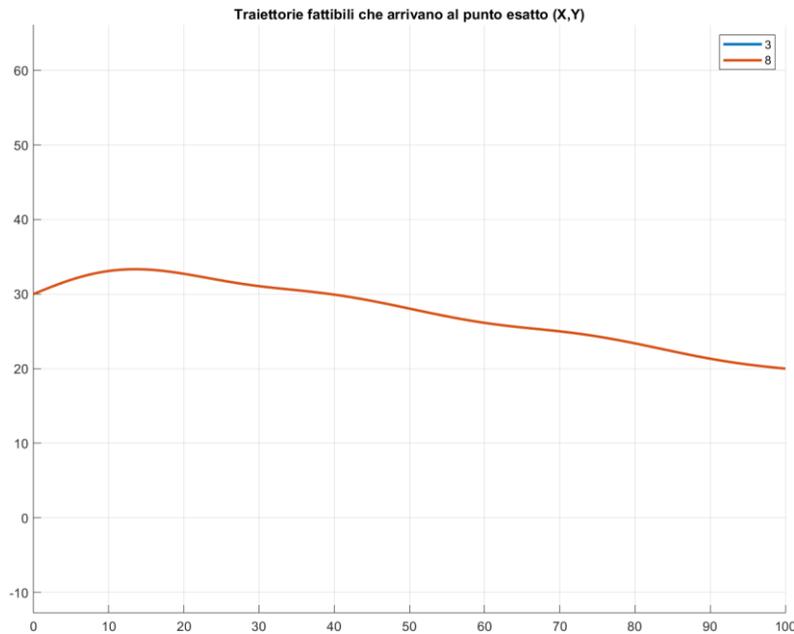


Figura 47 – Traiettorie valide che arrivano al punto esatto di destinazione

In questo caso, sono presenti due traiettorie uguali sovrapposte: la 3 e la 8 (tra le 10 di quelle selezionate). Un messaggio di avviso definisce la presenza di 2 traiettorie che soddisfino tutti i requisiti desiderati:

```
La traiettoria fattibile: 1 ovvero la: 616 non arriva al punto esatto di destinazione
La traiettoria fattibile: 2 ovvero la: 617 non arriva al punto esatto di destinazione
La traiettoria fattibile: 4 ovvero la: 621 non arriva al punto esatto di destinazione
La traiettoria fattibile: 5 ovvero la: 622 non arriva al punto esatto di destinazione
La traiettoria fattibile: 6 ovvero la: 866 non arriva al punto esatto di destinazione
La traiettoria fattibile: 7 ovvero la: 867 non arriva al punto esatto di destinazione
La traiettoria fattibile: 9 ovvero la: 871 non arriva al punto esatto di destinazione
La traiettoria fattibile: 10 ovvero la: 872 non arriva al punto esatto di destinazione
Le traiettorie valide che arrivano a destinazione sono: 2
```

Figura 48 - Messaggio di avviso di informazioni sulle traiettorie trovate valide

Infine, sono stati generati una serie di grafici 3D che permettono di analizzare

- VELOCITA'
- TEMPO
- ACCELERAZIONE
- ACCELERAZIONE TRASVERSALE
- ACCELERAZIONE LONGITUDINALE,

relativi alle traiettorie 'fattibili' (che non necessariamente arrivano a destinazione). Una cernita ulteriore permette di analizzare solo le 2 traiettorie designabili come 'eccellenti'.

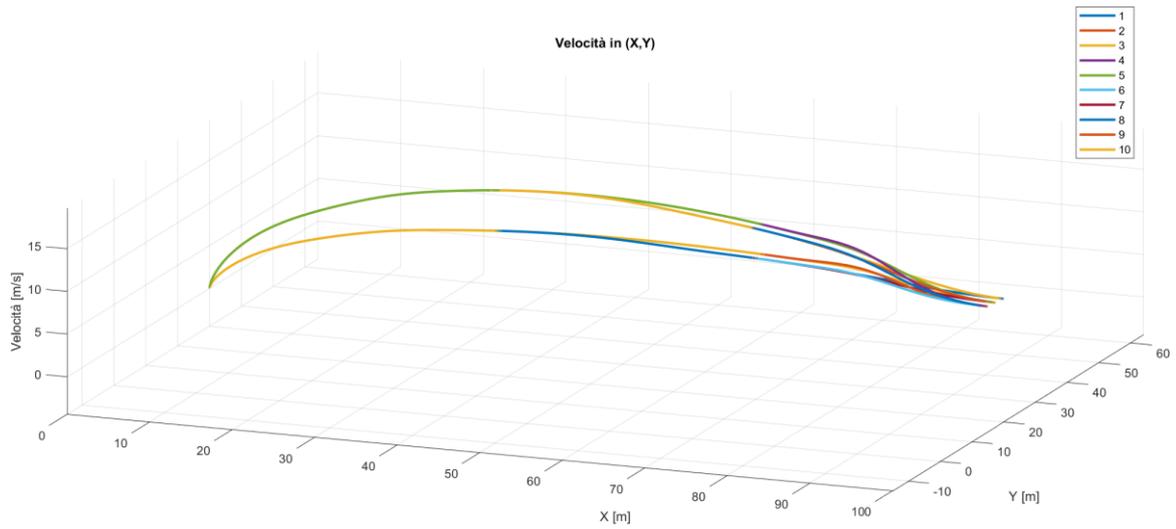


Figura 49- Velocità in funzione delle coordinate [X,Y]

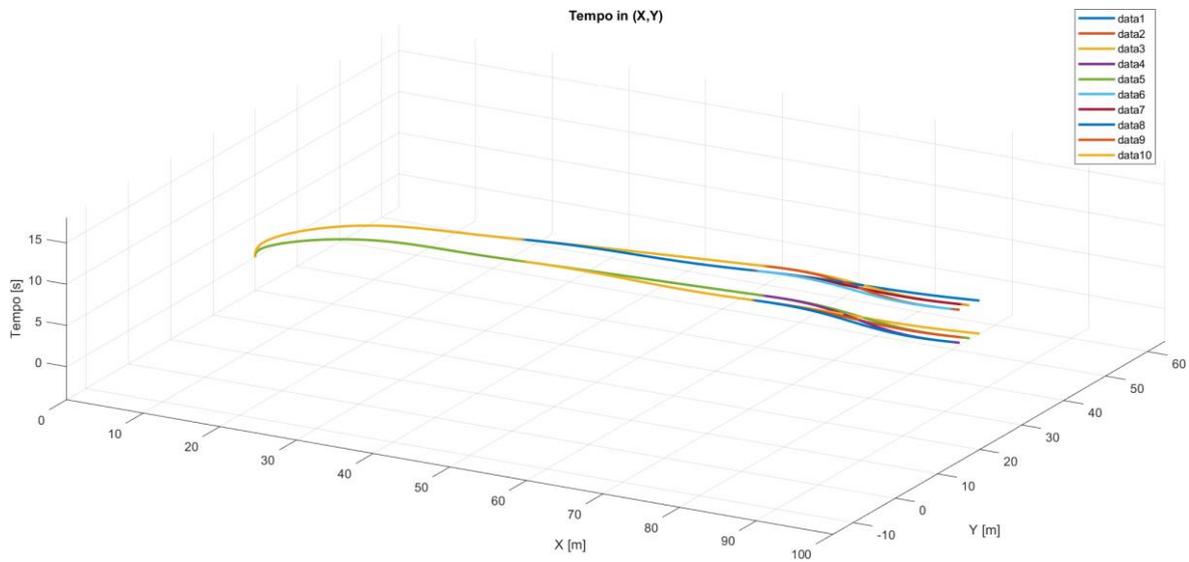


Figura 50 – Tempo di percorrenza in funzione delle coordinate [X,Y]

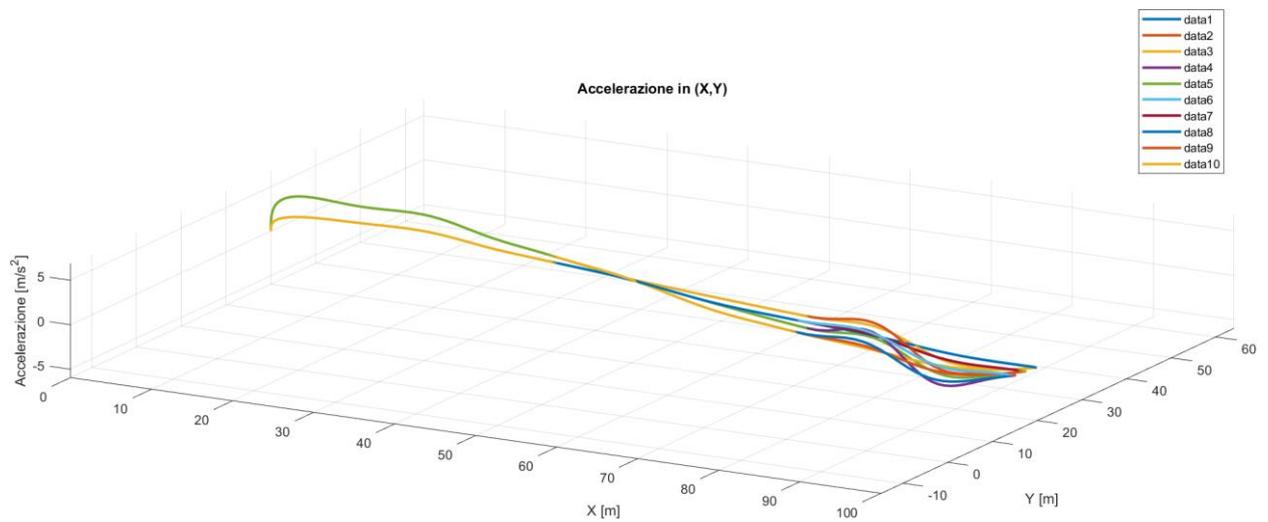


Figura 51 – Accelerazione restituita dal planner in funzione delle coordinate [X,Y]

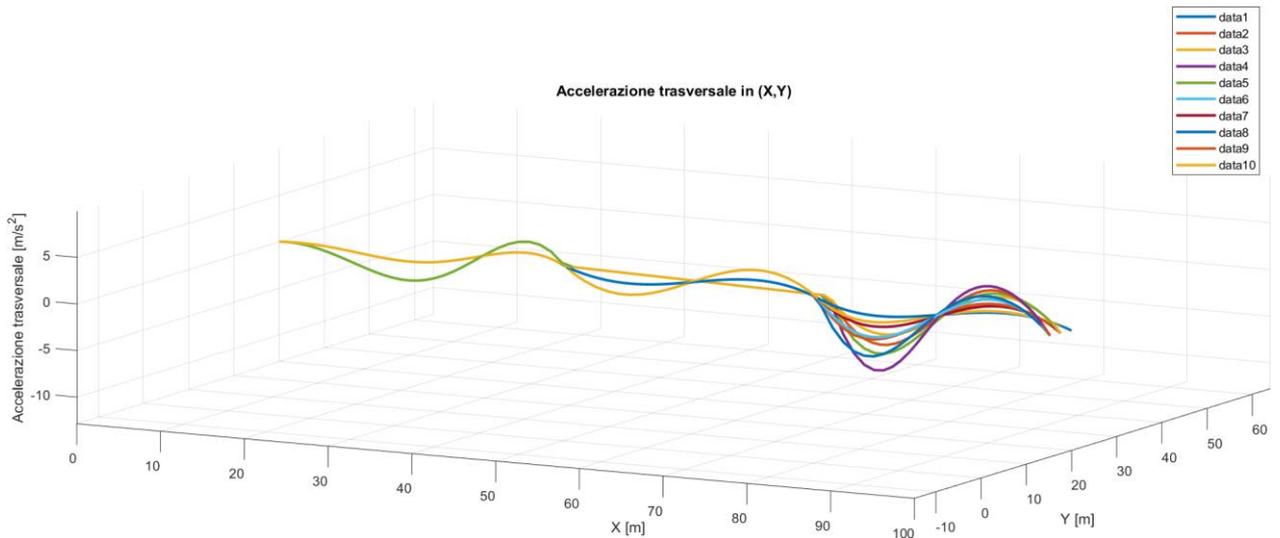


Figura 52 - Accelerazione trasversale in funzione delle coordinate [X,Y]

L'accelerazione trasversale è stata calcolata in ogni punto della traiettoria, per ogni singola traiettoria, come: $a_t = v^2 \cdot K$, dove v è la velocità longitudinale nel dato punto dello spazio e K la curvatura in quel punto ($K = 1/R$ dove R = raggio di curvatura della traiettoria in un dato punto).

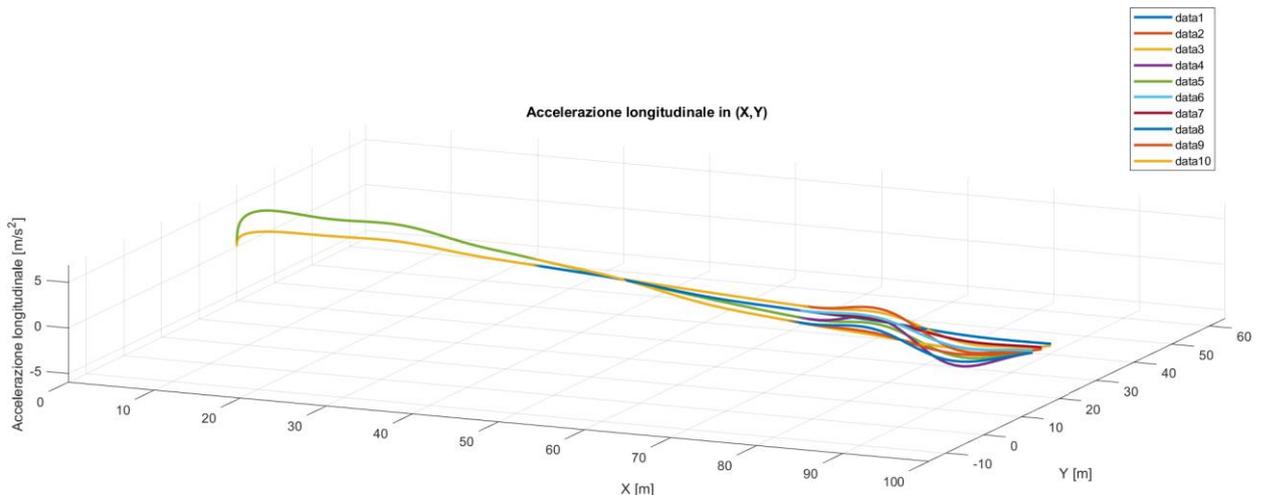


Figura 53 - Accelerazione longitudinale in funzione delle coordinate [X,Y]

Essa è stata calcolata con il metodo delle differenze finite centrate troncate al primo ordine. Calcolata, quindi, in ogni punto con la formula:

$$a_l = \frac{dv}{dt} = \frac{v^{(i+1)} - v^{(i-1)}}{2dt} \quad (2.7)$$

Dove i è un dato punto di campionamento della traiettoria. Con questo metodo, più preciso delle differenze finite in avanti o all'indietro, si escludono, però, l'accelerazione nel primo e nell'ultimo punto di campionamento di ogni traiettoria, quindi nel punto iniziale e finale.

Osservando i grafici dell'accelerazione e quello dell'accelerazione longitudinale ci si rende conto della stretta somiglianza tra i due. Pertanto, si è eseguito il calcolo della loro differenza per tutte e dieci le traiettorie valide trovate. Il risultato risulta chiaro:

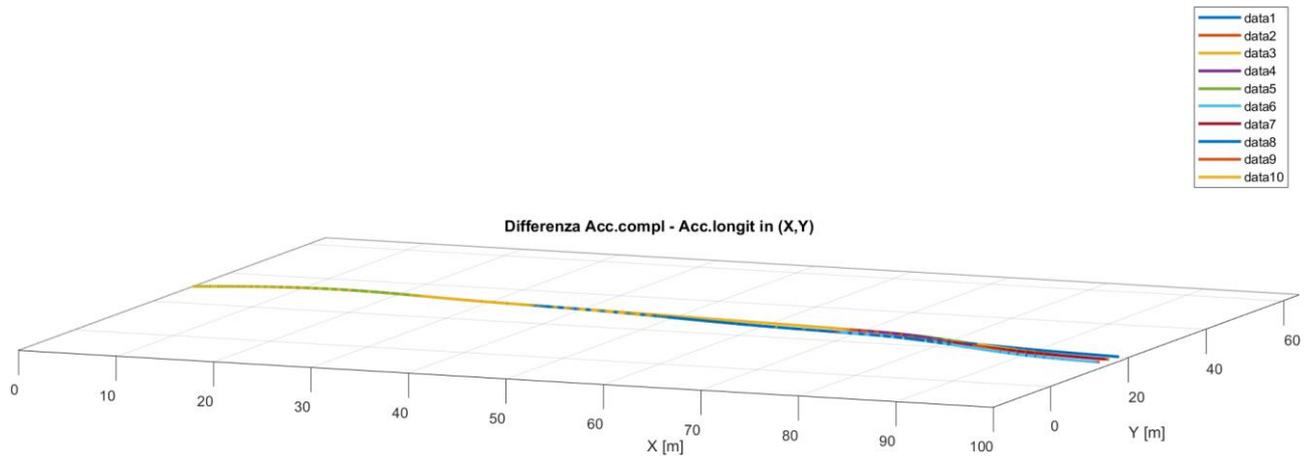


Figura 54 - Differenza Acc.pianificatore - Acc.longitudinale

Si può notare che **l'accelerazione calcolata dal pianificatore è quella longitudinale**. Pertanto, l'accelerazione complessiva è da ottenere sulla base di quella longitudinale e quella trasversale precedentemente calcolate.

Accelerazioni longitudinali vs Tempo

Si riporta il grafico che mostra le accelerazioni longitudinali in funzione del tempo di tutte le traiettorie valide. Le accelerazioni sono state calcolate a partire dalla velocità del veicolo e applicando il metodo delle differenze finite in avanti. Successivamente si è applicato un metodo di interpolazione, per rendere più graduale il grafico che si ottiene.

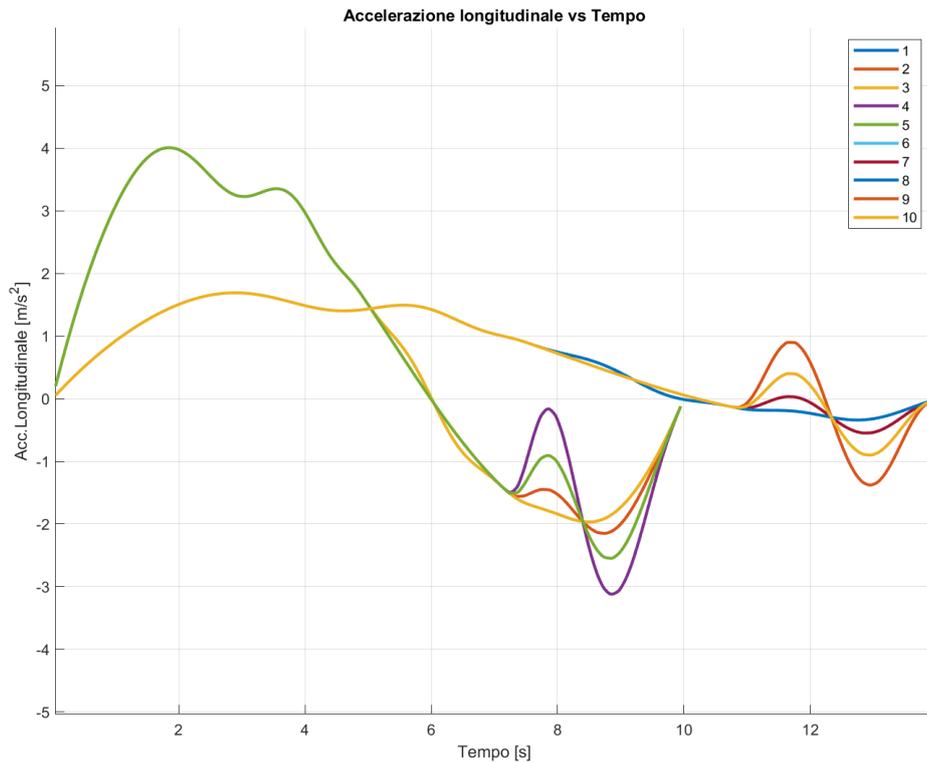


Figura 55 - Accelerazione longitudinale vs Tempo

Se si ritenesse opportuno contenere le accelerazioni longitudinali del veicolo si dovrebbero scegliere tempi di percorrenza più elevati, viceversa se si richiedesse prestazioni elevate al veicolo, selezionare le traiettorie a tempo ridotto.

Terminate le analisi possibili su questo tipo di planner, sarebbe interessante esplorare un altro metodo di definizione dei tracciati tramite polinomi di terzo grado in funzione della variabile curvilinea, come è stato fatto da [26]. L'obiettivo della presente tesi, però, è quello di utilizzare le coordinate di riferimento della traiettoria generata per farla eseguire a un veicolo CarMaker tramite una logica di path tracking discussa nei capitoli 3 e 4.

Le coordinate della traiettoria di riferimento ottenuta grazie al path planner vengono dunque esportate nel modello Simulink, sviluppato nel prossimo capitolo, per testare l'altra logica fondamentale nei veicoli autonomi che è il path tracking. Tali coordinate verranno passate come riferimenti sotto forma di matrici con righe contenenti:

$$[X_{ref}, Y_{ref}, \psi_{ref}, \rho_{ref}, S_{ref}, vx_{ref}]$$

La velocità longitudinale vx_{ref} verrà generata anche in altri modi, che verranno spiegati nel capitolo 4.4.

3. Progettazione Path tracking

Come proposto da [27], [28] e [29] è stato scelto di far eseguire la traiettoria, estratta dal path-planner ‘trajectoryOptimalFrenet’, al veicolo selezionato su IPG-CM tramite una logica di path tracking basata sull’LQR (‘Linear Quadratic Regulator’) che tende a minimizzare l’errore su una serie di scostamenti del percorso eseguito dal veicolo rispetto la traiettoria ideale generata dal path-planner.

Il metodo di path tracking che si vuole utilizzare è di tipo dinamico, quindi, tiene conto delle forze scambiate con il terreno durante le manovre e i limiti/vincoli imposti dal veicolo stesso. Viene però utilizzato un modello lineare di veicolo che non tiene conto delle non-linearità e delle discontinuità nel carattere di un veicolo reale. Si usa, quindi, un modello semplice di dinamica laterale, per poi andare a sviluppare il modello di path tracking LQR.

3.1 Modello dinamico di veicolo

È stato utilizzato il modello a bicicletta [30], ovvero il più semplice per descrivere la dinamica di un veicolo a quattro ruote. Facendo riferimento alla seguente figura:

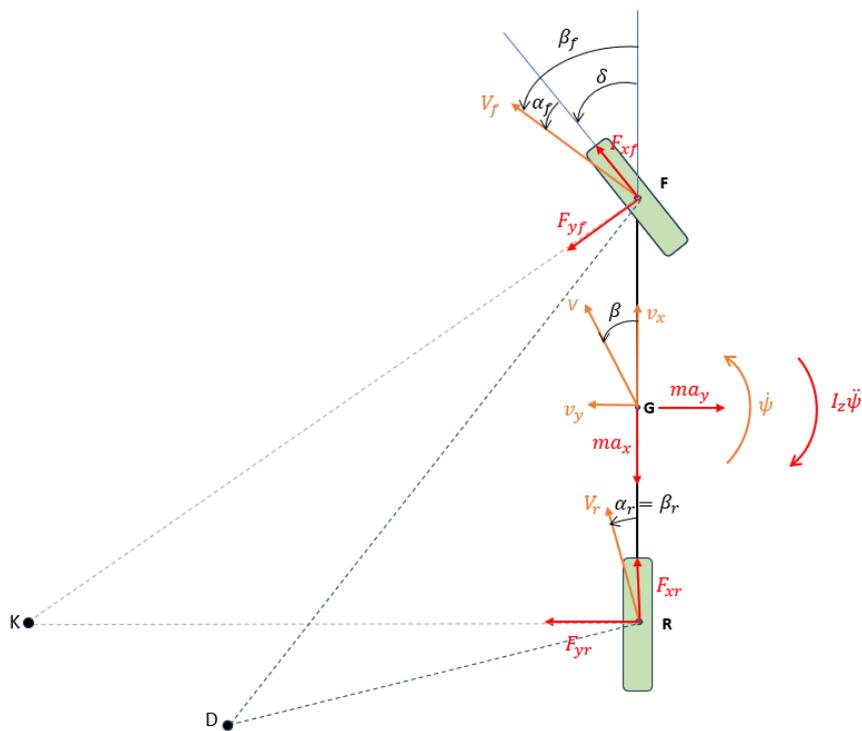


Figura 56 - Modello a bicicletta

In questa immagine $\dot{\psi}$ è la velocità di imbardata, $\ddot{\psi}$ corrisponde all'accelerazione angolare di imbardata del veicolo.

Vengono chiamati equivalentemente a e b i semipassi anteriori e posteriori, ovvero le distanze $\overline{FG} = a$ e $\overline{GR} = b$.

Il modello qui presentato tiene conto di entrambi gli assali (front e rear) motrici ($F_{xf} \neq 0, F_{xr} \neq 0$), e solo l'assale anteriore sterzante (δ). Si ricordi che il modello a bicicletta concentra in una ruota sola le due ruote di uno stesso assale; quindi, apparentemente trascura la dinamica legata al trasferimento di carico sul laterale, anche se ne tiene conto complessivamente per il calcolo delle rigidità in deriva dei due assali.

Si riportano una serie di relazioni relative alla dinamica di questo modello in forma non linearizzata:

L'equilibrio delle forze laterali agenti sul veicolo è:

$$F_{yf} \cos(\delta) + F_{xf} \sin(\delta) + F_{yr} = m(v_y + v_x \dot{\psi}) \quad (3.1)$$

L'equilibrio dei momenti imbardanti il veicolo è:

$$a(F_{yf} \cos(\delta) + F_{xf} \sin(\delta)) - bF_{yr} = I_z \ddot{\psi} \quad (3.2)$$

Dove I_z è il momento di inerzia imbardante.

Gli angoli di assetto della ruota anteriore e posteriore sono:

$$\beta_f = \tan^{-1} \left(\frac{v_y + a\dot{\psi}}{v_x} \right) \quad (3.3)$$

$$\beta_r = \tan^{-1} \left(\frac{v_y - b\dot{\psi}}{v_x} \right) \quad (3.4)$$

Gli angoli di scorrimento non linearizzati sono i seguenti:

$$\alpha_f = \tan^{-1} \left(\frac{v_y + a\dot{\psi}}{v_x} \right) - \delta = \beta_f - \delta \quad (3.5)$$

$$\alpha_r = \tan^{-1} \left(\frac{v_y - b\dot{\psi}}{v_x} \right) = \beta_r \quad (3.6)$$

Le forze laterali generate dall'assale anteriore e posteriore sono considerate linearmente proporzionali ai coefficienti di rigidità in deriva C_f e C_r (che dipendono dalla forza verticale premente lo pneumatico) e agli angoli di scorrimento:

$$F_{yf} = -C_f \alpha_f \quad (3.7)$$

$$F_{yr} = -C_r \alpha_r \quad (3.8)$$

Queste formulazioni valgono anche nel campo non lineare della tenuta degli pneumatici, ma i coefficienti di rigidità in deriva non sono delle costanti per tutti i valori degli angoli di scorrimento ma variano secondo, appunto, leggi non lineari rappresentate nei grafici ($F_y - \alpha$).

Assumendo la velocità longitudinale del baricentro del veicolo costante (in un dato istante temporale):

$$F_{xf} = F_{xr} = 0$$

Mettendo insieme le varie equazioni e risolvendo al fine di ricavare l'accelerazione laterale \dot{v}_y e l'accelerazione imbardante $\ddot{\psi}$:

$$\dot{v}_y = \frac{-C_f \left(\tan^{-1} \left(\frac{v_y + a\dot{\psi}}{v_x} \right) - \delta \right) \cos(\delta) - C_r \tan^{-1} \left(\frac{v_y - b\dot{\psi}}{v_x} \right)}{m} - v_x \dot{\psi} \quad (3.9)$$

$$\ddot{\psi} = \frac{-aC_f \left(\tan^{-1} \left(\frac{v_y + a\dot{\psi}}{v_x} \right) - \delta \right) \cos(\delta) - bC_r \tan^{-1} \left(\frac{v_y - b\dot{\psi}}{v_x} \right)}{I_z} \quad (3.10)$$

Linearizzando tale modello a bicicletta, quindi per angoli di scorrimento e angoli di sterzo relativamente piccoli:

$$\cos(\delta) \approx 1$$

$$\sin \delta \approx \delta$$

$$\tan^{-1} \left(\frac{v_y + a\dot{\psi}}{v_x} \right) \approx \frac{v_y + a\dot{\psi}}{v_x} = \beta + \frac{a\dot{\psi}}{v_x} \quad (3.11)$$

$$\tan^{-1} \left(\frac{v_y - b\dot{\psi}}{v_x} \right) \approx \frac{v_y - b\dot{\psi}}{v_x} = \beta - \frac{b\dot{\psi}}{v_x} \quad (3.12)$$

Pertanto, dopo aver aggregato i termini legati alla stessa variabile di interesse, le EQUAZIONI DEL MOTO, espressione di \dot{v}_y e di $\ddot{\psi}$ diventano:

$$\dot{v}_y = \frac{-(C_f + C_r)}{mv_x} v_y + \left[\frac{(bC_r - aC_f)}{mv_x} - v_x \right] \dot{\psi} + \frac{C_f}{m} \delta \quad (3.13)$$

$$\ddot{\psi} = \frac{bC_r - aC_f}{I_z v_x} v_y + \frac{-(a^2 C_f + b^2 C_r)}{I_z v_x} \dot{\psi} + \frac{aC_f}{I_z} \delta \quad (3.14)$$

Che riscritto nella forma dello SPAZIO DEGLI STATI (state space):

$$(\dot{x}) = [A](x) + [B](u) \quad (3.15)$$

Dove:

$$(x) = \begin{pmatrix} v_y \\ \dot{\psi} \end{pmatrix} \text{ vettore degli stati}$$

$$(u) = \delta \text{ 'vettore' degli input}$$

$$(\dot{x}) = \begin{pmatrix} \dot{v}_y \\ \ddot{\psi} \end{pmatrix} \text{ vettore derivate degli stati}$$

$$\begin{bmatrix} \dot{v}_y \\ \ddot{\psi} \end{bmatrix} = \begin{bmatrix} \frac{-(C_f + C_r)}{mv_x} & \frac{bC_r - aC_f}{mv_x} - v_x \\ \frac{bC_r - aC_f}{I_z v_x} & \frac{-(a^2 C_f + b^2 C_r)}{I_z v_x} \end{bmatrix} \begin{bmatrix} v_y \\ \dot{\psi} \end{bmatrix} + \begin{bmatrix} \frac{C_f}{m} \\ \frac{aC_f}{I_z} \end{bmatrix} \delta \quad (3.16)$$

Equivalentemente tale sistema si può riscrivere esplicitando l'angolo di assetto baricentrico (β) del veicolo come variabile di stato al posto della velocità di scorrimento trasversale baricentrica (v_y), ricordando che per angoli piccoli si può fare la seguente approssimazione:

$$\beta = \tan^{-1}\left(\frac{v_y}{v_x}\right) \approx \frac{v_y}{v_x}$$

$$\dot{\beta} \approx \frac{\dot{v}_y}{v_x}$$

Si ricorda che per angoli piccoli (β) la velocità di movimento del baricentro longitudinale può essere scambiata con:

$$v_x \approx V$$

Dove V è la velocità assoluta del baricentro del veicolo:

$$V = \sqrt{v_x^2 + v_y^2} \quad (3.17)$$

Pertanto, lo state space risultante, considerando come vettori degli stati e degli input i seguenti:

$$(x) = \begin{pmatrix} \beta \\ \dot{\psi} \end{pmatrix} \text{ vettore degli stati}$$

$$(u) = \delta \text{ 'vettore' degli input}$$

$$(\dot{x}) = \begin{pmatrix} \dot{\beta} \\ \ddot{\psi} \end{pmatrix} \text{ vettore derivate degli stati}$$

Sarà:

$$\begin{bmatrix} \dot{\beta} \\ \ddot{\psi} \end{bmatrix} = \begin{bmatrix} \frac{-(C_f + C_r)}{mV} & \frac{bC_r - aC_f - mV^2}{mV^2} \\ \frac{bC_r - aC_f}{I_z} & \frac{-(a^2 C_f + b^2 C_r)}{I_z V} \end{bmatrix} \begin{bmatrix} \beta \\ \dot{\psi} \end{bmatrix} + \begin{bmatrix} \frac{C_f}{m} \\ \frac{aC_f}{I_z} \end{bmatrix} \delta \quad (3.18)$$

Volendo completare la scrittura nello SPAZIO DEGLI STATI secondo la forma classica:

$$\begin{cases} \dot{x} = Ax + Bu \\ y = Cx + Du \end{cases} \quad (3.19)$$

In cui i singoli vettori di interesse sono:

$$(\dot{x}) = \begin{pmatrix} \dot{\beta} \\ \ddot{\psi} \end{pmatrix} \text{ vettore derivate degli stati}$$

$$(x) = \begin{pmatrix} \beta \\ \dot{\psi} \end{pmatrix} \text{ vettore degli stati}$$

$(u) = \delta$ 'vettore' degli input

$$(y) = \begin{pmatrix} \beta \\ \dot{\psi} \\ \rho \\ \alpha_f \\ \alpha_r \\ a_y \end{pmatrix} \text{ vettore degli output}$$

Al fine di ottenere le quattro matrici rappresentanti lo spazio degli stati: A, B, C, D, si ricordano le seguenti relazioni esprimendo gli output di interesse: Curvatura, Angolo di deriva anteriore, Angolo di deriva posteriore, Accelerazione laterale:

$$\rho = \frac{a_y}{V^2} = \left(\frac{-C_f - C_r}{mV^2} \right) \beta + \left(\frac{-aC_f + bC_r}{mV^3} \right) \dot{\psi} + \left(\frac{C_f}{mV^2} \right) \delta \quad (3.20)$$

$$\alpha_f = \delta - \beta - \frac{\dot{\psi}a}{V} \quad (3.21)$$

$$\alpha_r = -\beta + \frac{\dot{\psi}b}{V} \quad (3.22)$$

$$a_y = V^2 \rho = \left(\frac{-C_f - C_r}{m} \right) \beta + \left(\frac{-aC_f + bC_r}{mV} \right) \dot{\psi} + \left(\frac{C_f}{m} \right) \delta \quad (3.23)$$

Queste equazioni inserite all'interno del sistema appartenente allo state space:

$$(y) = [C](x) + [D](u) \quad (3.24)$$

Portano ad ottenere:

$$\begin{bmatrix} \beta \\ \dot{\psi} \\ \rho \\ \alpha_f \\ \alpha_r \\ a_y \end{bmatrix} = \begin{bmatrix} 1 & 0 \\ 0 & 1 \\ \left(\frac{-C_f - C_r}{mV^2} \right) & \left(\frac{-aC_f + bC_r}{mV^3} \right) \\ -1 & -\frac{a}{V} \\ -1 & \frac{b}{V} \\ \left(\frac{-C_f - C_r}{m} \right) & \left(\frac{-aC_f + bC_r}{mV} \right) \end{bmatrix} \begin{bmatrix} \beta \\ \dot{\psi} \end{bmatrix} + \begin{bmatrix} 0 \\ 0 \\ \left(\frac{C_f}{mV^2} \right) \\ 1 \\ 0 \\ \left(\frac{C_f}{m} \right) \end{bmatrix} \delta \quad (3.25)$$

Si riassumono, quindi, tutte e quattro le MATRICI caratterizzanti lo spazio degli stati, che serviranno nella successiva implementazione del modello su Matlab. I risultati di tale modello (quindi gli output (y)) verranno successivamente confrontati, in alcune manovre di prova classiche, con i risultati restituiti dal veicolo di simulazione su IPG-CM per valutare le differenze tra questo modello con meno gradi di libertà rispetto a quello più completo utilizzato dal programma.

$$[A] = \begin{bmatrix} \frac{-(C_f + C_r)}{mV} & \frac{bC_r - aC_f - mV^2}{mV^2} \\ \frac{bC_r - aC_f}{I_z} & \frac{-(a^2C_f + b^2C_r)}{I_zV} \end{bmatrix}$$

$$[B] = \begin{bmatrix} C_f \\ mV \\ aC_f \\ I_z \end{bmatrix}$$

$$[C] = \begin{bmatrix} 1 & 0 \\ 0 & 1 \\ \left(\frac{-C_f - C_r}{mV^2}\right) & \left(\frac{-aC_f + bC_r}{mV^3}\right) \\ -1 & -\frac{a}{V} \\ -1 & \frac{b}{V} \\ \left(\frac{-C_f - C_r}{m}\right) & \left(\frac{-aC_f + bC_r}{mV}\right) \end{bmatrix}$$

$$[D] = \begin{bmatrix} 0 \\ 0 \\ \left(\frac{C_f}{mV^2}\right) \\ 1 \\ 0 \\ \left(\frac{C_f}{m}\right) \end{bmatrix}$$

3.2 Parametri fisici del modello di veicolo impiegato

I parametri fisici del veicolo di interesse in questa trattazione sono quelli relativi al veicolo usato nella simulazione su IPG-CM [31]: Tesla Model-S.

Si è scelta questa vettura perché la trasmissione di un veicolo elettrico risulta più semplice di quella di un veicolo a combustione interna, poiché non presenta cambi di velocità; quindi, permette di trascurare l'effetto delle marce durante le successive simulazioni.

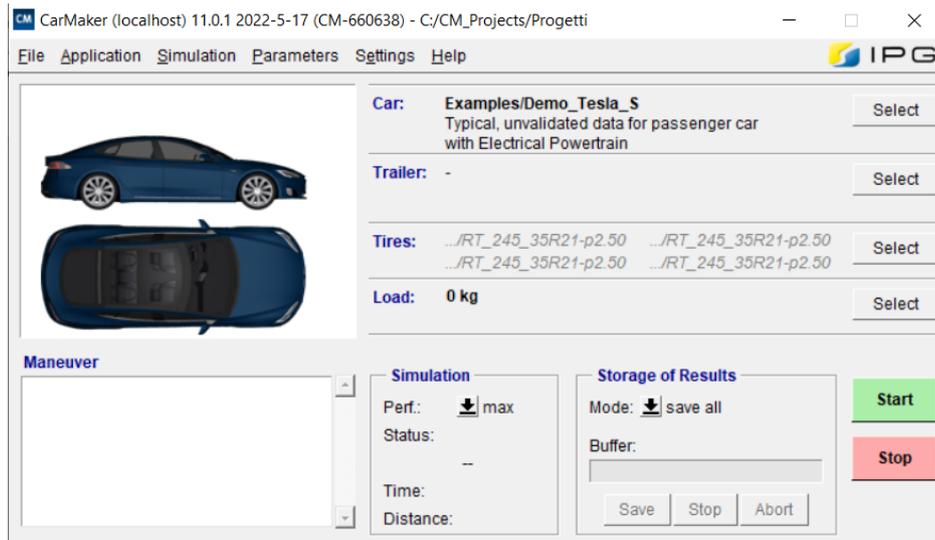


Figura 57 - GUI CarMaker con Tesla-Model-S

Si è, però, deciso di considerare un veicolo idealmente infinitamente rigido per trascurare gli effetti dinamici legati alla natura elastica dei veicoli reali. Quindi, sono state modificate parzialmente le proprietà del modello di veicolo all'interno della seguente schermata, cambiando la caratteristica riquadrata in rosso con la proprietà 'Rigid':

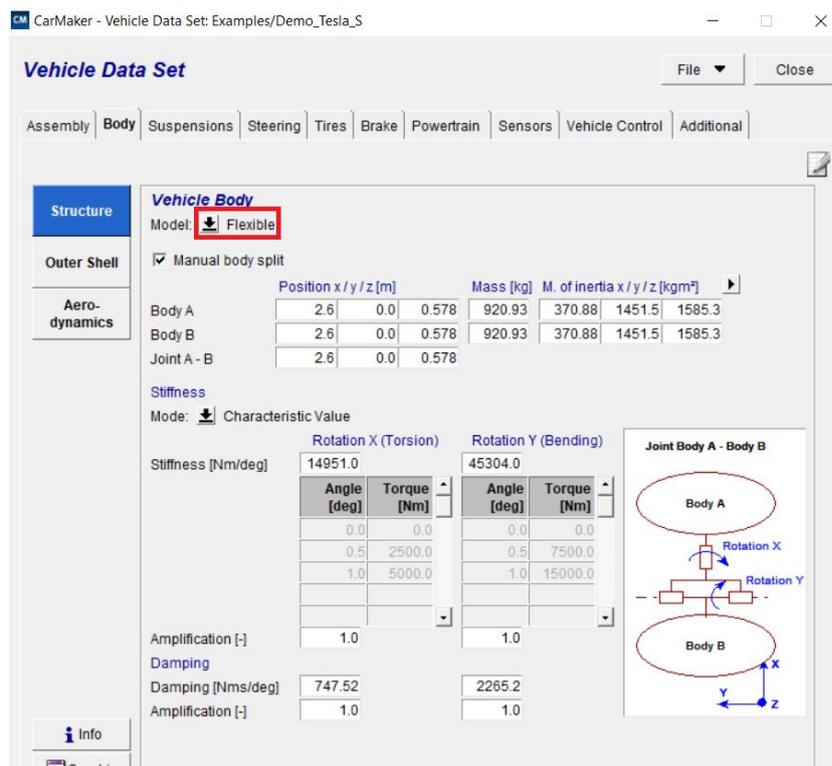


Figura 58 - Caratteristiche di default veicolo

Andando a selezionare la proprietà di rigidità del veicolo, le proprietà inerziali sono solo più presenti accanto alla dicitura 'Body A', poiché, se il veicolo è considerato come idealmente infinitamente rigido, tutta la massa è come fosse concentrata in un punto caratterizzato dalle proprietà inerziali riassunte accanto a 'Body A'.

Tali proprietà devono, però, rispecchiare il più possibile quelle iniziali della Tesla, riportate qui di seguito (ricavabili selezionando la casella 'info').

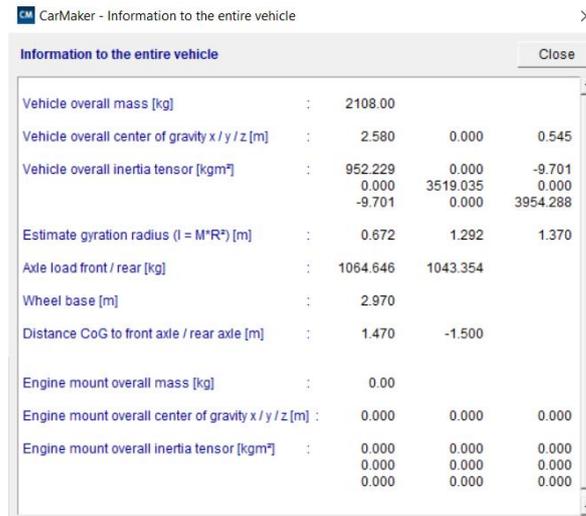


Figura 59 - Informazioni veicolo

Si procede, pertanto, a creare un nuovo modello di veicolo RIGIDO che presenti, circa, le stesse proprietà inerziali, inserendole manualmente come presentato qui di seguito in queste schermate:

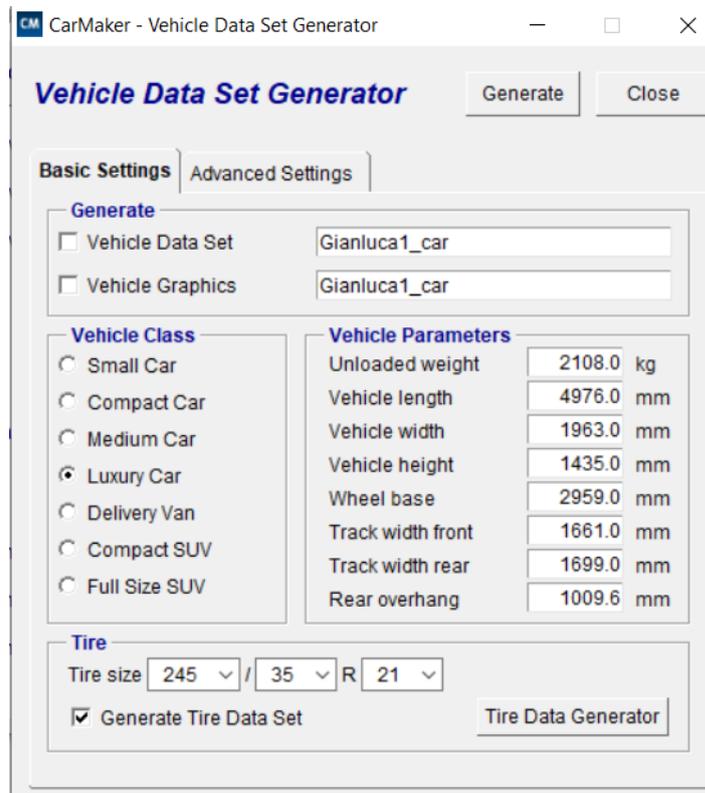


Figura 60 - Riconfigurazione nuovo modello veicolo

Questi dati sono stati mantenuti identici alla Tesla-S.

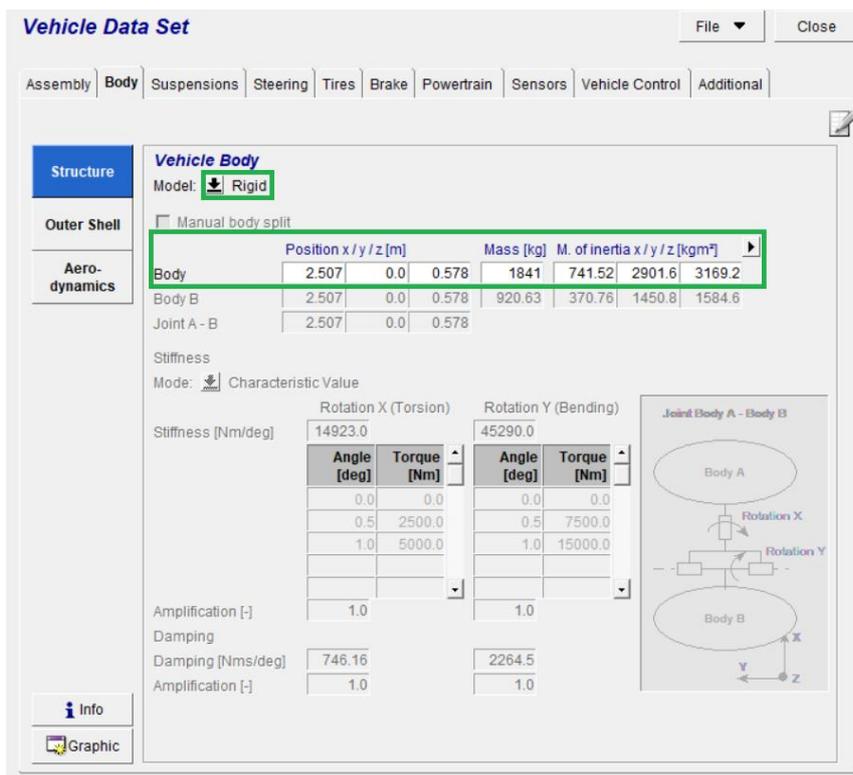


Figura 61 - Modello di veicolo reso rigido con caratteristiche

Come si può notare il nuovo modello di veicolo, reso RIGIDO, presenta circa le stesse inerzie della Tesla-S. Si riportano, quindi, le caratteristiche presenti all'interno delle 'info' del nuovo veicolo così modificato:

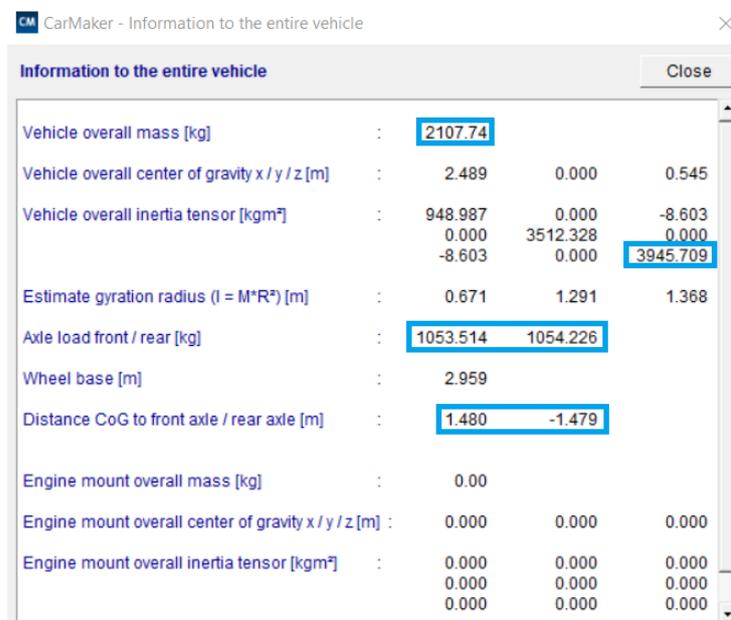


Figura 62 - Informazioni nuovo modello veicolo

Si possono notare solo leggere variazioni rispetto al modello di Tesla-S, ma questo non inficia la validità della trattazione che seguirà, relativa al confronto del modello a bicicletta con il modello di veicolo a più gradi di libertà su IPG-CM reso rigido. Questo

perché, nel modello a bicicletta si utilizzeranno esattamente questi parametri fisici e non quelli del modello di Tesla originale.

Quindi, i parametri da inserire nel modello dinamico a bicicletta (caratterizzato dalle quattro matrici A, B, C, D) sono:

- Massa totale del veicolo: m
- Semipassi anteriore e posteriore: a, b
- Momento di inerzia di imbardata: I_z
- Coefficienti di rigidezza in deriva assale anteriore e posteriore: C_f, C_r

3.2.1 Massa totale

Si può facilmente vedere come la massa totale del veicolo di simulazione è:

$$m = 2107,74 \text{ kg}$$

Tale massa ovviamente tiene conto di tutte le masse agenti sui quattro punti di appoggio del veicolo:

$$m = m_{fr} + m_{fl} + m_{rr} + m_{rl} \quad (3.26)$$

Concentrando la massa agente sul singolo assale anteriore (front) e posteriore (rear), come avviene nel modello a bicicletta:

$$m_f = m_{fr} + m_{fl} = 1053,514 \text{ kg} \quad (3.27)$$

$$m_r = m_{rr} + m_{rl} = 1054,226 \text{ kg} \quad (3.28)$$

3.2.2 Semipassi anteriore e posteriore

Come è noto, dalla User Guide di IPG-CM il sistema di riferimento locale del veicolo (Fr1) ha origine nell'estremità posteriore più remota del veicolo, come segnalato nella seguente immagine:

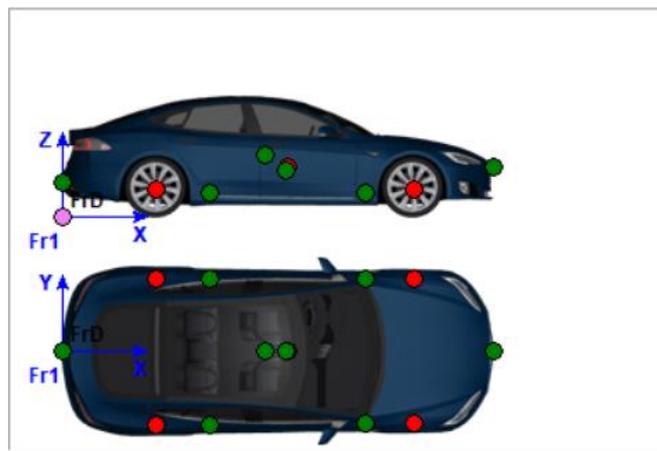


Figura 63 - Sistemi di riferimento veicolo CarMaker

Il passo complessivo del veicolo si può ricavare dalle coordinate dell'assale anteriore e posteriore nel sistema di riferimento appena definito.

CarMaker - Vehicle Data Set: TeslaS.car

Vehicle Data Set File ▾

Assembly | Body | Suspensions | Steering | Tires | Brake | Powertrain | Sensors | Vehicle Control | Additional

Config-uration		Position x / y / z [m]			Mass [kg]	M. of inertia x / y / z [kgm ²]		
Engine Mount	Wheel carrier FL	3.969	0.826	0.321	27.947	0.447	0.447	0.447
	Wheel carrier FR	3.969	-0.826	0.321	27.947	0.447	0.447	0.447
	Wheel carrier RL	1.01	0.841	0.321	43.97	0.57	0.57	0.57
	Wheel carrier RR	1.01	-0.841	0.321	43.97	0.57	0.57	0.57
Body	Wheel FL	3.969	0.826	0.321	33.05	1.139	2.279	1.139
Chassis	Wheel FR	3.969	-0.826	0.321	33.05	1.139	2.279	1.139
	Wheel RL	1.01	0.841	0.321	28.402	1.079	2.158	1.079
	Wheel RR	1.01	-0.841	0.321	28.402	1.079	2.158	1.079

Figura 64 - Posizione assali veicolo

Come si può notare il passo complessivo risulterà essere pari a:

$$L = x_{front,wheel} - x_{rear,wheel} = 3,969 - 1,01 = 2,959 \text{ m} \quad (3.29)$$

Questo dato rispecchia il valore fornito anche dalla schermata 'info' riportato precedentemente.

I valori dei semipassi anteriori e posteriori si possono ottenere sia come:

$$a = L \left(1 - \frac{m_f}{m} \right) \quad (3.30)$$

$$b = L \left(1 - \frac{m_r}{m} \right) \quad (3.31)$$

Dove la m_f e la m_r sono rispettivamente la massa concentrata (idealmente) sull'assale anteriore e su quello posteriore nel modello a bicicletta.

Su IPG-CM si trovano più facilmente i dati relativi ai semipassi anteriore e posteriore nelle 'info':

$$a = 1,480 \text{ m}$$

$$b = 1,479 \text{ m}$$

Questi dati corrispondono ai valori che si otterrebbero utilizzando le formule che utilizzano le masse per il calcolo dei semipassi.

3.2.3 Momento di inerzia di imbardata

Il momento di inerzia attorno l'asse z del veicolo (imbardata) si può ricavare nel modello a bicicletta considerando il veicolo come due masse puntiformi (m_f e m_r) collegate da una trave rigida priva di massa di lunghezza il passo del veicolo (L).

$$I_z = m_f a^2 + m_r b^2 \quad (3.32)$$

Ma questa formula porterebbe a una sovrastima del momento di inerzia di imbardata, perché la massa di un veicolo reale è più distribuito rispetto a due masse puntiformi distanti il passo del veicolo.

Nel presente caso il momento di inerzia imbardante si può ottenere dalle info del veicolo:

$$I_z = 3945,709 \text{ kgm}^2$$

3.2.4 Coefficienti di rigidezza in deriva

I coefficienti di rigidezza in deriva vanno identificati da schede di dati inerenti al veicolo con i corrispettivi pneumatici. Come è ben noto, tali coefficienti non sono costanti al variare della forza laterale sviluppata tra pneumatico ed asfalto e della forza verticale agente sullo pneumatico.

Il rapporto tra le forze laterali e gli angoli di deriva è il coefficiente di rigidezza in deriva secondo la formula generale (che esprime solo i moduli):

$$F_y = C_\alpha \alpha \quad (3.33)$$

Come già detto C_α rimane costante solo nel tratto lineare della curva ($F_y - \alpha$), mentre tende a scendere gradualmente quando ci si avvicina alla saturazione della tenuta degli pneumatici.

Si riporta la curva ($F_y - \alpha$) per una prova di 'RAMP STEER', eseguita dal veicolo su IPG-CM, per tutti i quattro pneumatici:

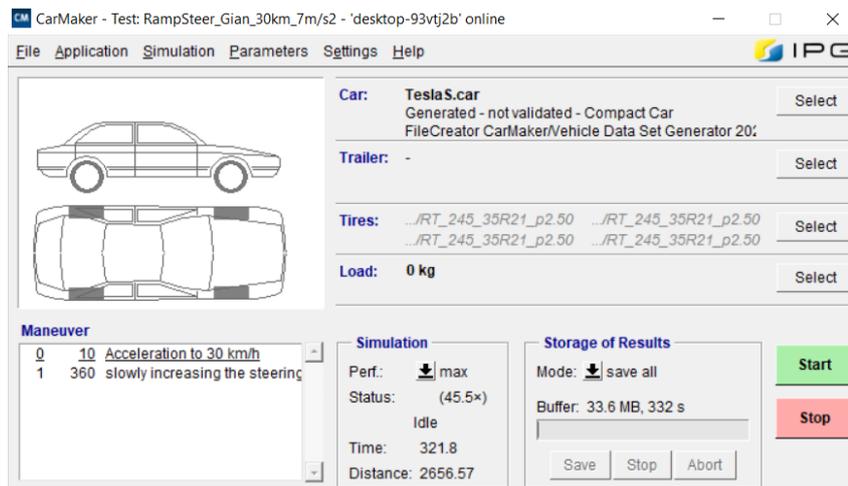


Figura 65 - Impostazione manovra Ramp Steer

Le manovre (maneuver) eseguite durante questa prova di RAMP STEER sono:

- 1) Accelerazione da 0 a 30km/h in 10 secondi
- 2) Lento aumento dell'angolo di sterzo a velocità costante (30 km/h) fino al raggiungimento di un'accelerazione laterale $a_y \geq \frac{7m}{s^2}$ (360s)

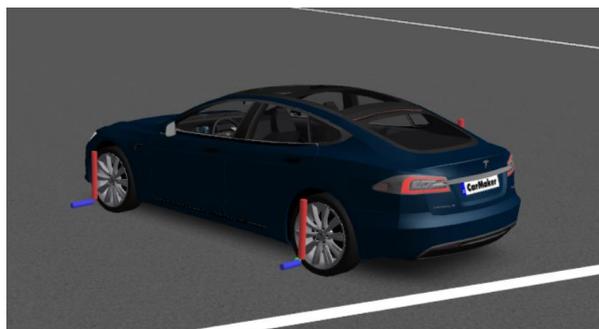


Figura 66 - Istantaneo veicolo durante simulazione

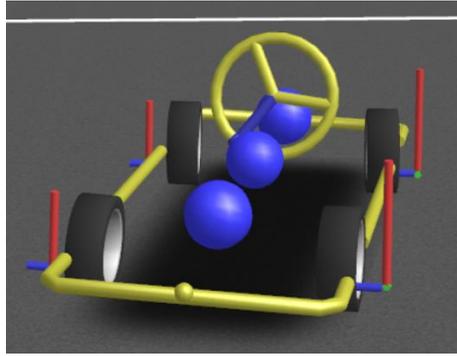


Figura 67 - Istantanea veicolo reso rigido durante simulazione

Prima di presentare i grafici relativi alla forza laterale generata sullo pneumatico, al variare degli angoli di scorrimento laterale, si devono verificare le seguenti condizioni che confermano la validità della prova di ramp steer che è stata eseguita:

- Si verifica che la velocità mantenuta dal veicolo durante tutta la fase di ramp steer sia rimasta costante e pari a: $V = \frac{30km}{h} = 8,333 \frac{m}{s}$
Si può notare come questa condizione venga rispettata lungo tutto il tempo di prova:

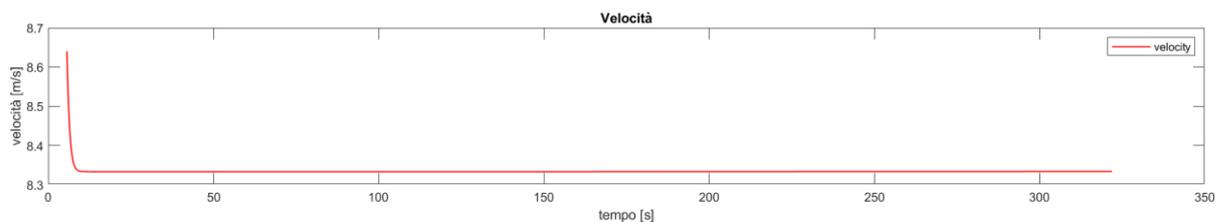


Figura 68 – Ramp steer a velocità costante

- Si deve verificare il raggiungimento di un'accelerazione laterale $a_y \geq \frac{7m}{s^2}$
Tale condizione come si può notare è verificata:

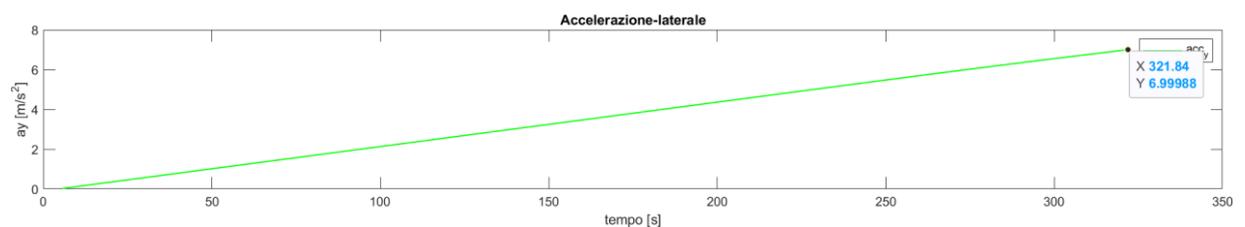


Figura 69 - Accelerazione laterale manovra RAMP STEER

Si puntualizza che solo i dati relativi alla fase di sterzata sono stati riportati nei risultati, pertanto, solo i valori legati a angoli volante $\delta_{SW} \geq 1^\circ = 0,017453 \text{ rad}$, per eliminare tutti i disturbi grafici che vengono generati nelle prime fasi della manovra:

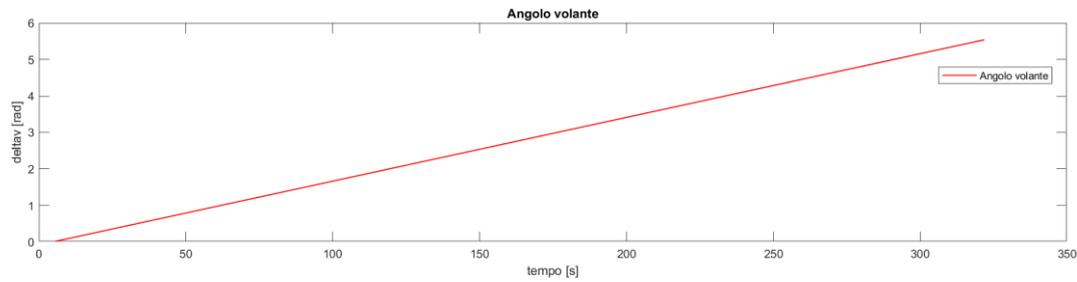


Figura 70 - Angolo volante manovra RAMP STEER

Si riportano, infine, le curve $(F_y - \alpha)$ su cui si vanno a identificare i coefficienti di rigidità in deriva tramite un'interpolazione lineare nel primo tratto di curva relativo alla fase lineare di aderenza dello pneumatico:

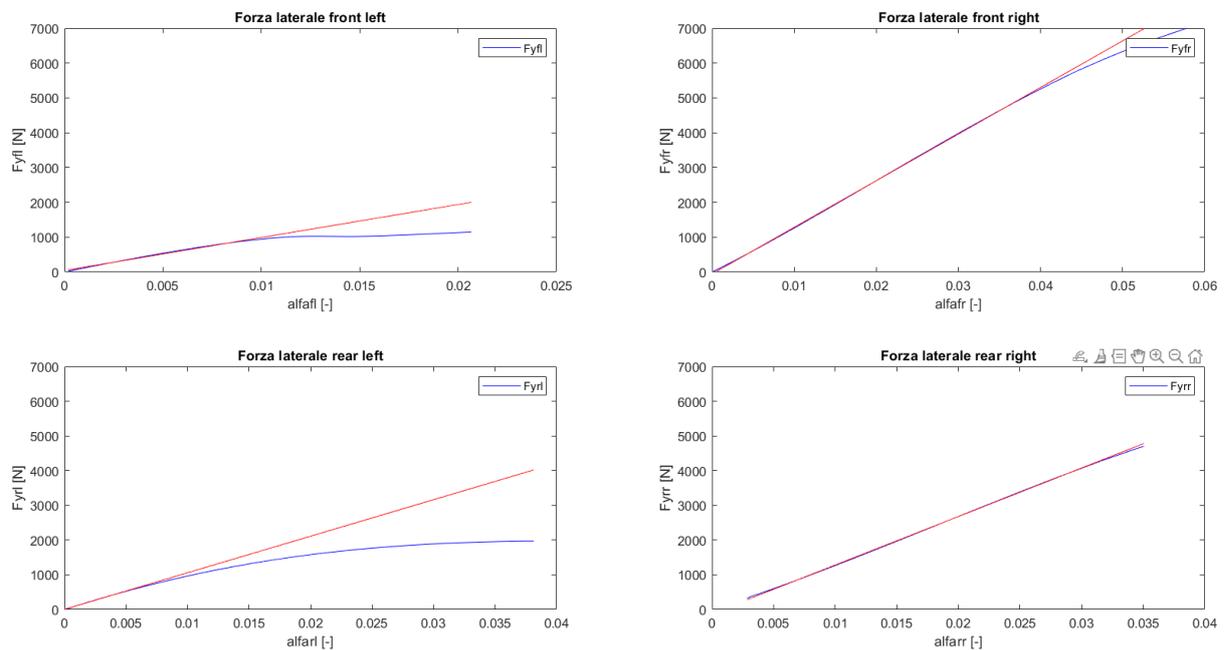


Figura 71 - Forze laterali - Angoli di scorrimento - Manovra RAMP STEER

L'estensione lungo α del tratto lineare differisce per i diversi quattro pneumatici; perché, ad esempio, quelli interno curva (in questo caso le gomme di sinistra) sono scaricate a causa dei trasferimenti di carico dal lato sinistro di veicolo al lato destro, pertanto, avendo una forza premente lo pneumatico inferiore, presentano un passaggio alla non linearità per angoli di scorrimento inferiori a quelli di destra.

La pendenza delle rette di interpolazione viene, quindi, presa come coefficiente di rigidità in deriva del singolo assale, che si può considerare costante, almeno nel tratto lineare della curva; quindi, per basse forze e accelerazioni laterali sviluppate.

I valori così misurati, tramite la funzione 'polyfit' di Matlab, delle rigidità in deriva risultano essere:

$C_{\alpha fl}$	94711 N/rad
$C_{\alpha fr}$	133884 N/rad
$C_{\alpha rl}$	105214 N/rad
$C_{\alpha rr}$	139694 N/rad

Nel modello a bicicletta le rigidità in deriva di due ruote su uno stesso assale vengono ‘condensate’ sommandole. Pertanto, le rigidità in deriva sull’assale front e su quello rear del veicolo sono:

$$C_f = 228595 \frac{N}{rad}$$

$$C_r = 244908 \frac{N}{rad}$$

3.2.5 Angoli di sterzo e angolo volante in funzione dell’accelerazione laterale

Si riporta, infine, le caratteristiche (angolo di sterzo – accelerazione laterale): $(\delta - a_y)$ e (angolo volante – accelerazione laterale): $(\delta_{SW} - a_y)$ da cui si possono estrarre gli angoli di sterzo sull’assale anteriore necessari per ottenere delle accelerazioni laterali di 2 e 7 m/s^2 nelle prove di ‘Step-Steer’ (colpo di sterzo) che verranno presentate nel prossimo capitolo:

Curva $(\delta - a_y)$

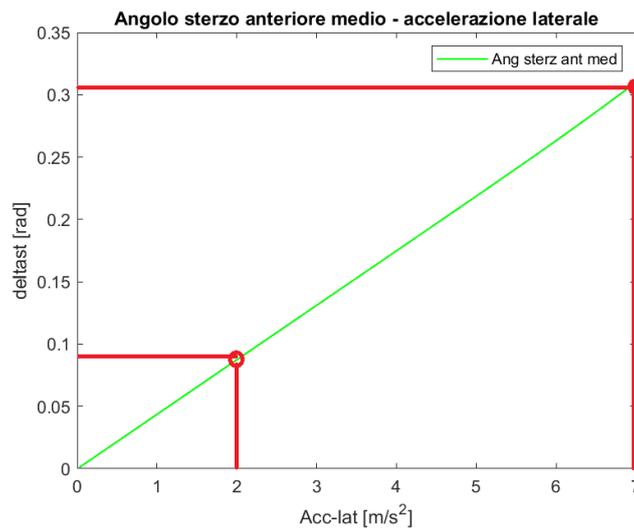


Figura 72 - Angolo sterzo ruote - Accelerazione laterale - RAMP STEER

Curva $(\delta_{SW} - a_y)$

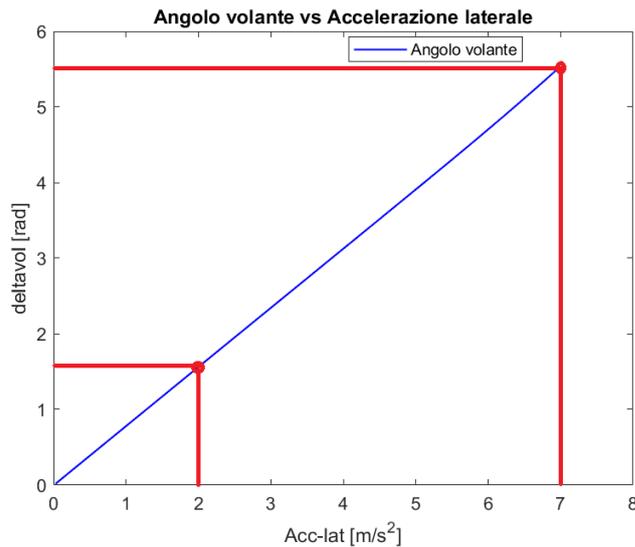


Figura 73 - Angolo volante - Accelerazione laterale - RAMP STEER

Si estraggono i seguenti valori di angolo di sterzo (al contatto terra) e angolo volante per ottenere le accelerazioni laterali rispettivamente di:

- $a_y = 2 \frac{m}{s^2} \rightarrow \begin{cases} \delta = 0,087 \text{ rad} = 4,985^\circ \\ \delta_{SW} = 1,56 \text{ rad} = 89,38^\circ \end{cases}$
- $a_y = 7 \frac{m}{s^2} \rightarrow \begin{cases} \delta = 0,31 \text{ rad} = 17,762^\circ \\ \delta_{SW} = 5,53 \text{ rad} = 316,85^\circ \end{cases}$

Da questi dati si evince come il rapporto di trasmissione allo sterzo sia:

$$R = \frac{\delta_{SW}}{\delta} \approx 17,88 \approx 18 \quad (3.34)$$

3.3 Confronto modello a bicicletta e modello di veicolo IPG-CM

Utilizzando i parametri fisici, determinati nel capitolo precedente, sono state svolte in parallelo due simulazioni di manovre di **COLPO DI STERZO**:

- Simulazione su IPG-CM di una manovra di colpo di sterzo a 30 km/h per il raggiungimento di $a_y = 2$ e 7 m/s^2 , tramite il modello di veicolo creato e descritto nel capitolo precedente su IPG-CM.
- Simulazione su Matlab delle stesse due manovre di colpo di sterzo, ma con il semplice modello a bicicletta a due stati $((\beta, \psi))$.

La prova di COLPO DI STERZO prevede le seguenti manovre generali:

- Aumento della velocità del veicolo da 0 a 30 km/h
- Brusca sterzata, quindi rapida rotazione dell'angolo volante in un tempo molto ridotto ($\dot{\delta}_{SW} \approx 400^\circ/s$)

- Mantenimento dell'angolo volante costante per il tempo necessario al raggiungimento di una condizione stazionaria (priva di oscillazione degli output)

3.3.1 Manovra su IPG-CM

Si esegue il colpo di sterzo sia per ottenere 2 m/s^2 di accelerazione laterale, che per il caso di 7 m/s^2 .

Gli output del sistema verranno riportati sugli stessi grafici in cui saranno presentati i risultati generati dal modello a bicicletta implementato su Matlab.

L'input del sistema, che viene passato nella casella di dialogo 'Maneuver' di IPG-CarMaker è l'angolo volante massimo che si vuole imporre nella manovra di colpo di sterzo. Il suo valore è il δ_{SW} che è stato ricavato in precedenza tramite il 'Ramp-Steer' e che viene riportato qui per comodità:

- $a_y = 2 \frac{\text{m}}{\text{s}^2} \rightarrow \delta_{SW} = 1,56 \text{ rad} = 89,38^\circ \approx 90^\circ$
- $a_y = 7 \frac{\text{m}}{\text{s}^2} \rightarrow \delta_{SW} = 5,53 \text{ rad} = 316,85^\circ \approx 317^\circ$

Questi sono i valori di angolo volante, che si devono applicare al nostro veicolo, tramite il Driver di CarMaker, nel colpo di sterzo, per ottenere, in condizioni stazionarie, un'accelerazione laterale rispettivamente di 2 e 7 m/s^2 , alla velocità di crociera di $30 \frac{\text{km}}{\text{h}} \approx 8,33 \frac{\text{m}}{\text{s}}$. Come già accennato nel precedente capitolo, è importante mantenere tale velocità costante durante tutta la prova, per avere un valido confronto tra questo modello di veicolo e il modello a bicicletta che utilizza, per il calcolo degli output, delle matrici (A, B, C, D) contenenti termini caratterizzati dalla velocità del veicolo (V) costante.

La prova che è stata impostata è la seguente, con l'utilizzo del modello di veicolo generato nel capitolo precedente.

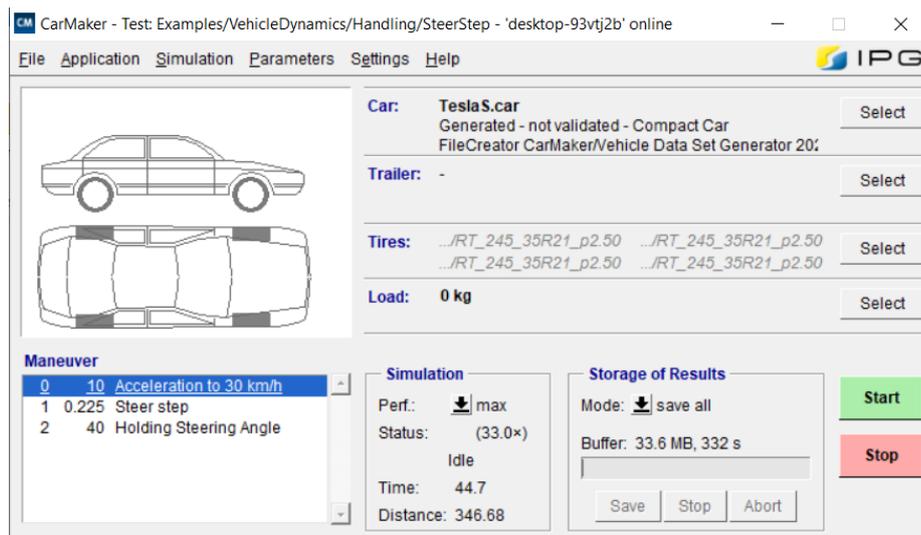


Figura 74 - Impostazione COLPO DI STERZO- 2 m/s^2

Si riportano, in ordine, le curve che mostrano la corretta attuazione della manovra, di:

- Velocità in stazionario costante di $30 \frac{\text{km}}{\text{h}} \approx 8,33 \frac{\text{m}}{\text{s}}$

- Accelerazione laterale in stazionario di $a_y = 2 \frac{m}{s^2}$
- Angolo volante di $\delta_{SW} \approx 1,56 \text{ rad}$
- Angolo di sterzo assale anteriore medio $\delta \approx 0,087 \text{ rad}$

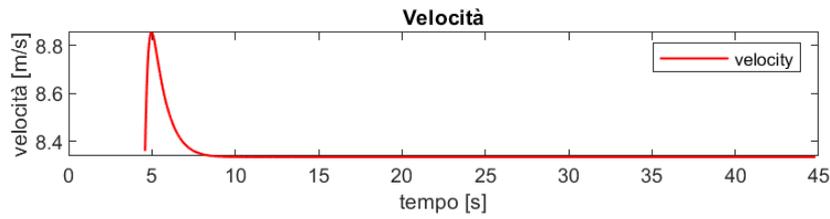


Figura 75 - Velocità COLPO STERZO $2m/s^2$

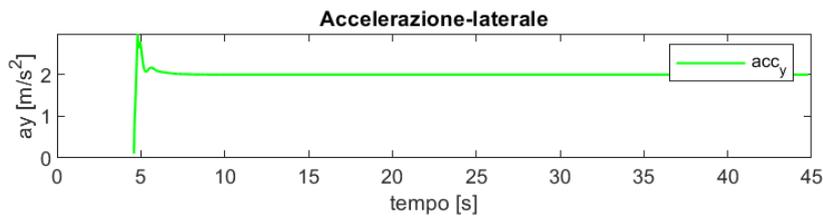


Figura 76 - Accelerazione laterale COLPO STERZO $2m/s^2$

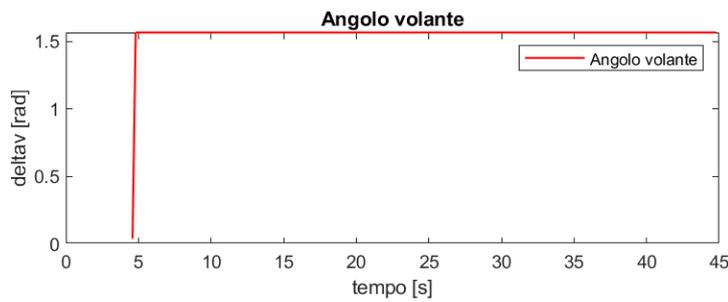


Figura 77 - Angolo volante COLPO STERZO $2 m/s^2$

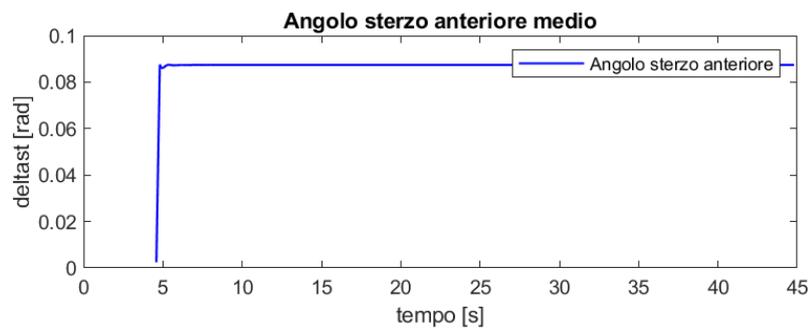


Figura 78 - Angolo di sterzo COLPO DI STERZO $2m/s^2$

È evidente come le manovre siano state eseguite correttamente da IPG-CM.

Si riportano, allo stesso modo, le curve relative alla manovra con accelerazione laterale più elevata:

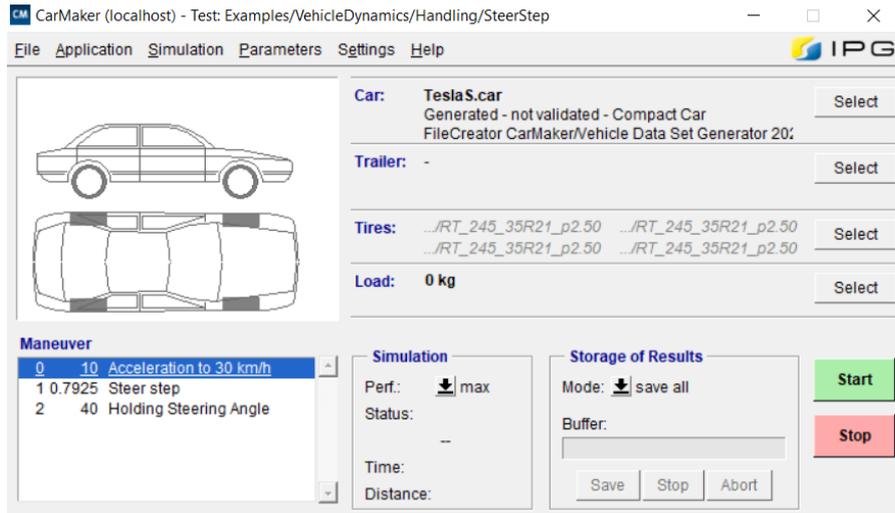


Figura 79 - Impostazione COLPO DI STERZO-7m/s²

- Velocità in stazionario costante di $30 \frac{km}{h} \approx 8,33 \frac{m}{s}$
- Accelerazione laterale in stazionario di $a_y = 7 \frac{m}{s^2}$
- Angolo volante di $\delta_{SW} \approx 5,53 \text{ rad}$
- Angolo di sterzo assale anteriore medio $\delta \approx 0,31 \text{ rad}$

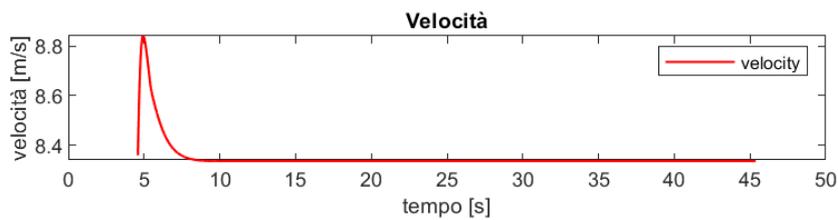


Figura 80 - Velocità COLPO DI STERZO 7m/s²

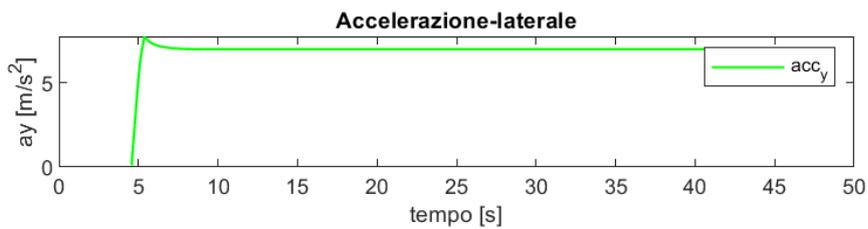


Figura 81 - Accelerazione laterale COLPO DI STERZO 7m/s²

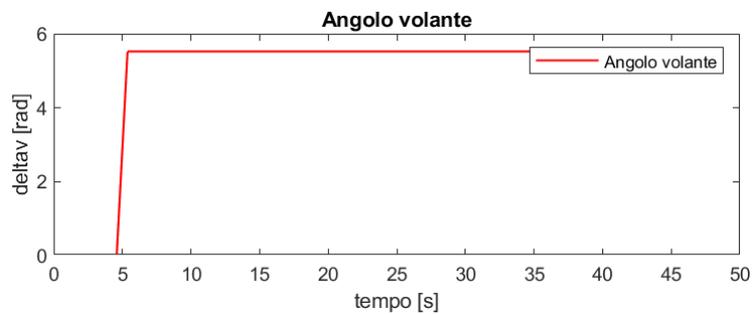


Figura 82 - Angolo volante COLPO STERZO 7m/s²

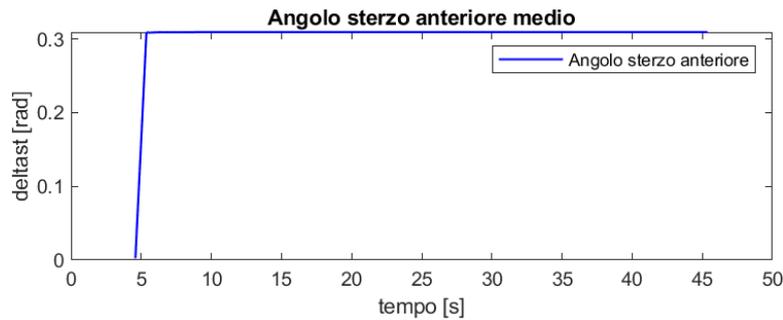


Figura 83 - Angolo sterzo COLPO DI STERZO $7m/s^2$

Anche in questo caso le manovre sono state eseguite correttamente.

3.3.2 Manovra su Matlab

La manovra su Matlab viene eseguita, comunicando lo stesso angolo di sterzo $\delta = \delta(t)$ che viene eseguito dal veicolo di IPG-CM. Tale angolo di sterzo (alle ruote) viene calcolato come la media tra i due angoli di sterzo delle ruote sull'assale anteriore.

Gli output del sistema vengono restituiti dalle seguenti righe di codice:

```
% Fattori fisici caratterizzanti il modello a bicicletta:
m = 2107.74; % kg
a = 1.480; % m
b = 1.479; % m
Iz = 3954.709; % kgm^2
Cf = 213757; % N/rad
Cr = 273576; % N/rad
V = 8.333; % m/s

A = [ -(Cf+Cr)/(m*V) (b*Cr-a*Cf-m*V^2)/(m*V^2);
      (b*Cr-a*Cf)/Iz -(Cf*a^2+Cr*b^2)/(m*Iz) ];
B = [ Cf/(m*V);
      (a*Cf)/Iz ];
C = [ 1 0;
      0 1;
      (-Cf-Cr)/(m*V^2) (b*Cr-a*Cf)/(m*V^3);
      -1 -(a/V);
      -1 (b/V);
      (-Cf-Cr)/m (b*Cr-a*Cf)/(m*V) ];
D = [ 0;
      0;
      Cf/(m*V^2);
      1;
      0;
      Cf/m ];
% Creazione del sistema dinamico nello spazio degli stati
G = ss(A,B,C,D);
% Output del sistema dinamico in funzione dell'input angolo di sterzo
y = lsim(G,delta_sterF,tempo)
```

Figura 84- Script per manovra di COLPO DI STERZO su Matlab

Gli output si trovano all'interno della matrice y dell'ultima riga, e sono gli stessi presentati nel modello a bicicletta dinamico caratterizzato dalle matrici A, B, C, D (riportate in queste righe di codice Matlab).

Output:

Angolo di assetto (β), Velocità di imbardata (ψ), Curvatura (ρ), Angolo di deriva front (α_f), Angolo di deriva rear (α_r), Accelerazione laterale (a_y).

$$(y) = \begin{pmatrix} \beta \\ \psi \\ \rho \\ \alpha_f \\ \alpha_r \\ a_y \end{pmatrix}$$

L'input del sistema è 'delta_sterF' ovvero la variabile contenete l'angolo di sterzo alle ruote in funzione del 'tempo'; entrambi vettori presi direttamente dai risultati di simulazione di IPG-CM.

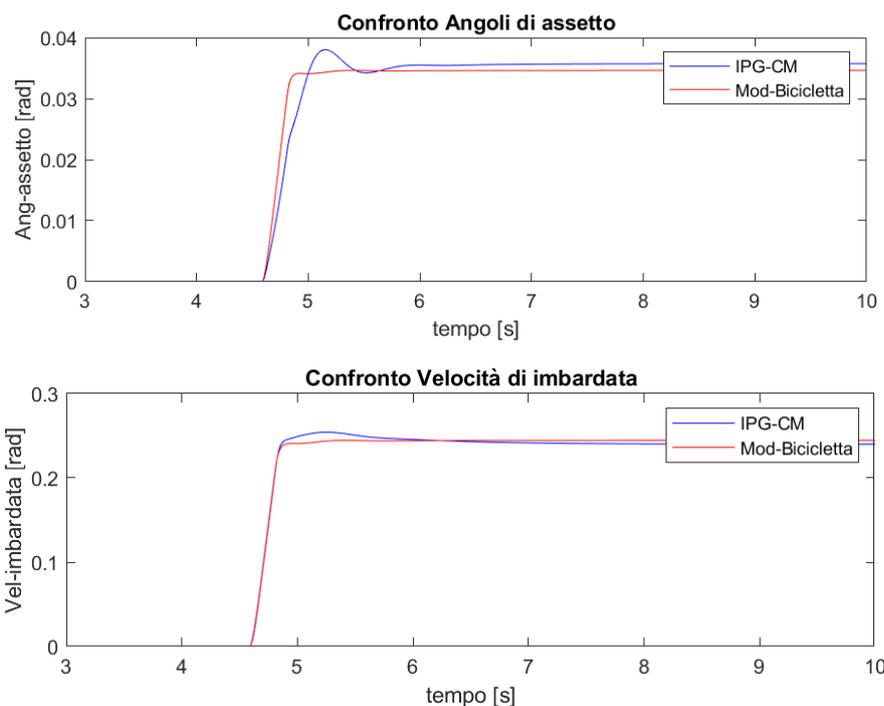
Si utilizzano gli stessi input di IPG-CM perché l'obiettivo della presente sezione è proprio quello di confrontare la risposta dinamica del modello di veicolo a più gradi di libertà di IPG-CM con il modello di veicolo a bicicletta a minor numero di gradi di libertà.

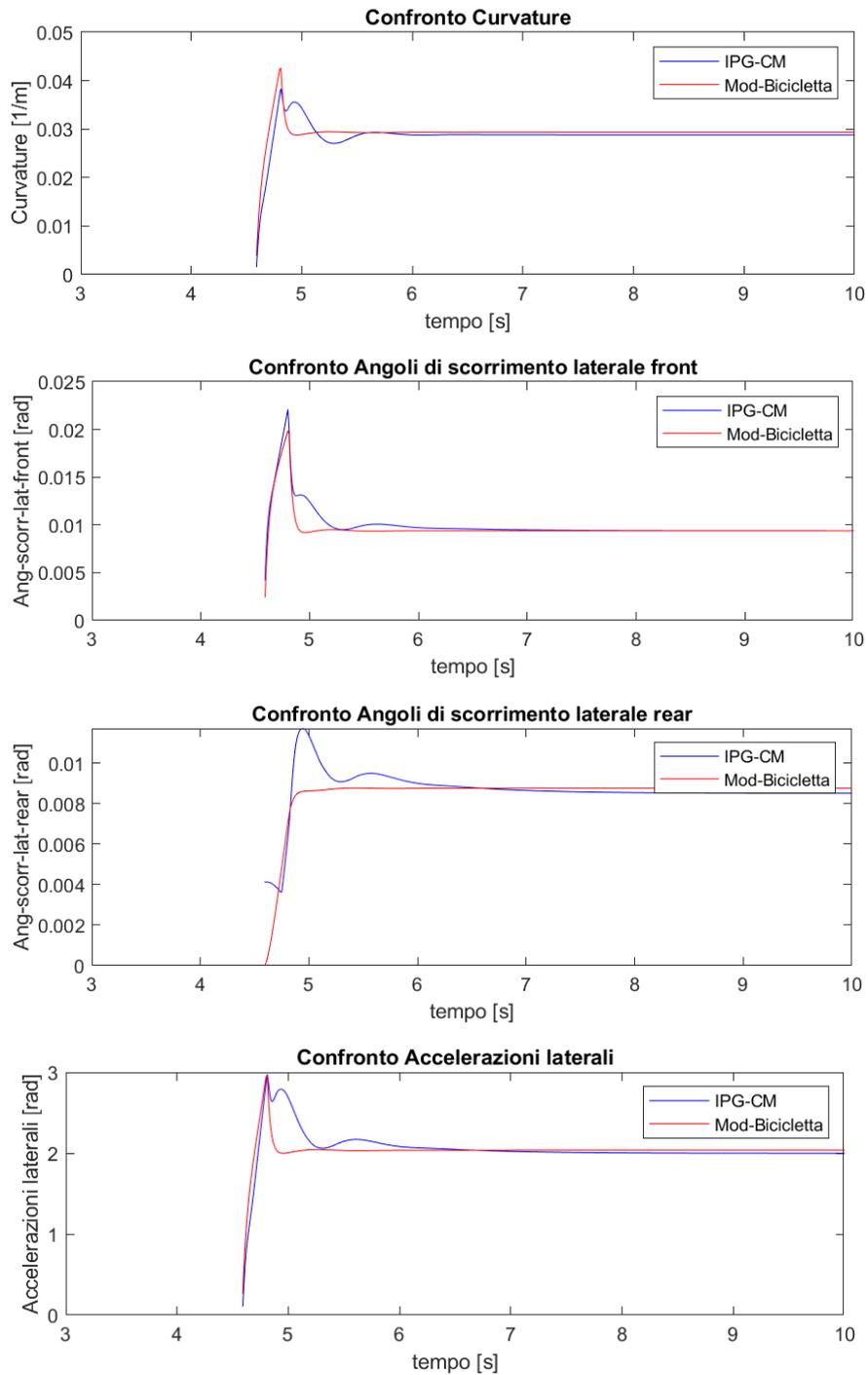
Ci si aspetta una risposta dei due modelli molto simile a basse accelerazioni laterali ($a_y = 2 \text{ m/s}^2$), quindi nel tratto lineare delle curve di lavoro, mentre un comportamento differente nella fase non lineare della curva, quindi ad accelerazioni superiori ($a_y = 7 \text{ m/s}^2$).

3.3.3 Risultati e confronto

Riportiamo i risultati, messi a confronto, degli output del modello di IPG-CM (in blu) con quelli del modello a bicicletta riportato su Matlab (in rosso). Vengono presentati i risultati per i tempo compreso tra i 3 e i 10 secondi della manovra, poiché, risulta di particolare interesse lo studio della fase transitoria, mentre la fase di accelerazione e quella a sterzo costante non variano significativamente gli output nel tempo:

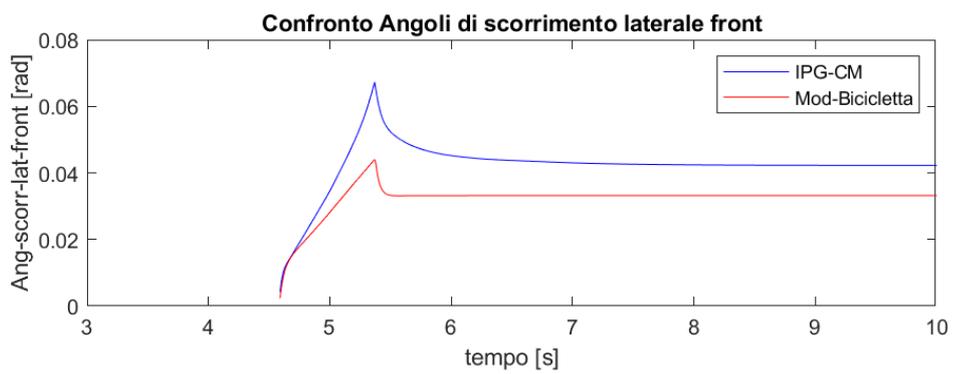
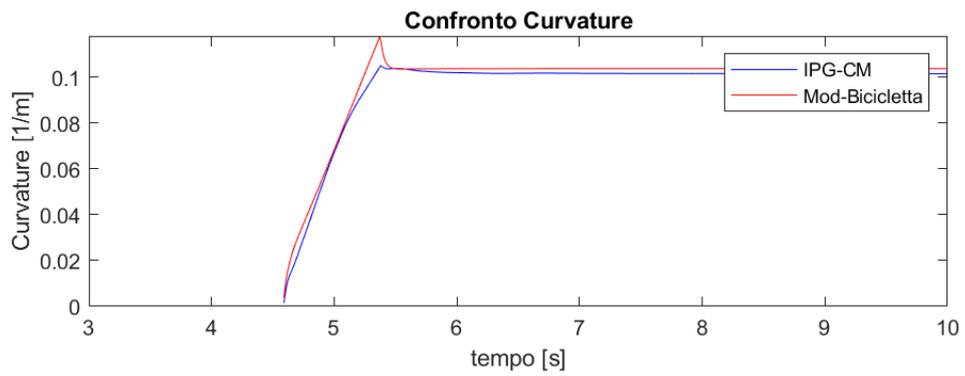
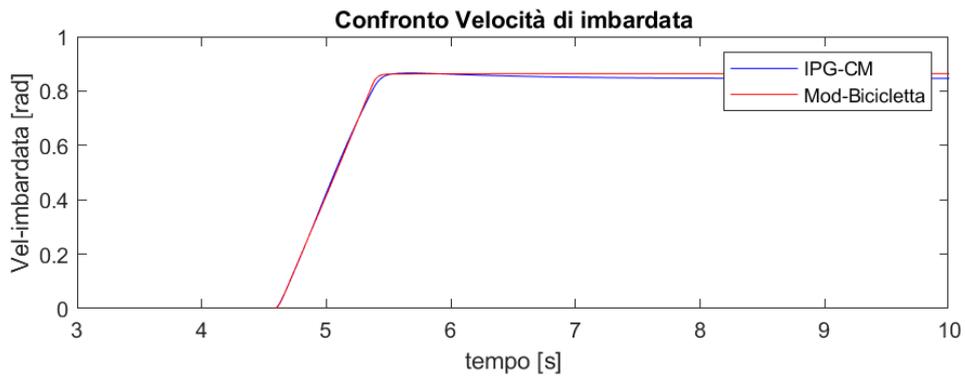
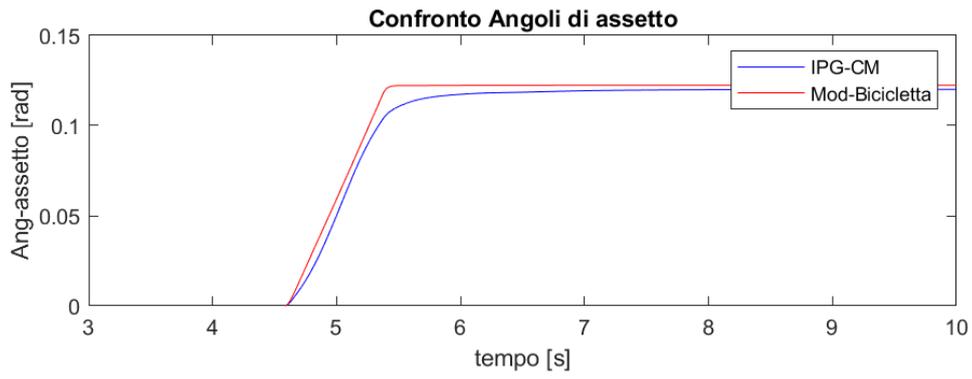
Accelerazione laterale $a_y = 2 \text{ m/s}^2$:

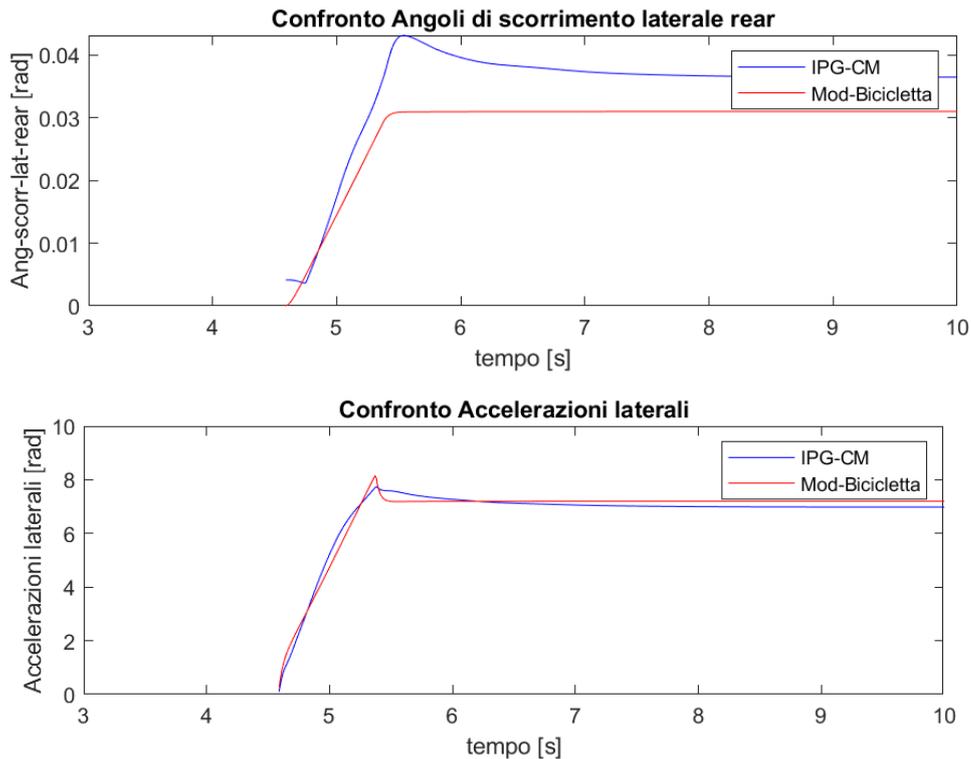




Come previsto, i risultati, nel tratto di lavoro lineare, sono molto simili tra il modello di IPG e il modello a bicicletta su Matlab, soprattutto nelle fasi stazionarie della manovra.

Accelerazione laterale $a_y = 7 \text{ m/s}^2$:





Si noti, come in questo caso, gli angoli di scorrimento siano differenti nei due modelli perché il modello a bicicletta su Matlab presenta dei coefficienti di rigidità in deriva, inseriti all'interno delle matrici (A, B, C, D) caratterizzanti il sistema dinamico ivi definito corrispondente alla fase lineare delle curve ($F_y - \alpha$). In particolare, le rigidità in deriva che caratterizzano il tratto lineare sono più elevate di quelle inerenti al tratto a pendenza decrescente delle curve ($F_y - \alpha$), pertanto, si hanno degli angoli di scorrimento in deriva (rispetto al tempo) del modello a bicicletta minori del modello più completo di IPG-CM che tiene conto dei comportamenti non lineari per accelerazioni laterali più spinte.

In generale, però, il modello a bicicletta restituisce dei valori di angolo di assetto, velocità di imbardata, curvatura e accelerazione sufficientemente simili ai valori restituiti dal modello a più gradi di libertà su IPG-CM, quindi, nei limiti della dinamica di cassa del veicolo i risultati che restituisce il modello a bicicletta sono accettabili.

3.4 Sistema di controllo

L'obiettivo della precedente validazione è quello di sfruttare il modello a bicicletta per lo sviluppo di un controllo (in ambiente Matlab e Simulink) sulla base di esso (quindi sulla base di un modello semplificato), per poi riportarlo all'interno della logica Simulink che governa l'angolo di sterzo del veicolo di simulazione CarMaker.

L'algoritmo di path tracking che viene qui sviluppato e presentato è di tipo 'feed-back' (la componente di 'feed-forward' viene trascurata anche se in un modello più realistico dovrebbe essere tenuta in considerazione per ridurre i tempi di calcolo in real-time). Tale tipo di controller si basa sull'esecuzione di un angolo volante che dipende direttamente dagli errori di posizione e di angolo di imbardata (oltre che delle rispettive derivate) del

veicolo rispetto la traiettoria di riferimento che si desidera eseguire e che può coincidere con quella generata dal path planner.

3.4.1 Modello dinamico di veicolo

Per la creazione del sistema di controllo, che permetterà l'esecuzione della traiettoria di riferimento generata dal path planning, si fa riferimento a un modello di veicolo caratterizzato da quattro variabili errore (nello state space) rispetto la traiettoria desiderata. Si fa riferimento al single-track model già presentato, ma si rappresentano solo le velocità e gli angoli di assetto caratterizzanti il baricentro del veicolo:

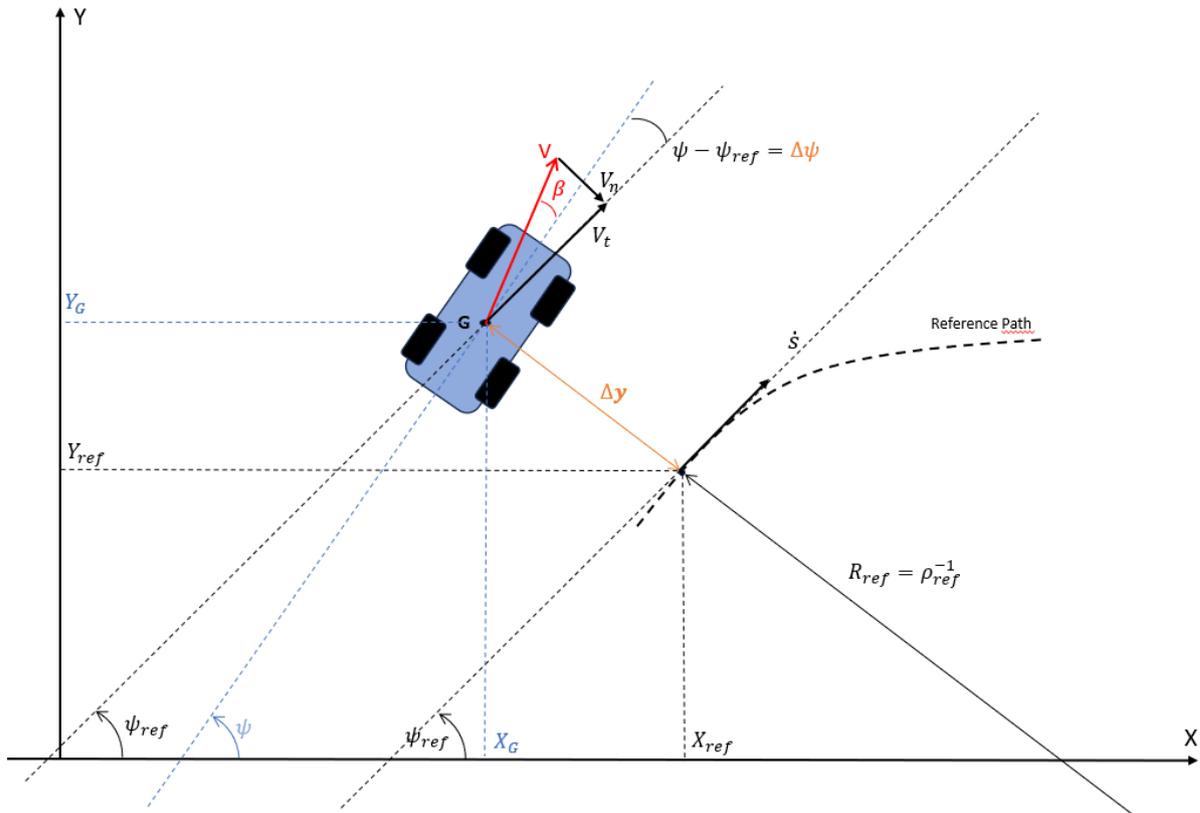


Figura 85 - Modello di veicolo con errori di posizione e imbardata

La variabile Δy è l'errore di posizione del baricentro del veicolo rispetto la traiettoria di riferimento in direzione trasversale e può essere calcolata nel seguente modo:

$$\Delta y = \sqrt{(X_G - X_{ref})^2 + (Y_G - Y_{ref})^2} \quad (3.35)$$

La variabile ψ è l'angolo di imbardata (yaw rate) che differisce dall'angolo di riferimento ψ_{ref} di un certo angolo di errore di imbardata $\Delta\psi = \psi - \psi_{ref}$. Quindi ψ è l'integrale nel tempo della velocità di imbardata $\dot{\psi}$ tenendo conto dell'angolo di imbardata iniziale.

Infine, viene indicata la curvatura della traiettoria di riferimento tramite la lettera greca ρ_{ref} .

Sotto ipotesi di velocità longitudinale costante ($v_x = cost$), la velocità di imbardata relativa alla curvatura della traiettoria di riferimento è:

$$\dot{\psi}_{ref} = \rho_{ref} v_x \quad (3.36)$$

L'accelerazione laterale sulla traiettoria di riferimento sarebbe:

$$\dot{v}_y(s) = \rho_{ref} v_x^2 = \dot{\psi}_{ref} v_x \quad (3.37)$$

L'errore di accelerazione e di velocità laterale rispetto la traiettoria di riferimento si possono esprimere come:

$$\Delta\dot{y} = (\dot{v}_y + v_x \dot{\psi}) - \dot{v}_y(s) = \dot{v}_y + v_x(\dot{\psi} - \dot{\psi}_{ref}) = \dot{v}_y + v_x \Delta\dot{\psi} \quad (3.38)$$

$$\Delta\dot{y} = v_y + v_x \sin(\Delta\psi) \approx v_y + v_x \Delta\psi \quad (3.39)$$

Gli errori di accelerazione di imbardata e di velocità di imbardata sono rispettivamente:

$$\Delta\ddot{\psi} = \ddot{\psi} - \ddot{\psi}_{ref} \quad (3.40)$$

$$\Delta\dot{\psi} = \dot{\psi} - \dot{\psi}_{ref} \quad (3.41)$$

Sostituendo queste espressioni all'interno delle equazioni che esprimono \dot{v}_y e $\ddot{\psi}$ del modello a bicicletta precedentemente presentato e riordinando i termini in modo tale da avere uno spazio degli stati caratterizzato dal vettore degli stati:

$$x = \begin{bmatrix} \Delta y \\ \Delta\dot{y} \\ \Delta\psi \\ \Delta\dot{\psi} \end{bmatrix}$$

E da un 'vettore' degli input:

$$u = \delta$$

Si noti come l'unico input scelto in questa trattazione è l'angolo di sterzo. Una trattazione più approfondita potrebbe utilizzare un ulteriore momento imbardante dato da un sistema di Torque Vectoring o da sistemi simili come presentato in [28].

$$\Delta\dot{y} - v_x \Delta\dot{\psi} = -\frac{(C_f + C_r)}{m v_x} (\Delta\dot{y} - v_x \Delta\dot{\psi}) + \left[\frac{b C_r - a C_f}{m v_x} \right] (\Delta\dot{\psi} + \dot{\psi}_{ref}) + \frac{C_f}{m} \delta \quad (3.42)$$

$$\Delta\ddot{\psi} + \ddot{\psi}_{ref} = \frac{b C_r - a C_f}{I_z v_x} (\Delta\dot{y} - v_x \Delta\dot{\psi}) - \frac{(a^2 C_f + b^2 C_r)}{I_z v_x} (\Delta\dot{\psi} + \dot{\psi}_{ref}) + \frac{a C_f}{I_z} \delta \quad (3.43)$$

Riorganizzando le equazioni in modo conforme all'ottenimento della rappresentazione nello spazio degli stati:

$$(\dot{x}) = [A](x) + [B](u) + [E](\dot{\psi}_{ref}) + [F](\ddot{\psi}_{ref}) \quad (3.44)$$

Si ottiene:

$$\begin{pmatrix} \Delta \dot{y} \\ \Delta \ddot{y} \\ \Delta \dot{\psi} \\ \Delta \ddot{\psi} \end{pmatrix} = \begin{bmatrix} 0 & 1 & 0 & 0 \\ 0 & \frac{-(C_f + C_r)}{mv_x} & \frac{C_f + C_r}{m} & \frac{bC_r - aC_f}{mv_x} \\ 0 & 0 & 0 & 1 \\ 0 & \frac{bC_r - aC_f}{I_z v_x} & \frac{aC_f - bC_r}{I_z} & \frac{-(a^2 C_f + b^2 C_r)}{I_z v_x} \end{bmatrix} \begin{pmatrix} \Delta y \\ \Delta \dot{y} \\ \Delta \psi \\ \Delta \dot{\psi} \end{pmatrix} + \begin{bmatrix} 0 \\ C_f \\ m \\ aC_f \\ I_z \end{bmatrix} \delta \\
+ \begin{bmatrix} 0 \\ \frac{bC_r - aC_f}{mv_x} - v_x \\ 0 \\ \frac{-(a^2 C_f + b^2 C_r)}{I_z v_x} \end{bmatrix} \dot{\psi}_{ref} + \begin{bmatrix} 0 \\ 0 \\ 0 \\ -1 \end{bmatrix} \ddot{\psi}_{ref} \quad (3.45)$$

Si tenga presente che i due termini $\dot{\psi}_{ref}$ e $\ddot{\psi}_{ref}$ sono dati derivanti dalla traiettoria di riferimento generata dal sistema di path planning presentato nel capitolo 2.

Il sistema appena creato servirà per controllare la posizione e l'orientamento del veicolo rispetto la traiettoria di riferimento, e tramite l'ottimizzatore LQR ('Linear Quadratic Regulator') si riuscirà a ottenere un valore del guadagno K che moltiplicato per lo spazio degli stati x porterà al calcolo di un input (angolo volante) atto all'esecuzione della traiettoria desiderata più possibile vicina a quella di riferimento. Questo viene eseguito tramite la minimizzazione delle variabili "errore" (di posizione e di orientamento) presenti all'interno del vettore degli stati (x). Una descrizione più approfondita seguirà nella sezione relativa allo sviluppo in Simulink del modello di controllo.

Per l'implementazione del controllo è necessaria la creazione della seconda equazione matriciale relativa allo spazio degli stati:

$$(y) = [C](x) + [D](u)$$

Vista la necessità di controllare le variabili errore (già presenti nello spazio degli stati) si vogliono come output y del sistema gli stati stessi. Pertanto, la seconda equazione risulta essere:

$$\begin{pmatrix} \Delta y \\ \Delta \dot{y} \\ \Delta \psi \\ \Delta \dot{\psi} \end{pmatrix} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{pmatrix} \Delta y \\ \Delta \dot{y} \\ \Delta \psi \\ \Delta \dot{\psi} \end{pmatrix} + \begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \end{bmatrix} \delta \quad (3.46)$$

Riassumendo, le matrici dello spazio degli stati, così creato, saranno:

$$[A] = \begin{bmatrix} 0 & 1 & 0 & 0 \\ 0 & \frac{-(C_f + C_r)}{mv_x} & \frac{C_f + C_r}{m} & \frac{bC_r - aC_f}{mv_x} \\ 0 & 0 & 0 & 1 \\ 0 & \frac{bC_r - aC_f}{I_z v_x} & \frac{aC_f - bC_r}{I_z} & \frac{-(a^2 C_f + b^2 C_r)}{I_z v_x} \end{bmatrix}$$

$$[B] = \begin{bmatrix} 0 \\ \frac{C_f}{m} \\ 0 \\ \frac{aC_f}{I_z} \end{bmatrix}$$

$$[C] = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$[D] = \begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \end{bmatrix}$$

$$[E] = \begin{bmatrix} 0 \\ \frac{bC_r - aC_f}{mv_x} - v_x \\ 0 \\ \frac{-(a^2C_f + b^2C_r)}{I_z v_x} \end{bmatrix}$$

$$[F] = \begin{bmatrix} 0 \\ 0 \\ 0 \\ -1 \end{bmatrix}$$

Le tre matrici $[B]$, $[E]$, $[F]$ saranno condensate, nella fase di validazione di questo modello tramite CarMaker e Matlab, in un'unica matrice degli input $[\tilde{B}]$ (per riuscire a creare un sistema dinamico in ambiente Matlab):

$$[\tilde{B}] = \begin{bmatrix} 0 & 0 & 0 \\ \frac{C_f}{m} & \frac{bC_r - aC_f}{mv_x} - v_x & 0 \\ 0 & 0 & 0 \\ \frac{aC_f}{I_z} & \frac{-(a^2C_f + b^2C_r)}{I_z v_x} & -1 \end{bmatrix}$$

3.4.2 Validazione modello stati errore

Il modello di veicolo con quattro stati errore deve essere validato mostrando il suo reale funzionamento tramite l'osservazione dei risultati (errori di posizione trasversale Δy e di imbardata $\Delta\psi$) restituiti da semplici manovre eseguite direttamente in Matlab tramite la funzione '*lsim*' già sfruttata nella validazione del modello a bicicletta a due stati. Si osservano le risposte del sistema y che coincidono, per come sono state definite le equazioni dello spazio degli stati, con gli stati errore da valutare.

Le due manovre eseguite sono:

- 1) Traiettoria rettilinea parallela alla traiettoria di riferimento con un Δy cost

2) Traiettoria rettilinea inclinata rispetto la traiettoria di riferimento Δy *crescente*

1) Traiettoria rettilinea con offset costante

La prima simulazione prevede l'esecuzione di una traiettoria rettilinea parallela alla linea di riferimento con un offset costante ($\Delta y = \text{cost}$) scelto arbitrariamente pari a 5 metri. Si imposta un valore degli stati iniziali paria a:

$$x_0 = \begin{pmatrix} \Delta y_0 \\ \Delta \dot{y}_0 \\ \Delta \psi_0 \\ \Delta \dot{\psi}_0 \end{pmatrix} = \begin{pmatrix} 5 \\ 0 \\ 0 \\ 0 \end{pmatrix}$$

Si vuole mantenere questo vettore degli errori per tutto il percorso; quindi, si impone un vettore angolo volante costante pari a 0, un vettore della velocità di imbardata e dell'accelerazione di imbardata di riferimento costantemente pari a 0. Il vettore degli input risulta quindi essere:

$$u = \begin{pmatrix} \delta \\ \dot{\psi}_{ref} \\ \ddot{\psi}_{ref} \end{pmatrix}$$

Questa condensazione degli input è stata effettuata per poter creare uno spazio degli stati G tramite la funzione 'ss' di Matlab che accetta solo quattro matrici (A, B, C, D) e non ammetterebbe un set di matrici (A, B, C, D, E, F). Quindi, le seguenti righe di codice sono state implementate per calcolare gli output del sistema:

$$G = ss(A, \tilde{B}, C, D) \quad (3.47)$$

$$y = lsim(G, u, tempo, x_0) \quad (3.48)$$

Si ottengono pertanto i seguenti risultati:

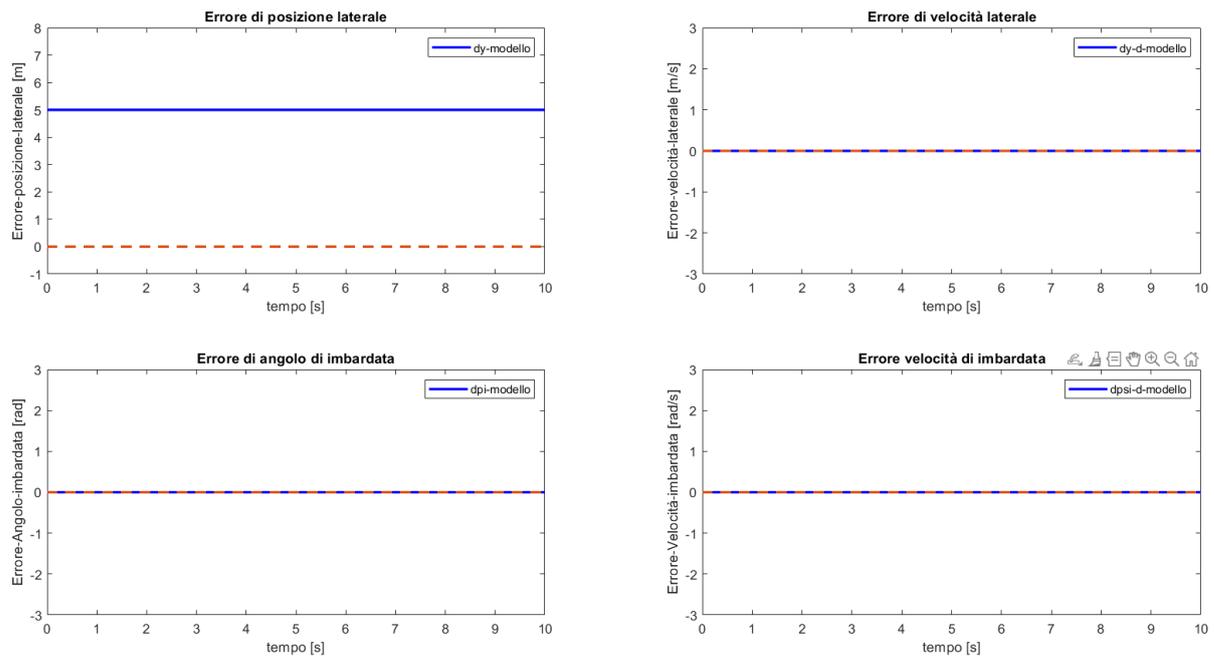


Figura 86 - Risultati validazione modello 4 stati su traiettoria rettilinea

Come ci si poteva aspettare l'errore di posizione laterale è costantemente pari al suo valore iniziale, mentre gli altri errori sono correttamente pari a 0.

2) Traiettoria rettilinea inclinata con offset crescente

Questa simulazione è analoga alla precedente in quanto presenta gli stessi input del sistema ma un diverso vettore degli stati errore iniziale, poiché è caratterizzato da un angolo di imbardata errore iniziale pari a $\pi/4$ radianti:

$$x_0 = \begin{pmatrix} \Delta y_0 \\ \Delta \dot{y}_0 \\ \Delta \psi_0 \\ \Delta \dot{\psi}_0 \end{pmatrix} = \begin{pmatrix} 5 \\ 0 \\ \pi/4 \\ 0 \end{pmatrix}$$

Ci si aspetta un errore crescente di posizione laterale, un errore di velocità laterale costante e pari a un dato valore poiché ci si sta allontanando dalla linea di riferimento, un errore di angolo di imbardata costante e pari a $\pi/4$ rad, e un errore di velocità di imbardata pari a 0 poiché non dovrebbe aumentare l'angolo ($\delta = \dot{\psi}_{ref} = \ddot{\psi}_{ref} = cost = 0$).

I risultati ottenuti dalle stesse righe di codice sopra riportate sono:

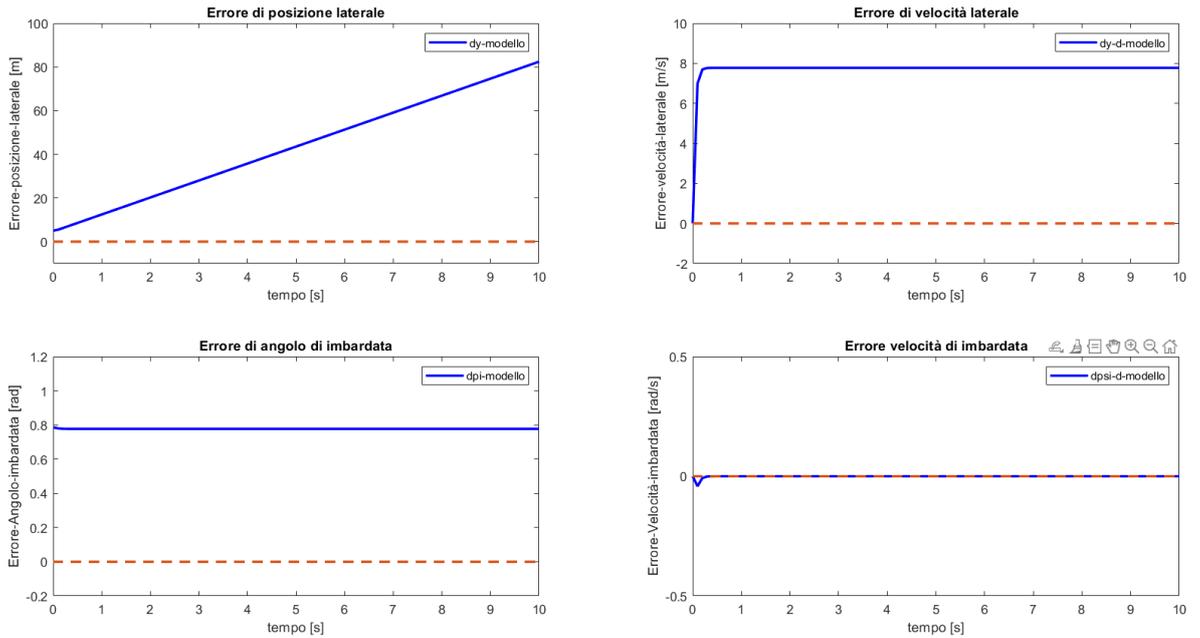


Figura 87 - Risultati validazione modello 4 stati su traiettoria rettilinea inclinata

Si può concludere che il modello dinamico di veicolo a quattro stati rappresenta opportunamente gli stati errore di posizione, di angolo di imbardata e le rispettive derivate.

3.4.3 Modello Simulink

Il modello Simulink che è stato realizzato per la creazione e calibrazione del controller che verrà implementato successivamente in IPG-CarMaker per il controllo dello sterzo presenta il seguente schema complessivo:

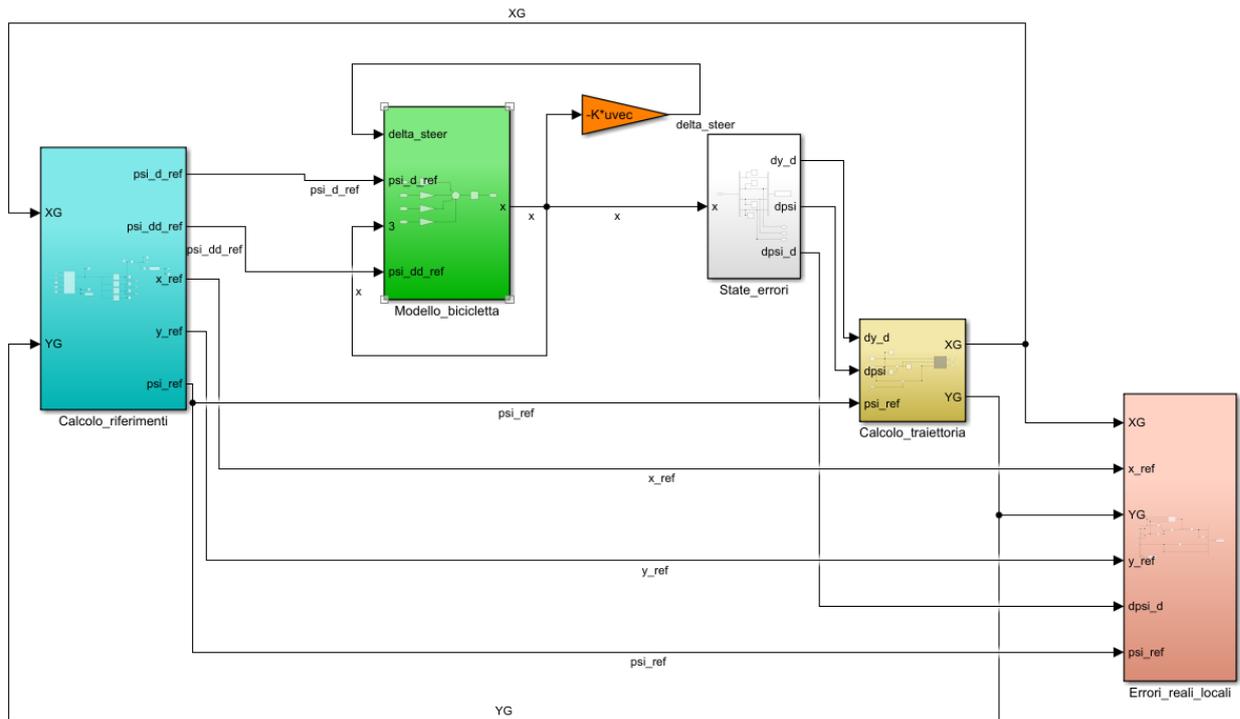


Figura 88 - Modello Simulink per creazione e validazione modello path tracking

Si identificano i seguenti sottosistemi partendo da sinistra:

- 1) **‘Calcolo_riferimenti’**: Calcola, tramite la funzione ‘near_point’ e le quattro Look-up-Table 1-D, il vettore $[x_{ref}, y_{ref}, \psi_{ref}, \rho_{ref}]$ relativo al punto più vicino della traiettoria di riferimento rispetto la posizione attuale del veicolo per ogni passo di simulazione. Questo vettore di riferimenti viene utilizzato per il calcolo della velocità e dell’accelerazione di imbardata di riferimento $[\dot{\psi}_{ref}, \ddot{\psi}_{ref}]$, oltre che per la valutazione delle coordinate $[x_{ref}, y_{ref}]$ della traiettoria di riferimento a ogni passo iterativo, che è stato scelto costante e pari a $dt = 10^{-3} s$.

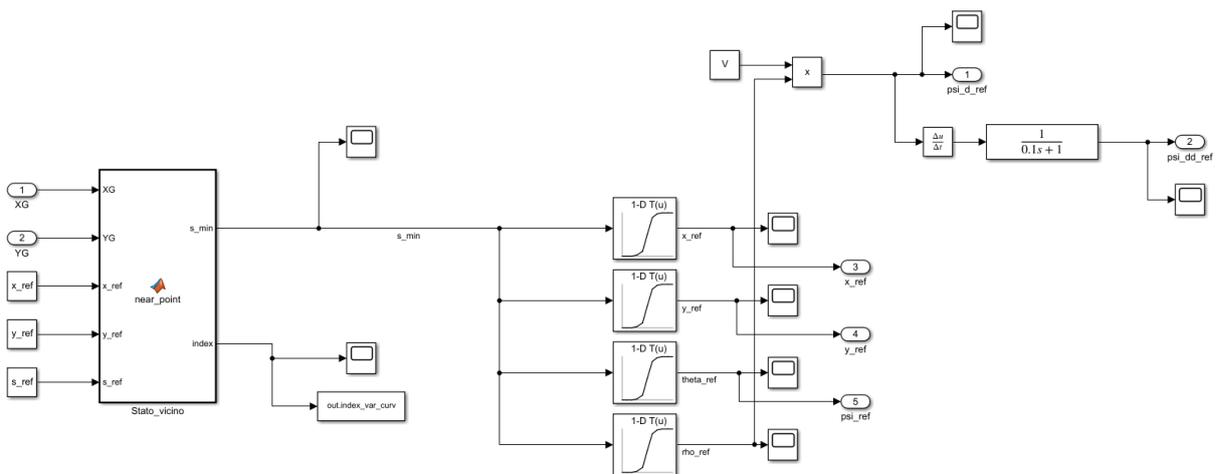


Figura 89 - Subsystem ‘Calcolo_riferimenti’

È stato inserito un filtro (funzione di trasferimento) alla frequenza di 10 Hz all'uscita della derivata della $\dot{\psi}_{ref}$ perché presentava una forte oscillazione numerica indesiderata ai fini della simulazione.

Le uscite di tale sottosistema sono pertanto: $[\dot{\psi}_{ref}, \ddot{\psi}_{ref}, x_{ref}, y_{ref}, \psi_{ref}]$. Tali segnali vengono inviati per ogni passo iterativo agli altri sottosistemi.

- 2) **'Modello_bicicletta'**: Dentro tale sottosistema si trova il modello di veicolo a quattro stati già lungamente presentato e validato nei precedenti paragrafi. Presenta la seguente struttura:

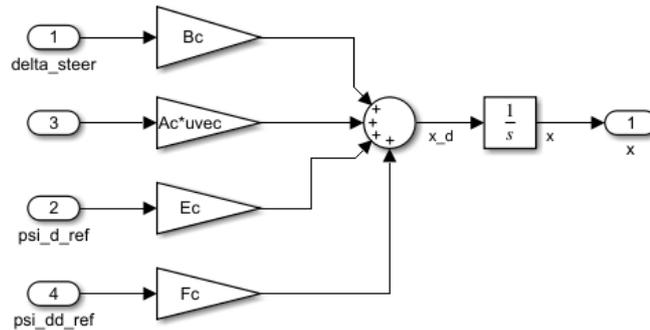


Figura 90 - Subsystem 'Modello_bicicletta'

Questo schema specifica semplicemente l'espressione (3.44):

$$\dot{(x)} = [A](x) + [B](u) + [E]\dot{\psi}_{ref} + [F]\ddot{\psi}_{ref}$$

Dove il 'vettore' degli input, che in questo caso è solo l'angolo volante $u = \delta = \text{delta_steer}$, è calcolato, esternamente a questo subsystem, come:

$$(u) = \delta = -Kx \quad (3.49)$$

Dove x è il vettore dei quattro stati errore:

$$(x) = \begin{pmatrix} \Delta y \\ \Delta \dot{y} \\ \Delta \psi \\ \Delta \dot{\psi} \end{pmatrix}$$

E K è il vettore dei guadagni ottimi per l'ottenimento dell'angolo volante di feedback, che viene calcolato come guadagno ottimo secondo la logica 'LQR' ('Linear Quadratic Regulator') (che verrà spiegata più approfonditamente nel paragrafo 3.4.5):

$$u = \delta = -(k_{\Delta y}\Delta y + k_{\Delta \dot{y}}\Delta \dot{y} + k_{\Delta \psi}\Delta \psi + k_{\Delta \dot{\psi}}\Delta \dot{\psi}) \quad (3.50)$$

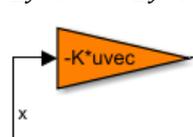


Figura 91 - Guadagno ottimo LQR

Gli ingressi del sottosistema 'Modello_bicicletta' sono $[\delta, (x), \dot{\psi}_{ref}, \ddot{\psi}_{ref}]$ e la singola uscita è (x) .

- 3) **'State_errori'**: Al suo interno è presente una semplice manipolazione del vettore degli stati errore (x) , quindi una sua suddivisione nei singoli elementi e un output

delle sole tre variabili di interesse $[\Delta\dot{y}, \Delta\psi, \Delta\dot{\psi}]$ per altri subsystem 'Calcolo_traiettoria' e 'Errori_reali_locali'.

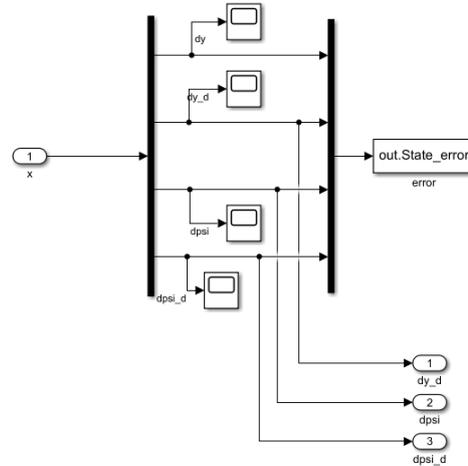


Figura 92 - Subsystem 'State_error'

- 4) **'Calcolo_traiettoria'**: Questo sottosistema serve per passare dalle velocità nel sistema di riferimento locale (x, y) alle velocità nel sistema di riferimento globale (X, Y) , quindi al calcolo della traiettoria globale $[X_G, Y_G]$ realmente eseguita durante la manovra.

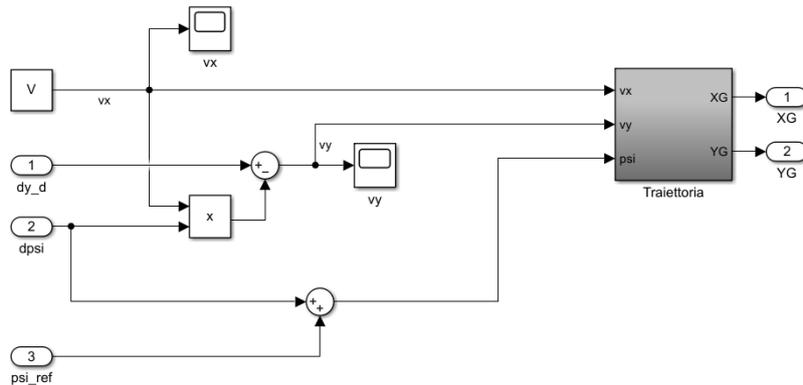


Figura 93 - Subsystem 'Calcolo_traiettoria'

In questa prima parte vengono estratte le variabili $[\Delta\dot{y}, \Delta\psi, \psi_{ref}]$, oltre che la velocità tangenziale $v_x \approx V$, tenuta costante per semplicità e perché il modello a bicicletta utilizza delle matrici che contengono termini caratterizzati dalla velocità del veicolo che deve, almeno in un dato istante di tempo, essere costante. Un modello più evoluto potrebbe tenere conto di velocità del veicolo variabile lungo il percorso.

Le tre variabili locali $[v_x, v_y, \psi]$ che entrano nel subsystem 'Traiettoria' vengono calcolate nel seguente modo (come si può osservare dallo schema:

$$\begin{aligned}
 v_x &\approx V \\
 v_y &= \Delta\dot{y} - V\Delta\psi \\
 \psi &= \Delta\psi + \psi_{ref}
 \end{aligned}
 \tag{3.51}$$

Dove ψ_{ref} è estratto dal sottosistema ‘Calcolo_riferimenti’, mentre le variabili errore da ‘State_errori’.

All’interno di ‘Traiettoria’ avviene il calcolo delle coordinate realmente eseguite durante la manovra:

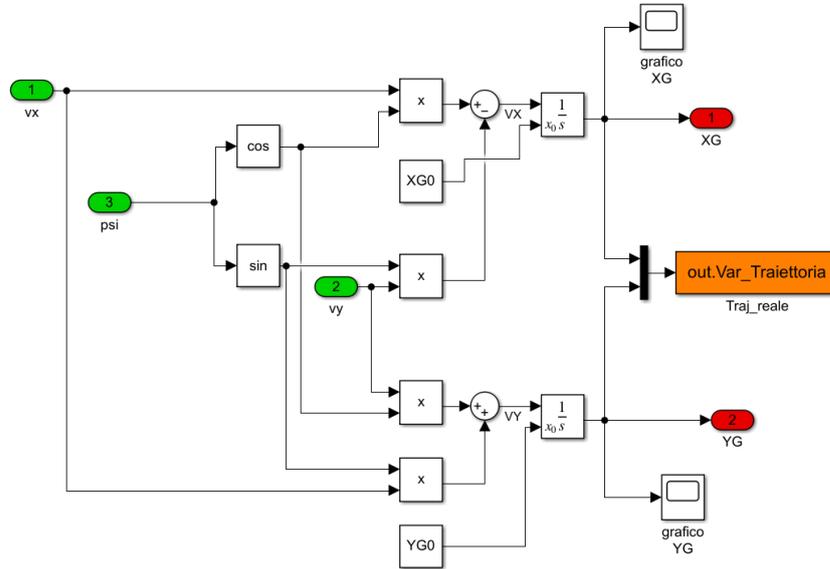


Figura 94 - Subsystem 'Traiettoria'

Le velocità lungo X e lungo Y nel sistema di riferimento globale, ottenute dalle velocità locali del veicolo solo calcolate come:

$$V_X = v_x \cos \psi - v_y \sin \psi \quad (3.52)$$

$$V_Y = v_x \sin \psi + v_y \cos \psi \quad (3.53)$$

Integrando tali espressioni e tenendo conto delle posizioni iniziali del veicolo $[X_{G0}, Y_{G0}]$ si ottengono le $[X_G, Y_G]$ all’avanzare della simulazione.

5) ‘**Errori_reali_locali**’: Qui vengono calcolati gli errori realmente commessi nell’esecuzione della traiettoria rispetto il punto di riferimento preso come guida. Tramite il confronto di questi errori con gli stati errore del modello a bicicletta, si può osservare come il modello a bicicletta restituisca degli errori di posizione $[\Delta y, \Delta \dot{y}]$ non corrispondenti a quelli reali. Questa difformità porterà a una necessaria rimodulazione dei guadagni del vettore K nel controllo dell’angolo volante che verrà sviluppato in ambiente Simulink for CarMaker.

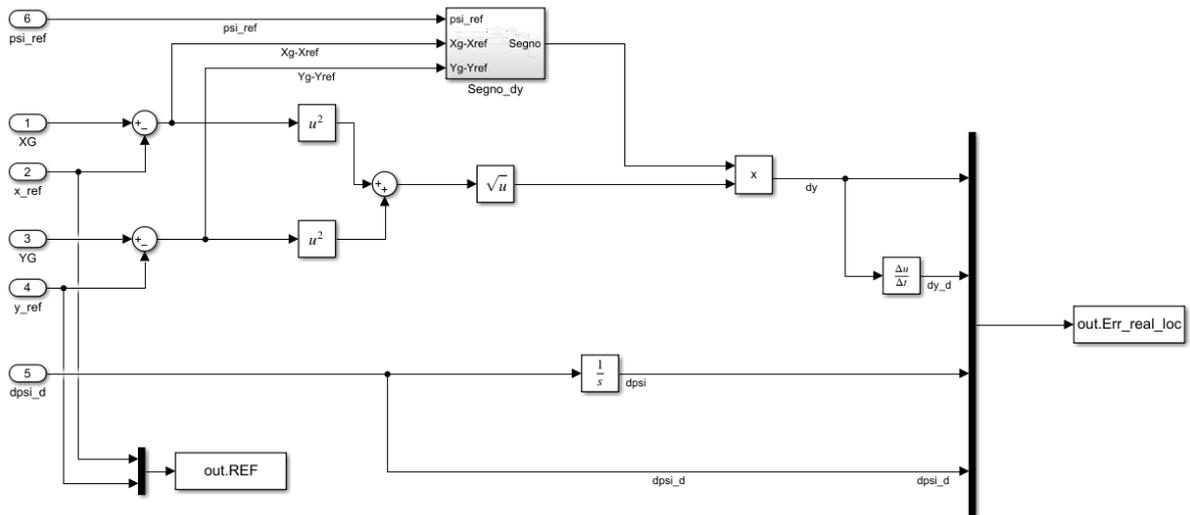


Figura 95 - Subsystem 'Errori_reali_locali'

L'errore di posizione, con modulo e segno è stato calcolato come:

$$\Delta y = \sqrt{(X - X_{ref})^2 + (Y - Y_{ref})^2} \text{Sign}(\Delta y^*) \quad (3.54)$$

Il segno dell'errore di posizione laterale è stato calcolato come suggerito in [27]:

$$\text{Sign}(\Delta y^*)$$

$$\Delta y^* = [(X_{ref} - X) (Y_{ref} - Y) 0] \times [\cos \psi_{ref} \sin \psi_{ref} 0] \quad (3.55)$$

Mentre l'errore di angolo di imbardata e di velocità di imbardata sono stati presi direttamente dal modello a quattro stati; pertanto, non è stato necessario calcolarne il segno.

$$\Delta \psi = \psi - \psi_{ref}$$

Modelli utilizzati per il calcolo del punto più vicino di riferimento

Sono stati messi a confronto i risultati restituiti da due modelli analoghi, differenziati solo per il metodo di calcolo della coordinata curvilinea ' s_{ref} ' in cui prendere il vettore $[x_{ref}, y_{ref}, \psi_{ref}, \rho_{ref}]$ di riferimento del punto più vicino alla posizione attuale del veicolo, per ogni passo di simulazione.

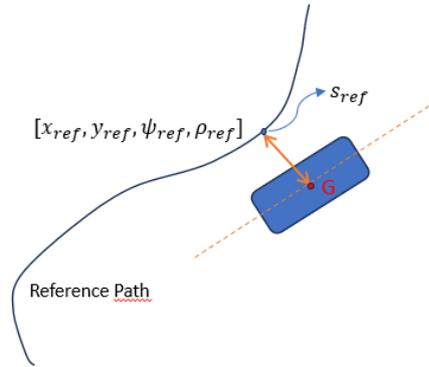


Figura 96 - Rappresentazione significato coordinata curvilinea di riferimento

Calcolo riferimenti con funzione ‘Stato_vicino’

La coordinate curvilinea ‘ s_{ref} ’ (s_{min}) da dare alle Look-up-Table $[s_{ref}, x_{ref}]$, $[s_{ref}, y_{ref}]$, $[s_{ref}, \psi_{ref}]$, $[s_{ref}, \rho_{ref}]$ può essere calcolata (come è stato fatto nel modello Simulink fin qui descritto) tramite una funzione, qui chiamata ‘near_point’ dentro la ‘fcn’ Simulink ‘Stato_vicino’ che presenta il seguente script:

```
function [s_min,index] = near_point(XG,YG,x_ref,y_ref,s_ref)

xg = XG*ones(length(x_ref),1);
yg = YG*ones(length(y_ref),1);

dist = sqrt((xg-x_ref).^2+(yg-y_ref).^2);
[val,index] = min(dist);
s_min = s_ref(index,1);
```

Figura 97 - Funzione ‘Stato_vicino’ per calcolo coordinata curvilinea di riferimento

In sostanza si trova la coordinata curvilinea s_{min} alla quale estrarre, ad un dato istante di tempo, i valori di riferimento del punto più vicino, alla posizione attuale del veicolo, del reference path. Si cerca quindi la coordinata curvilinea di riferimento alla quale calcolare i due valori di riferimento di imbardata $[\dot{\psi}_{ref}, \ddot{\psi}_{ref}]$.

Calcolo riferimenti con coordinata curvilinea approssimata

Nel secondo modello si usa una formulazione continua della variabile curvilinea ‘ s ’ (distanza percorsa nella traiettoria di riferimento), come integrale della velocità curvilinea \dot{s} . La distanza curvilinea percorsa in un dato lasso di tempo T è la seguente, confermata da [27] e [28]:

$$s = \int_0^T \frac{v_x \cos \Delta\psi - v_y \sin \Delta\psi}{1 - \rho_{ref} \Delta y} dt = \int_0^T \frac{v_x \cos \Delta\psi - v_y \sin \Delta\psi}{1 - \rho_{ref} [(Y - Y_{ref}) \cos \psi_{ref} + (X - X_{ref}) \sin \psi_{ref}]} dt \quad (3.56)$$

In questo secondo caso si nota una notevole riduzione del tempo di simulazione e sono scomparsi gli andamenti a gradino e/o oscillatori che si presentavano nella soluzione ‘Stato_vicino’ in alcuni segnali che andavano, quindi, successivamente filtrati.

Il valore di 's' così calcolato andava inserito nelle Look-up-Table come è stato fatto nel modello parallelo, ottenendo, così, i valori di riferimento $[x_{ref}, y_{ref}, \psi_{ref}, \rho_{ref}]$, $[\dot{\psi}_{ref}, \ddot{\psi}_{ref}]$.

Si riporta la differente configurazione del sottosistema 'Calcolo_riferimenti' che differisce per gli input rispetto al modello lungamente descritto in precedenza.

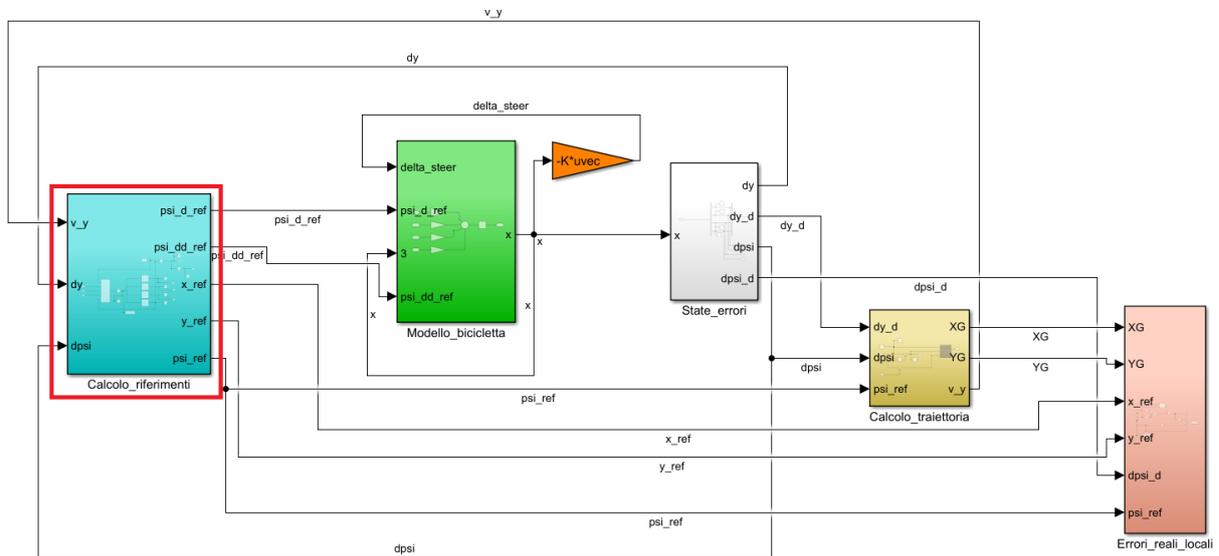


Figura 98 - Modello Simulink con calcolo traiettoria di riferimento continua ma approssimata

Qui gli input sono $[v_y, \Delta y, \Delta \psi]$ che vengono utilizzati nel seguente modo:

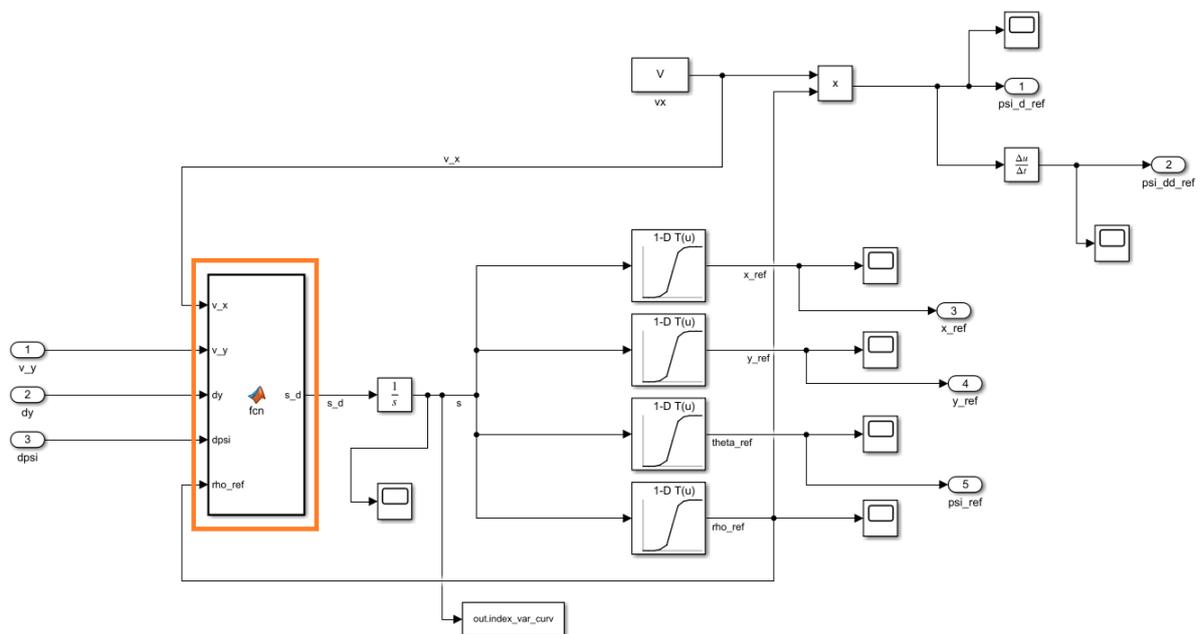


Figura 99 - Subsystem 'Calcolo_riferimenti'

La function 'fcn' contiene le righe di codice corrispondenti alla funzione per il calcolo di s sopra riportata.

```
function s_d = fcn(v_x,v_y,dy,dpsi,rho_ref)

s_d = (v_x*cos(dpsi)-v_y*sin(dpsi))/(1-rho_ref*dy);
```

Figura 100 - Funzione approssimata per calcolo della coordinata curvilinea di riferimento

Questa soluzione rimane pur sempre un'approssimazione della variabile curvilinea s , pertanto, nel proseguo, si è continuato ad utilizzare l'altro modello, anche se richiedente più tempo computazionale.

3.4.4 Risultati validazione modello controller su Simulink

Inizializzando le opportune variabili in uno script Matlab (matrici del modello, vettore dei guadagni K dell'LQR, ecc....) si può lanciare il modello Simulink appena presentato e osservare la risposta nella persecuzione della traiettoria di riferimento da parte del single-track model grazie al sistema di controllo associato.

Il funzionamento e significato del guadagno ottimo restituito dall' LQR verrà spiegato nel paragrafo successivo, mentre qui vengono presentati dei primi risultati relativi a questa logica nell'esecuzione di quattro traiettorie:

Traiettoria rettilinea

La traiettoria rettilinea viene impostata con un vettore degli stati errore iniziale:

$$(x_0) = [\Delta y_0 \quad \Delta \dot{y}_0 \quad \Delta \psi_0 \quad \Delta \dot{\psi}_0]' = [5 \ 0 \ \pi/8 \ 0]'$$

Le matrici dei pesi relativi agli stati errore vengono impostate come prima prova:

$$Q = \begin{bmatrix} q_1 & 0 & 0 & 0 \\ 0 & q_2 & 0 & 0 \\ 0 & 0 & q_3 & 0 \\ 0 & 0 & 0 & q_4 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 0.2 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0.2 \end{bmatrix}$$

Quella relativa all'angolo volante (control effort):

$$R = r = 0.1$$

Pertanto, si è scelto di penalizzare maggiormente la presenza degli errori di posizione e di imbardata rispetto al costo di attuazione dell'angolo volante per la correzione degli errori stessi (per dirla in parole povere: non si dà tanto peso ai consumi ma si vogliono limitare al massimo gli errori commessi nell'esecuzione della traiettoria).

Quello che si ottiene avviando la simulazione è il seguente risultato:

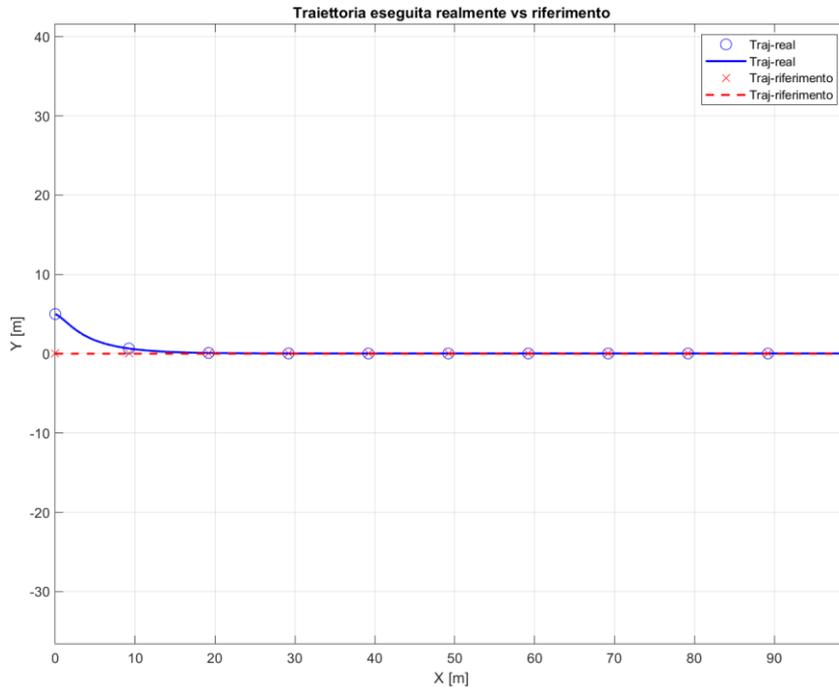


Figura 101 - Risultato inseguimento traiettoria rettilinea

Si riportano gli stati errore $[\Delta y, \Delta \dot{y}, \Delta \psi, \Delta \dot{\psi}]$ corrispondenti: in verde quelli calcolati dal modello a quattro stati, in nero quelli calcolati in funzione della traiettoria ottenuta tramite il sottosistema ‘Calcolo_trajettoria’:

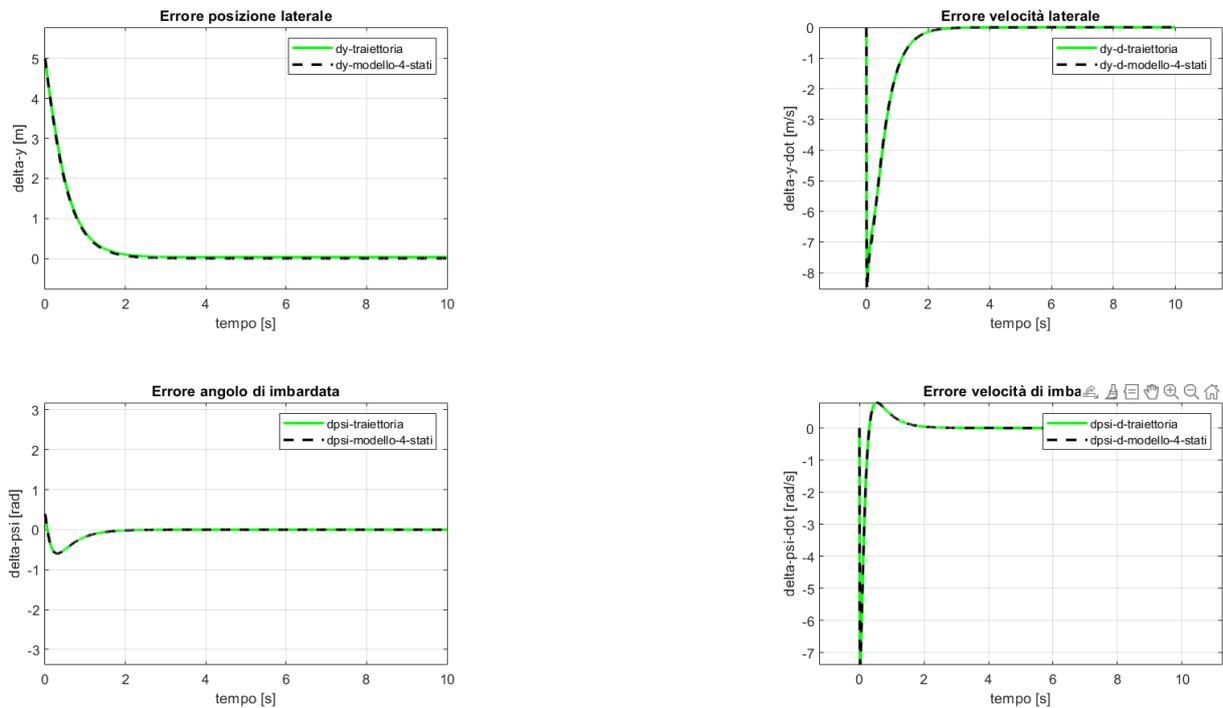


Figura 102 - Stati errore inseguimento traiettoria rettilinea

Una diversa impostazione dei pesi associati alle variabili errore (Q) e all’azione di controllo (R) (angolo volante) porta a una variazione degli stati errore tra il percorso eseguito dal single-track model e il tracciato di riferimento. Il suo studio di sensitività viene eseguita nel paragrafo 3.4.5.

Traiettoria rettilinea inclinata

Il vettore degli stati errore iniziale viene mantenuto uguale al caso rettilineo, come anche i pesi inseriti nelle matrici Q e R. La traiettoria di riferimento ottenuta, pertanto, è la seguente:

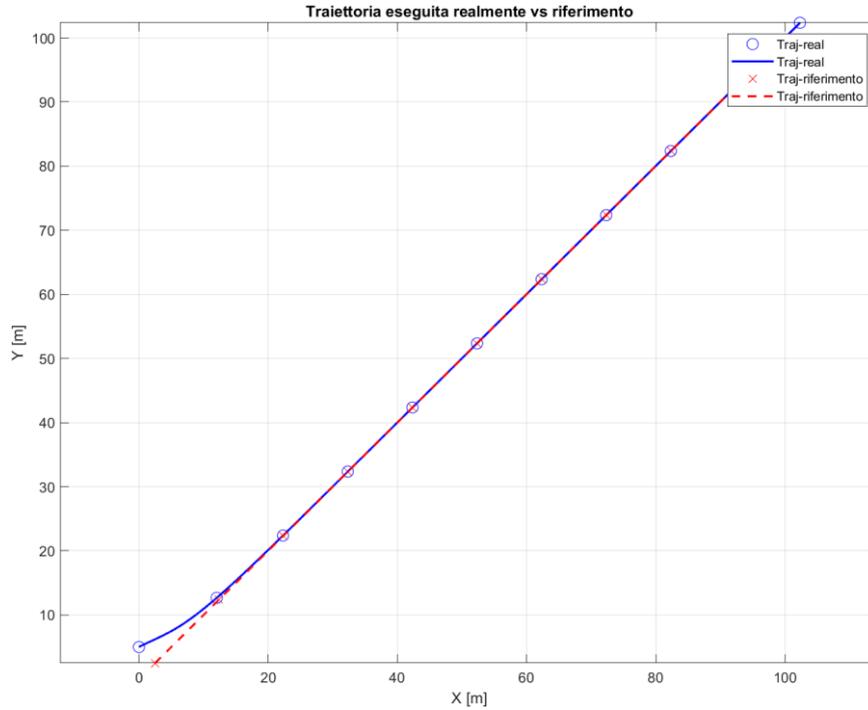


Figura 103 - Risultato perseguimento traiettoria rettilinea inclinata

Mentre il vettore degli stati errore riporta i seguenti risultati:

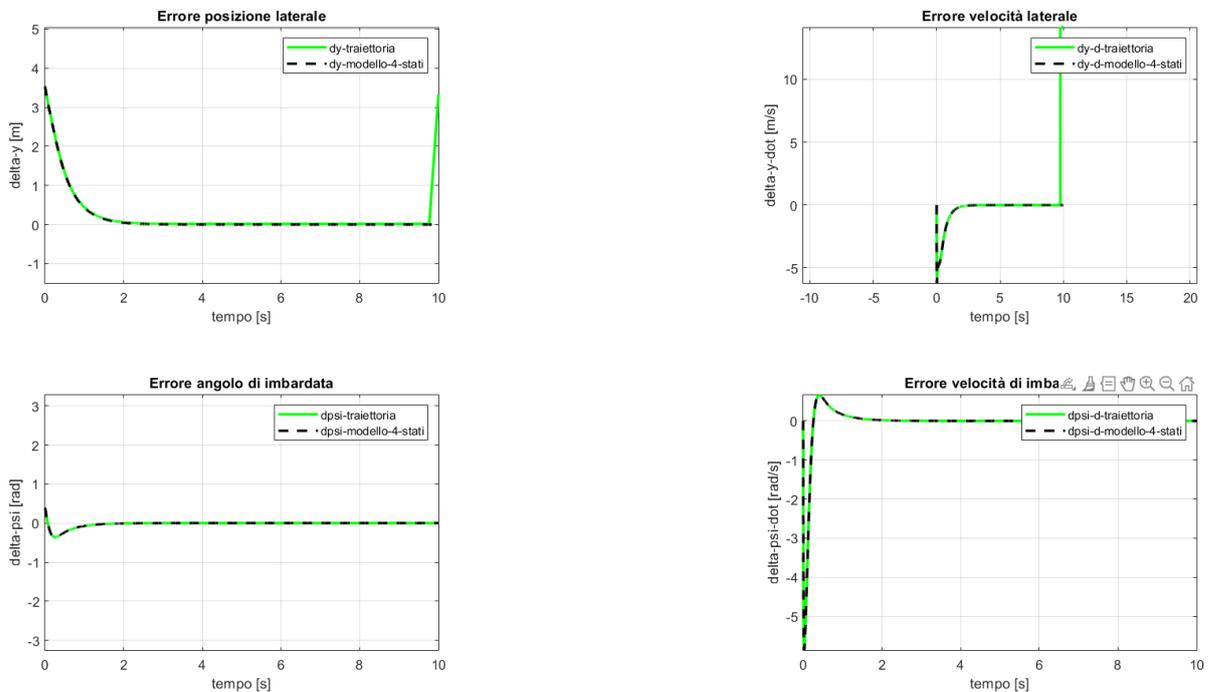


Figura 104 - Stati errore perseguimento traiettoria rettilinea inclinata

Anche in questo caso si nota una leggera differenza tra le variabili errore del modello a quattro stati e calcolate a partire dalla traiettoria generata durante la simulazione.

Traiettoria circolare

Anche in questo caso si è scelto di mantenere gli stessi stati errore iniziale e gli stessi costi Q ed R.

Il raggio della circonferenza che si è scelto come prova della presente logica di controllo è di 25 m:

$$R_{circonf,prova} = 25 \text{ m}$$

Il tracciato di riferimento viene seguito fedelmente dalla logica di controllo:

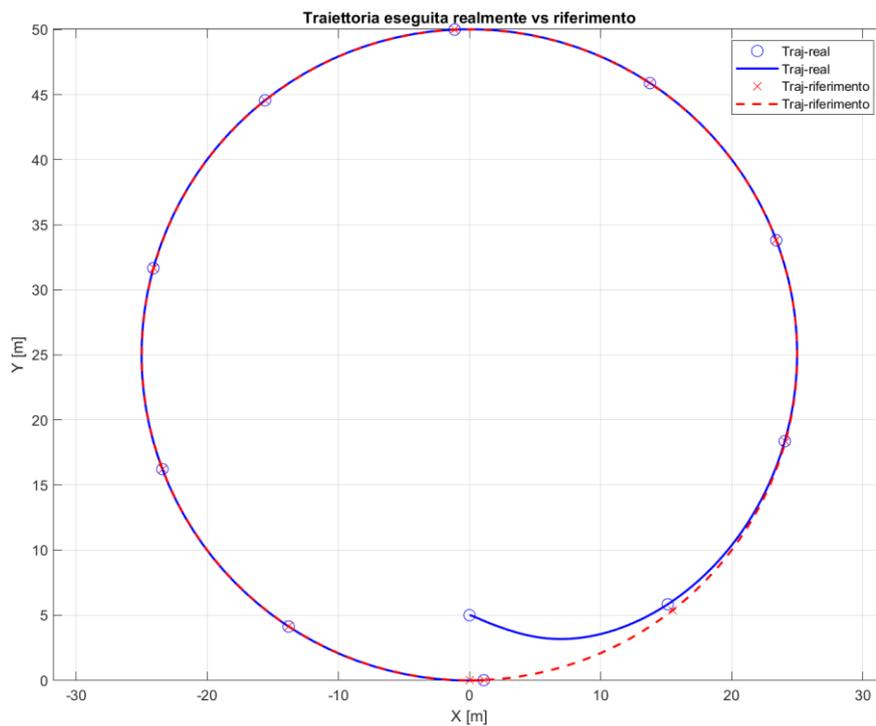


Figura 105 - Risultato perseguimento traiettoria circolare

I relativi errori presentano una piccola discrepanza tra quelli del modello a quattro stati e quelli calcolati in base alla traiettoria realmente eseguita, ma per lo più trascurabile:

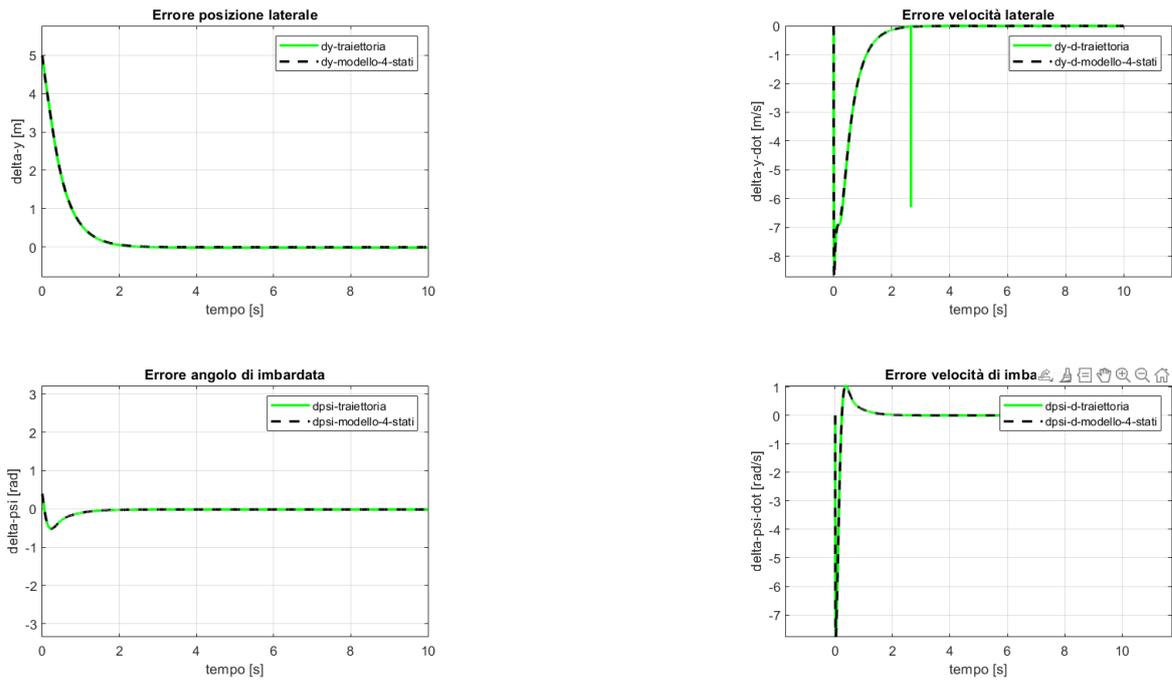


Figura 106 - Stati errore perseguimento traiettoria circolare

Tracciato del path planning

La traiettoria che è stata generata dal path planner, presentato nel capitolo 2, deve essere ora eseguita tramite questa logica di controllo, allo stesso modo in cui è riuscita ad eseguire questi tre esempi di traiettorie più semplici.

La stessa traiettoria andrà eseguita su IPG-CarMaker una volta implementato il controllo che si sta ora verificando e calibrando su Simulink.

Il vettore degli stati errore e i guadagni Q e R da dare alla logica LQR sono gli stessi di prima, e portano al seguente risultato:

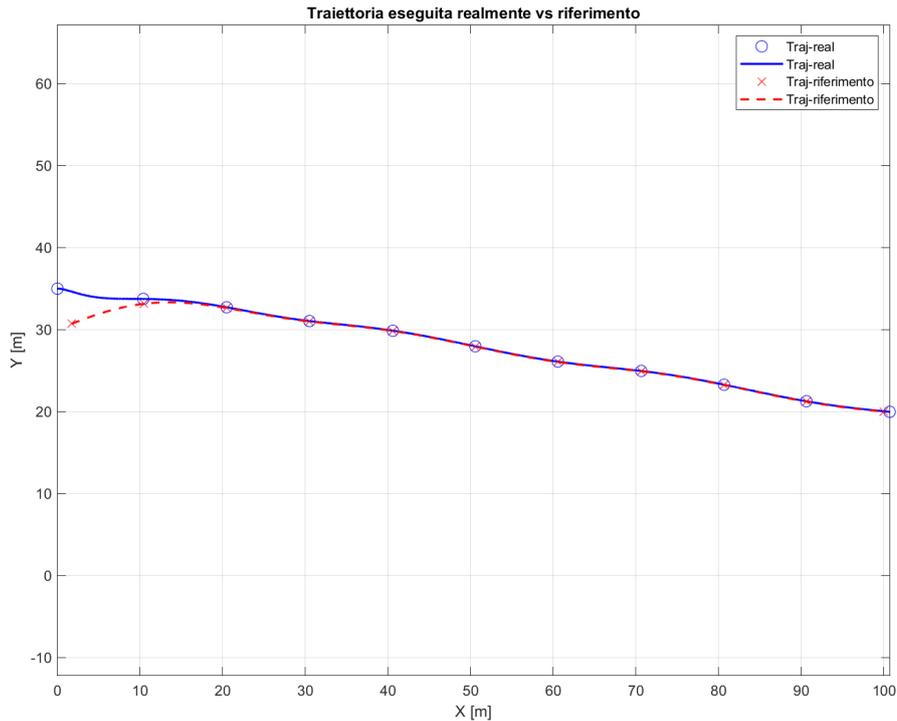


Figura 107 - Risultato perseguimento traiettoria path planning

Dopo una prima fase di raggiungimento della traiettoria di riferimento gli errori commessi sono molto ridotti, da come si può osservare dagli stati:

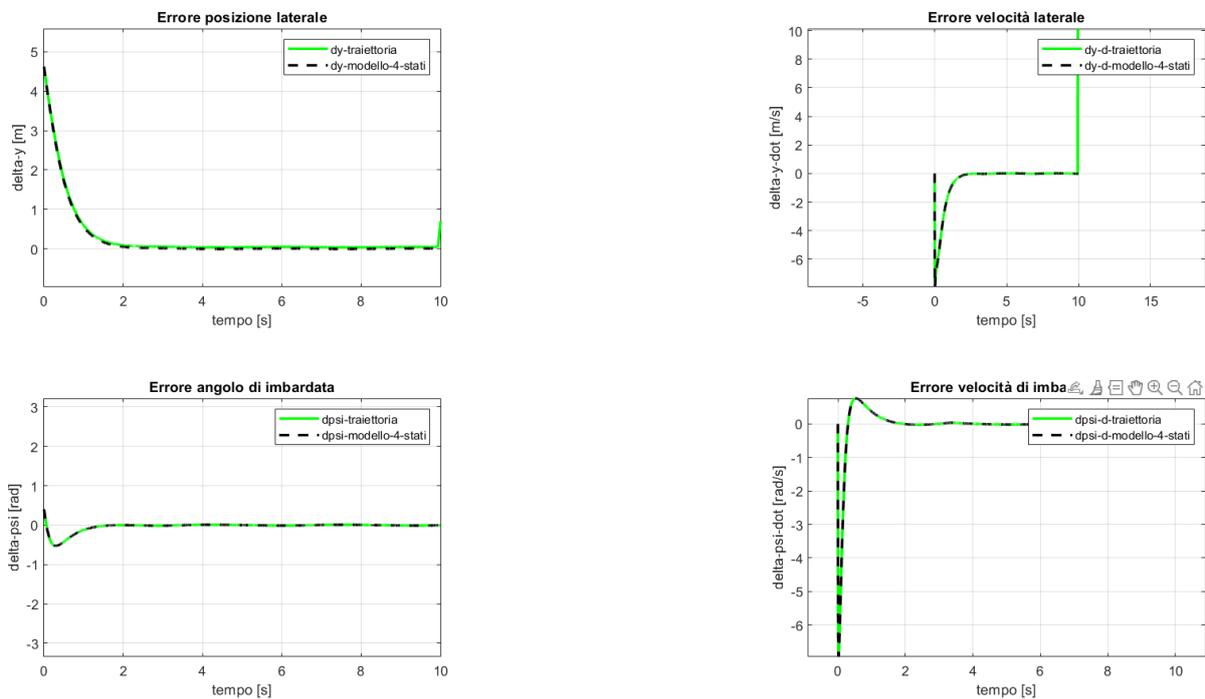


Figura 108 - Stati errore perseguimento traiettoria path planning

Le differenze tra i due errori (del modello a quattro stati e quelli calcolati nell'esecuzione della traiettoria) sono basse; pertanto, questo modello di path tracking può essere ritenuto valido per una sua successiva implementazione in ambiente CarMaker.

3.4.5 Calibrazione parametri ottimizzatore LQR

In questa sezione viene spiegata brevemente la teoria che c'è alla base dell'LQR ('Linear Quadratic Regulator'), utile per l'ottenimento della matrice K dei guadagni ottimi che moltiplicati per gli stati errore di posizione e di imbardata restituiscono l'azione di controllo (angolo volante):

$$u = \delta = - [k_{\Delta y} \quad k_{\Delta \dot{y}} \quad k_{\Delta \psi} \quad k_{\Delta \dot{\psi}}] \begin{bmatrix} \Delta y \\ \Delta \dot{y} \\ \Delta \psi \\ \Delta \dot{\psi} \end{bmatrix} = - [K][x] \quad (3.57)$$

Il vettore dei guadagni [K] viene calcolato solo una volta in tutta la simulazione (non cambia ad ogni istante temporale) sulla base dei pesi che si assegnano agli stati errore e al peso legato all'azione di controllo.

Il vettore dei guadagni, quindi, viene calcolato tramite la teoria del LQR che in Matlab è implementata all'interno della funzione 'lqr' di cui si spiega il funzionamento.

Teoria alla base del LQR

Il LQR [32], [33], [34] è un metodo, esattamente come il Pole Placement e il PID controller, per la determinazione della matrice guadagno da moltiplicare per gli stati o gli output di un sistema dinamico, per ridurre al minimo la differenza tra un output desiderato di riferimento e gli effettivi output che un dato sistema restituisce. Uno schema riassume questo concetto:

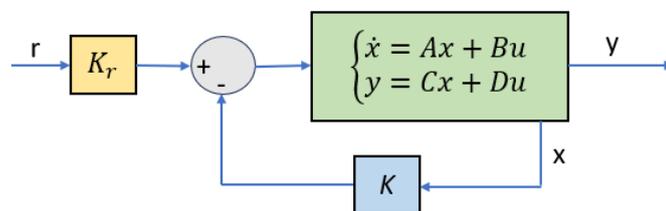


Figura 109 - Sistema generico closed-loop

Tale schema è comune a un qualsiasi sistema closed loop (soggetto a controllo). La caratteristica del controller sviluppato nel presente lavoro è che esso è solo un feedback control, mentre la parte di feedforward è mancante. La sua assenza comporta sicuramente dei valori di feedback control input maggiori che nel caso in cui il feedforward fosse presente; pertanto, comporta dei tempi di calcolo computazionale decisamente più lunghi, ma ai fini di testare la traiettoria di path planning offline è stato sufficiente implementare la parte di feedback. Di conseguenza, il vettore dei riferimenti 'r' è da considerarsi nullo, mentre, come già detto, il vettore degli stati errore 'x' viene moltiplicato per la matrice 'K' dei guadagni ottimi, ottenuta tramite il LQR.

Il LQR, esattamente come il Pole Placement è un 'full state feedback control', ovvero considera tutti gli stati errore nel calcolo del control input di ritorno, secondo la formulazione sopra riportata. La differenza tra il metodo Pole Placement e l'LQR è che:

- Il Pole Placement sceglie la matrice dei guadagni K in modo da coprire i poli del sistema aperto, rendendolo quindi chiuso (closed loop).
- Il LQR sceglie la matrice dei guadagni K in modo da ottimizzare il controllo.

Con il LQR la matrice K dei guadagni viene scelta al fine di minimizzare la Funzione di costo:

$$J = \int_0^{\infty} (x^T Q x + u^T R u) dt \quad (3.58)$$

In questa funzione si possono osservare le matrici Q e R legate, come già detto, ai pesi degli stati del sistema 'x' e agli input di controllo 'u'. Questi pesi vanno scelti in base agli obiettivi del progetto a cui si sta lavorando:

- Se si desidera avere alte Performance del feedback control, nel lavoro in esame bassi valori dell'errore di posizione e di imbardata, allora si dovrà impostare una matrice Q elevata, al fine di penalizzare alti valori degli stati.
- Se invece si desidera puntare su un basso consumo di risorse o bassi sforzi di controllo (Actuation efforts), è bene scegliere un valore dei pesi della matrice R elevati.

In definitiva dipende dal bilancio tra 'Performance' e 'Actuation efforts' che si vogliono ottenere:

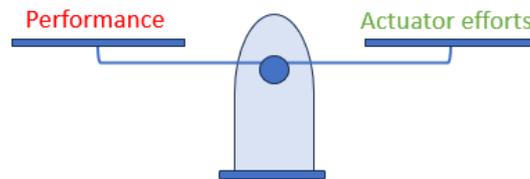


Figura 110 - Bilanciamento Performance - Actuation efforts

Nota: 'LQR' sta per 'Linear Quadratic Regulator'. Lineare perché l'input di controllo 'u' che si ottiene è una funzione lineare degli stati del sistema 'x'; Quadratica perché si basa sulla minimizzazione di una funzione di costo 'J' quadratica, che presenterà, pertanto, sempre un minimo.

Un requisito importante per il calcolo della funzione di costo è che le matrici Q e R siano definite positive:

$$x^T Q x > 0$$

$$u^T R u > 0$$

Solitamente questa condizione è verificata dal momento in cui le matrici Q e R sono scelte diagonali positive, assegnando a ogni stato appartenente al vettore x e a ogni control input un peso specifico:

$$Q = \begin{bmatrix} q_1 & 0 & & 0 \\ 0 & q_2 & & \\ & & \ddots & 0 \\ 0 & & 0 & q_n \end{bmatrix}$$

$$R = \begin{bmatrix} r_1 & 0 & & \\ 0 & r_2 & & \\ & & \ddots & 0 \\ 0 & & 0 & r_n \end{bmatrix}$$

La funzione di costo può essere riscritta in forma matriciale aggregata nel seguente modo:

$$J = \int_0^{\infty} [x^T \quad u^T] \begin{bmatrix} Q & N \\ N^T & R \end{bmatrix} \begin{bmatrix} x \\ u \end{bmatrix} dt = \int_0^{\infty} (x^T Q x + u^T R u + 2x^T N u) dt \quad (3.59)$$

Dove la matrice N è una matrice che penalizza i prodotti incrociati tra x ed u; pertanto, si cerca di mantenerla pari a 0.

Di solito, si parte da matrici Q e R identità, e pian piano si variano i loro valori in base agli output che il sistema restituisce, quindi, si aumenteranno e diminuiranno i coefficienti a seconda di cosa si vuole penalizzare o premiare.

Per ricavare la matrice dei guadagni K si deve risolvere la funzione di costo, cercandone il suo minimo. In generale non è un'operazione semplice, ma ci sono delle condizioni per cui la soluzione si trova molto più facilmente:

- Se la matrice N = 0
- Se il sistema dinamico presenta la seconda equazione: $y = Cx + Du$ in cui $C = I$ e $D = 0$, allora $y = x$, quindi il sistema con il controller di feedback sarà della forma:

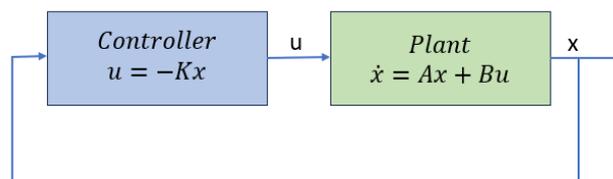


Figura 111 - Sistema di controllo solo feedback

Allora conterà solo la prima equazione del sistema $\dot{x} = Ax + Bu$.

Sotto queste ipotesi si possono attuare tre metodi per la ricerca della K ottimale che minimizzi la funzione di costo 'J':

- 1) Si calcolano molti valori di J per diversi valori di K, quindi per molti valori di Q e R, ed iterativamente si trova la K che minimizza J → Molto inefficiente
- 2) Tramite algoritmi di apprendimento basati sul calcolo dei gradienti in un punto della superficie della funzione J, mi avvicino verso il punto più basso di J → Inefficiente ma flessibile
- 3) Uso un approccio analitico per sistemi LINEARI → Equazione di Ricatti. → Basta risolvere un'equazione.

Ricatti equation

Segue una dimostrazione matematica dei passaggi per l'ottenimento dell'equazione finale di Ricatti [35]:

- Si introduce una matrice incognita $P = P^T$
- Riscrivo la funzione di costo:

$$J = x_0^T P x_0 - x_0^T P x_0 + \int_0^\infty (x^T Q x + u^T R u) dt \quad (3.60)$$

- Dove x_0 è lo stato iniziale del sistema.

$$J = x_0^T P x_0 + \int_0^\infty \left[\frac{d}{dt} (x^T P x) + (x^T Q x + u^T R u) \right] dt$$

- Questa espressione equivale alla precedente poiché:

$$\int_0^\infty \left[\frac{d}{dt} (x^T P x) \right] dt = [x^T P x]_0^\infty = 0 - x_0^T P x_0 \quad (3.61)$$

perché a tempo infinito, se il sistema tende a una soluzione stabile $x_0 \rightarrow 0$.

- Si può scrivere ancora:

$$\frac{d}{dt} (x^T P x) = \dot{x}^T P x + x^T P \dot{x} = (Ax + Bu)^T P x + x^T P (Ax + Bu) \quad (3.62)$$

- Quindi la funzione di costo:

$$J = x_0^T P x_0 + \int_0^\infty [(Ax + Bu)^T P x + x^T P (Ax + Bu) + (x^T Q x + u^T R u)] dt$$

- Ora vogliamo trovare la u che minimizzi questa funzione di costo; quindi,

$$J = x_0^T P x_0 + \int_0^\infty [x^T (A^T P + PA + Q)x + u^T R u + x^T P B u + u^T B^T P x] dt \quad (3.63)$$

suddividiamo i termini che caratterizzano questa equazione nel seguente modo:

- Solo i termini dentro l'integrale in blu dipendono dall'azione di controllo 'u', pertanto, per minimizzare la funzione J si deve trovare il valore di u che minimizzi quella componente.
- Tale espressione si può riscrivere come:

$$u^T R u + x^T P B u + u^T B^T P x = (u + R^{-1} B^T P x)^T R (u + R^{-1} B^T P x) - x^T (P B R^{-1} B^T P) x \quad (3.64)$$

- Quindi la funzione di costo viene riscritta come:

$$J = x_0^T P x_0 + \int_0^\infty [x^T (A^T P + PA + Q - P B R^{-1} B^T P) x + (u + R^{-1} B^T P x)^T R (u + R^{-1} B^T P x)] dt \quad (3.65)$$

- Questo termine

$$(u + R^{-1} B^T P x)^T R (u + R^{-1} B^T P x)$$

è uguale a 0 se:

$$u = -R^{-1} B^T P x \quad (3.66)$$

- Il control input si scrive come

$$u = -Kx$$

- Pertanto, si avrà:

$$K = R^{-1} B^T P \quad (3.67)$$

- Risolvendo parallelamente la richiesta dell'equazione di Riccati si trova la P:

$$A^T P + PA + Q - P B R^{-1} B^T P = 0 \quad (3.68)$$

sempre per minimizzare la funzione J

- Risolvendo la Riccati equation si ottiene la matrice P.

- Inserendo la matrice P in $K = R^{-1}B^T P$ si ottiene definitivamente la matrice di guadagno ottimo.
- Nota: L'equazione di Riccati ha due soluzioni, ma solo una da un sistema closed loop stabile.

Studio di sensitività e calibrazione

Vista la parte teorica di come la matrice dei guadagno ottimi viene ottenuta analiticamente si procede all'utilizzo della funzione 'lqr', già implementata in Matlab, che a partire dalle matrici (A, B, Q, R) trova la matrice K, secondo il seguente script:

$$[K, P, S] = lqr(A, B, Q, R) \quad (3.69)$$

Dove P è la soluzione dell'equazione algebrica di Riccati (come riportato sopra), S sono i poli del sistema dinamico closed loop, e K è la matrice dei guadagni ottimi. Questa matrice viene quindi implementata nel modello Simulink al paragrafo 3.4.3 nel blocco già rappresentato:

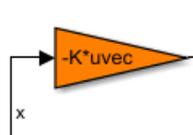


Figura 112 - Optimal Gain da LQR

Quindi, andando a variare le matrici dei pesi Q e R, andremo a variare questa matrice di controllo, che nel presente caso varia solamente i guadagni associati alle variabili errore dello spazio degli stati, quindi dell'angolo volante:

$$u = \delta = -[K](x) = -(k_{\Delta y}\Delta y + k_{\Delta \dot{y}}\Delta \dot{y} + k_{\Delta \psi}\Delta \psi + k_{\Delta \dot{\psi}}\Delta \dot{\psi})$$

Calibrazione e sensitività

Gli stati errori iniziali scelti sono:

$$(x) = [5 \quad 0 \quad \pi/8 \quad 0]$$

Vengono riportati i risultati di tre prove messe a confronto al variare dei pesi Q e R sul tracciato di prova generato dal path planner:

1° Prova) Matrici dei pesi scelte inizialmente uguali all'identità (stesso peso Performance ed Efforts):

$$Q = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \text{ ed } R = 1$$

Dal calcolo effettuato dalla funzione 'lqr' si ottiene:

$$K = [1 \quad 0.7074 \quad 3.4612 \quad 0.5086]$$

Il risultato di inseguimento e i relativi errori commessi:

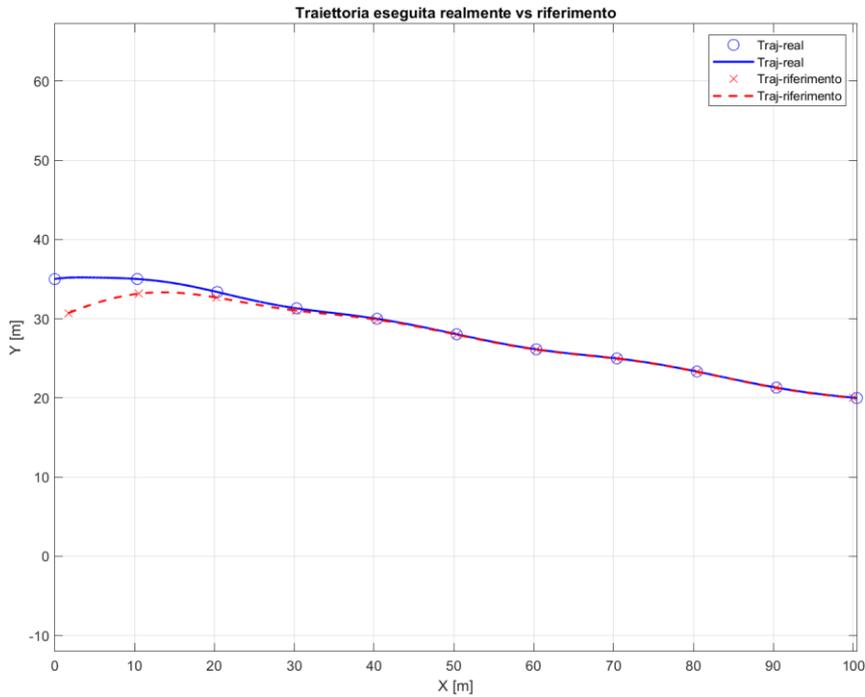


Figura 113 - Risultato inseguimento traiettoria 1° prova

Dopo circa 30 metri lungo la coordinata x viene raggiunta la traiettoria di riferimento. Questo perché non si è data una prioritaria importanza all'errore di posizione rispetto agli altri stati errore.

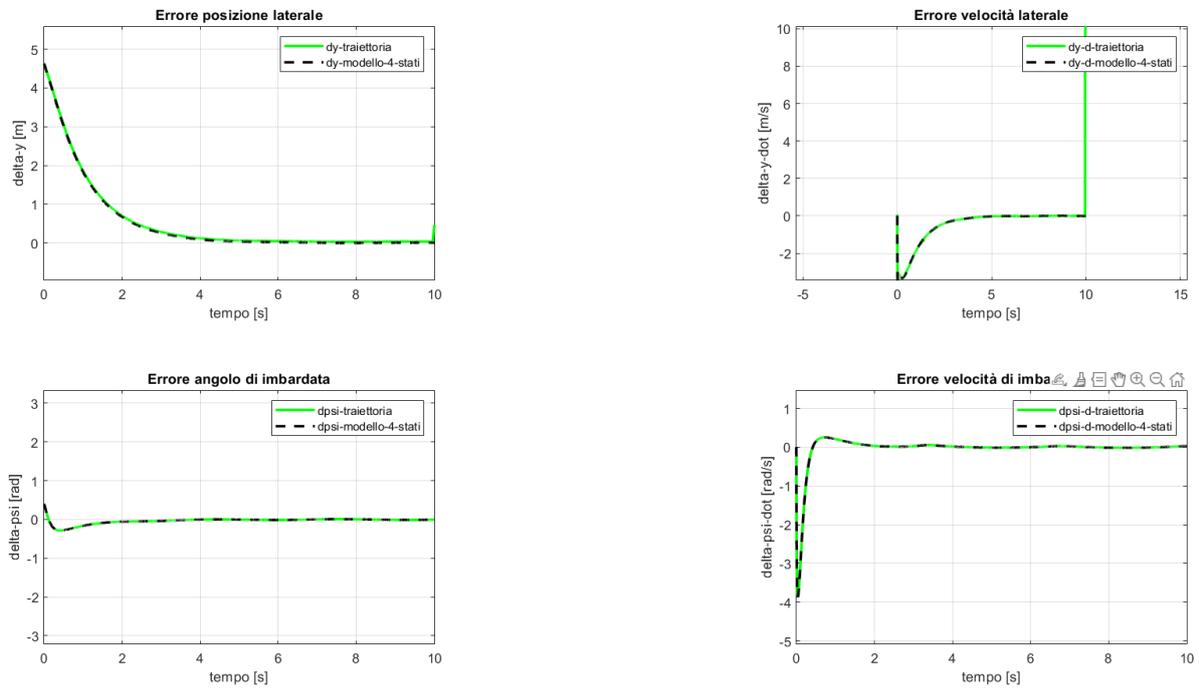


Figura 114 - Stati errore 1° prova

Si nota una buona corrispondenza tra gli stati errore valutati dal modello a quattro stati e quelli calcolati a partire dalla traiettoria eseguita realmente.

2° Prova) Matrici dei pesi stati errore mantenuti uguali e matrice dell'input di controllo diminuita

$$Q = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \text{ ed } R = 0.1$$

$$K = [3.1623 \quad 2.4754 \quad 8.2958 \quad 1.7387]$$

Si nota un sostanziale aumento dei guadagni, questo perché, dando meno importanza al costo di attuazione dell'angolo volante di input, si andranno a generare degli angoli di sterzo più nervosi ed accentuati:

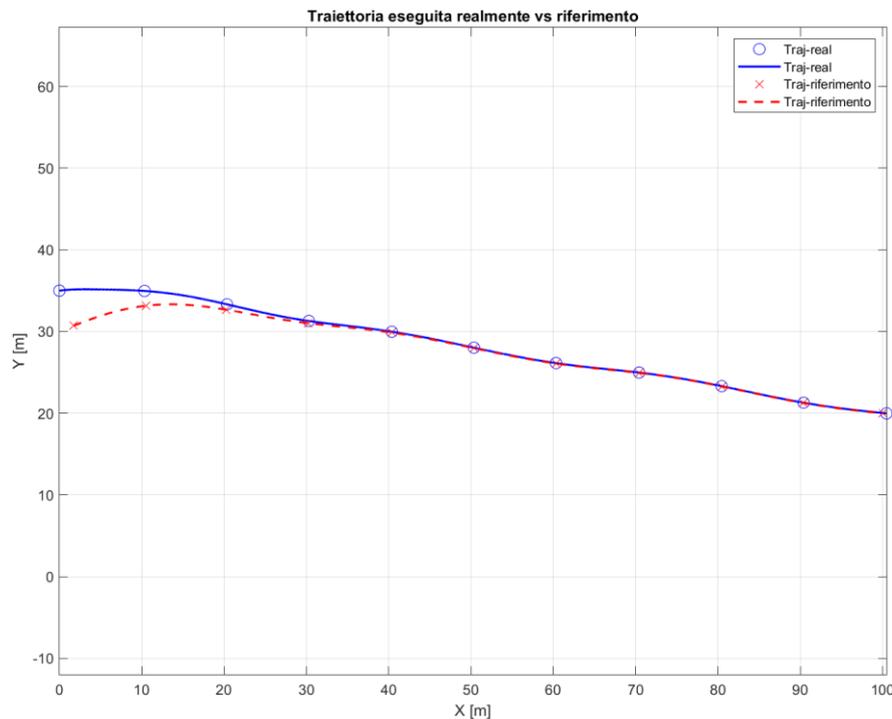


Figura 115 - Risultato inseguimento traiettoria 2° prova

Andando a ingrandire la traiettoria si nota una maggiore oscillazione del percorso eseguito, questo a fronte del fatto che l'angolo volante è molto più reattivo. Questa variazione ai pesi non ha portato, però, a un miglioramento nella persecuzione della traiettoria, infatti, si giunge al ricongiungimento con la traiettoria di riferimento sempre circa nello stesso punto.

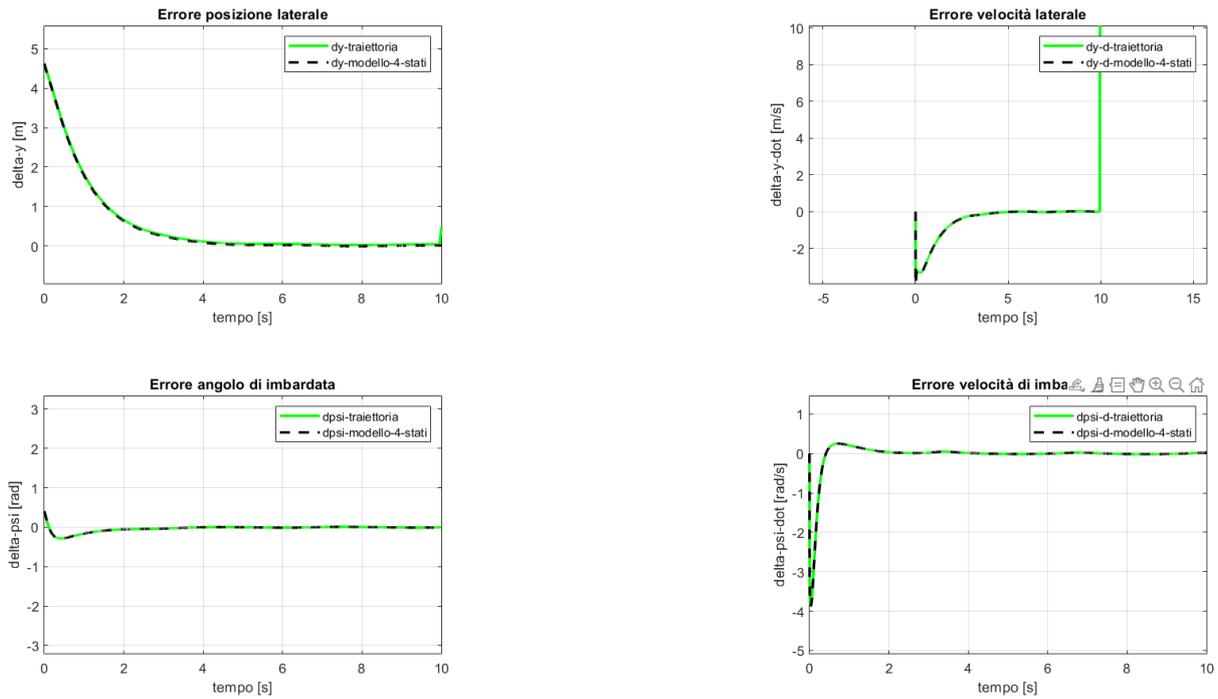


Figura 116 - Stati errore 2° prova

Anche gli andamenti degli stati errore non presentano miglioramenti rispetto a prima.

3° Prova) A seguito di numerose prove, qui non riportate, si è giunti a una soluzione che rendesse la traiettoria non troppo nervosa, ma che fosse molto reattiva nel ricongiungimento con la traiettoria di riferimento e nella successiva persecuzione. Le matrici dei pesi impostate sono:

$$Q = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 0.2 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0.2 \end{bmatrix} \text{ ed } R = 0.1$$

Anche la matrice K dei guadagni contiene dei valori intermedi (che andranno comunque diminuiti in ambiente IPG-CarMaker per non rendere troppo nervoso il controllo dello sterzo).

$$K = [3.1623 \quad 1.0660 \quad 4.9704 \quad 0.7287]$$

I risultati sono nettamente migliorati ricongiungendosi con la traiettoria di riferimento già a 10 metri lungo l'asse X:

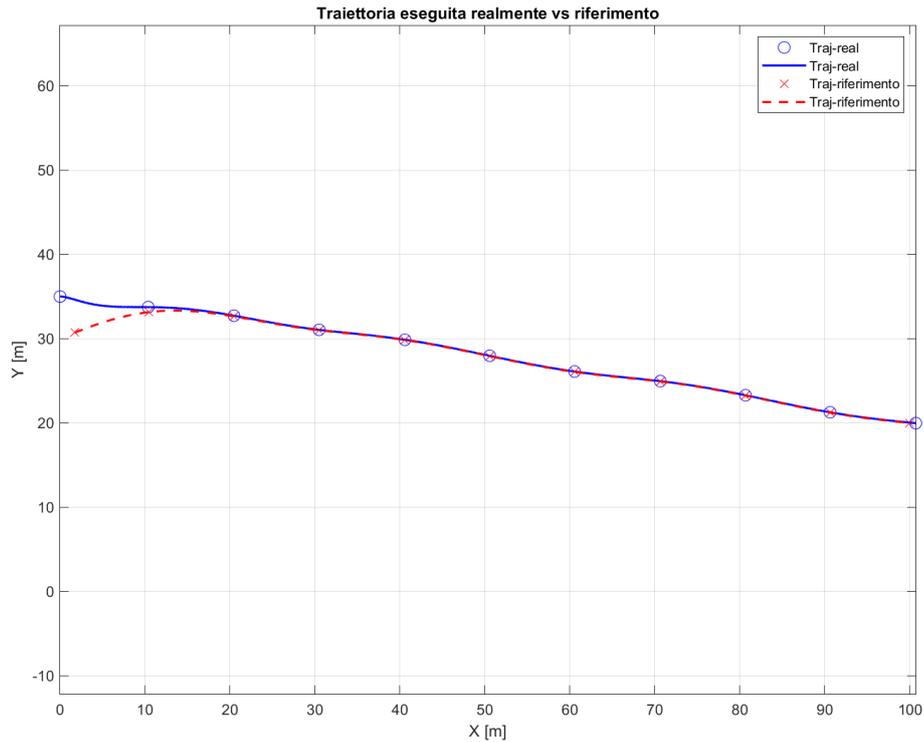


Figura 117 - Risultato inseguimento traiettoria 3° prova

Di conseguenza anche gli errori vanno più velocemente a 0:

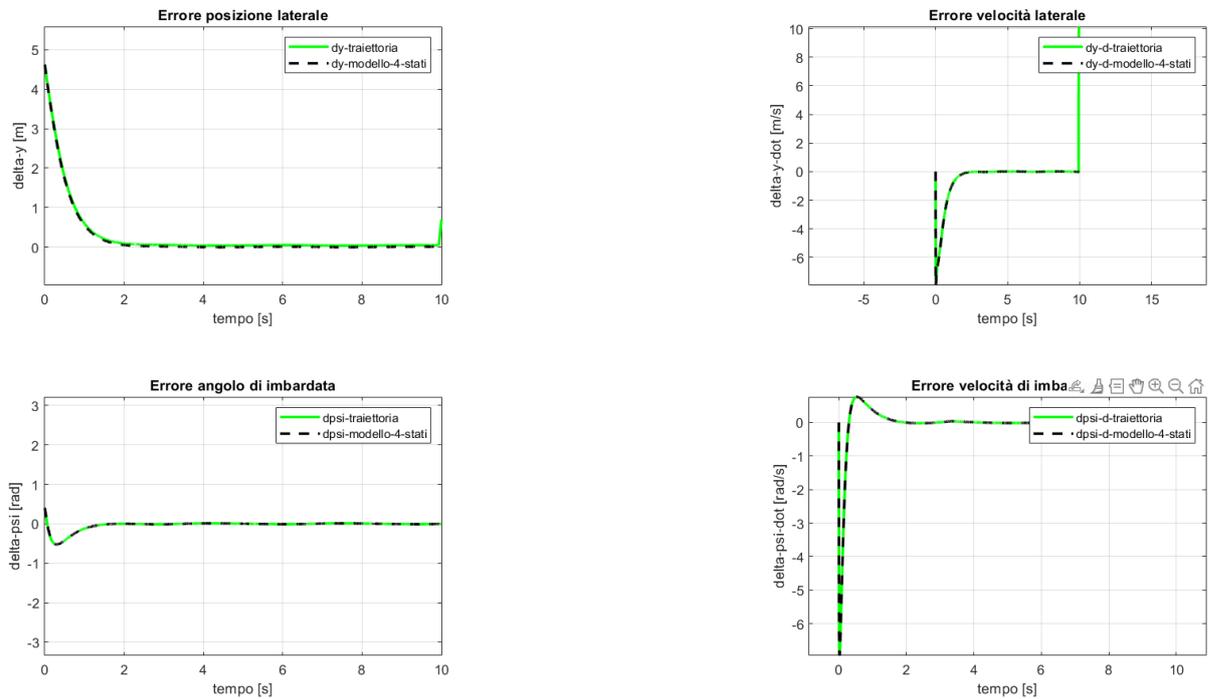


Figura 118 - Stati errore 3° prova

La matrice dei guadagni così ottenuta viene riportata nel modello Simulink che governa l'angolo di sterzo del veicolo a più gradi di libertà in CarMaker tramite l'interfaccia CarMaker-for-Simulink descritta nel capitolo 4.3.

4. Simulazioni CarMaker

Le simulazioni su IPG-CarMaker sono servite per validare gli algoritmi di path planning e path tracking implementati e ottimizzati nei precedenti due capitoli. Infatti, tramite un modello di veicolo di simulazione a molti gradi di libertà (che possono arrivare a ben 280 [36] nel caso di una modellazione Multibody estremamente accurata), si è potuto analizzare la risposta di un veicolo, pressoché reale, a tale algoritmo. Si tenga presente che le simulazioni virtuali stanno diventando sempre più aderenti alla realtà, questo permette una riduzione del numero di esperimenti condotti su veicoli fisici, quindi una riduzione dei costi di validazione.

4.1 Passaggio mappa Matlab a IPG-CM formato CRG con ASAM.CRG

Per permettere l'esecuzione della traiettoria di riferimento generata e ottimizzata dal path planner nel capitolo 2, si è deciso di utilizzare lo stesso scenario ad ostacoli in cui è stato creato il path di riferimento.

Il passaggio di una desiderata mappa, contenente determinati ostacoli, da ambiente Matlab ad ambiente IPG-CM avviene tramite l'interfaccia 'Scenario Editor' di IPG-CM, con una serie di step qui descritti sfruttando l'esempio di piazzale che è stato utilizzato nella sezione dedicata al path planning. Ovviamente il processo di trasferimento può essere applicato su una qualsivoglia mappa.

Per iniziare si deve generare una mappa in ambiente Matlab con coordinate $[X Y Z]$ tramite l'utilizzo delle funzioni che la MathWorks® mette a disposizione (e.g. 'peaks') oppure imponendo degli ostacoli, con specifica altezza, all'interno di una griglia delle dimensioni dell'ambiente in cui si vorrà eseguire una simulazione. Uno scenario di esempio è quello utilizzato nel capitolo legato al path planning in cui si ha un piazzale di larghezza 50 m, lunghezza 100 m in cui sono presenti una serie di dossi di altezza uguale o variabile:

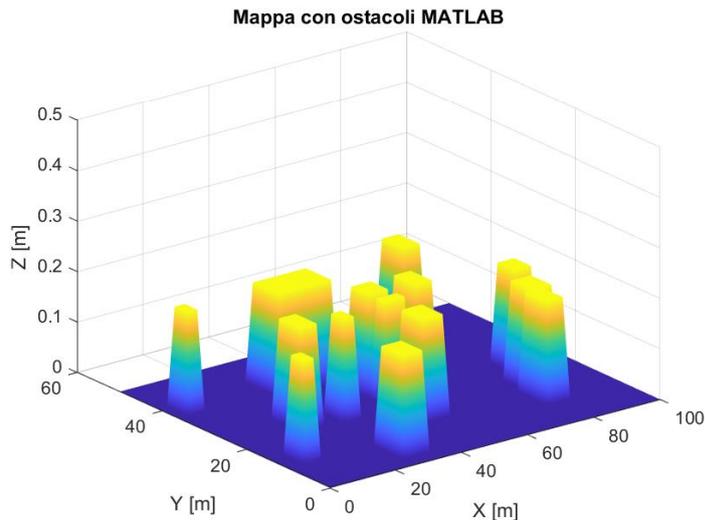


Figura 119 - Esempio mappa generata in Matlab

La proporzione degli assi non è la stessa, quindi si noti come l'altezza di questi ostacoli sia di soli 20 cm (poco più di un dosso). Tale altezza può essere, ovviamente, variata in base alle necessità di rappresentazione dello scenario che si vuole rappresentare.

Nota: Gli ostacoli che sono stati riprodotti in automatico per conversione e trasferimento da Matlab allo scenario di CarMaker si possono generare, allo stesso modo, direttamente nello 'Scenario Editor' messo a disposizione dal programma. Tuttavia, questo metodo, reso automatico, risulta essere di particolare comodità, quando si deve testare un algoritmo sviluppato in ambiente Matlab/Simulink, su CarMaker.

Proseguendo, una volta ottenuta la mappa su Matlab, si deve procedere alla trasformazione di tale dato in un formato compatibile allo Scenario Editor di IPG-CM. Tali formati sono:

- KML files da Google Earth o Google Maps
- OpenDRIVE (.xodr) files
- OpenCRG (.crg) files

Altre informazioni più approfondite sul metodo di comunicazione tra questi formati e IPG si trova nella 'User-Guide' di CarMaker [31].

CarMaker supporta anche alcuni altri formati meno rilevanti, mentre quello di base usato nelle mappe di esempio già presenti nel buffer del programma è 'Road5' (rd5). Tuttavia, il metodo di passaggio più rapido (oltre che a libero accesso) è quello di utilizzare il formato CRG.

Formato OpenCRG

Il formato CRG ("Curved Regular Grid") serve per la descrizione di superfici ad alta definizione con un basso tempo di generazione e ridotto utilizzo della memoria. I dati sull'elevazione vengono incamerati in una griglia (di righe e colonne) in cui ad ogni cella viene assegnato il proprio valore di elevazione.

Alla base di tale schema ci sono formati ASCII e Binari con intestazioni precise di informazioni legate alla strada da rappresentare.

Questo tipo di formato è stato creato per la descrizione precisa di profili stradali ad alta risoluzione, per il calcolo della sollecitazione dinamica/vibrazionale di componenti strutturali, quali sospensioni (si pensi a strade ciottolate di pavé o altamente sconnesse). Pertanto, le superfici che si possono descrivere, di default, sono legate a strade urbane e non a tracciati off-road. Tale limite viene agilmente sorvolato imponendo a posteriori, nello Scenario Editor di IPG-CM, una larghezza stradale pari all'ampiezza del piazzale "fuoristrada" in cui si vuole eseguire la simulazione. I vari passaggi verranno descritti nel dettaglio nel seguito.

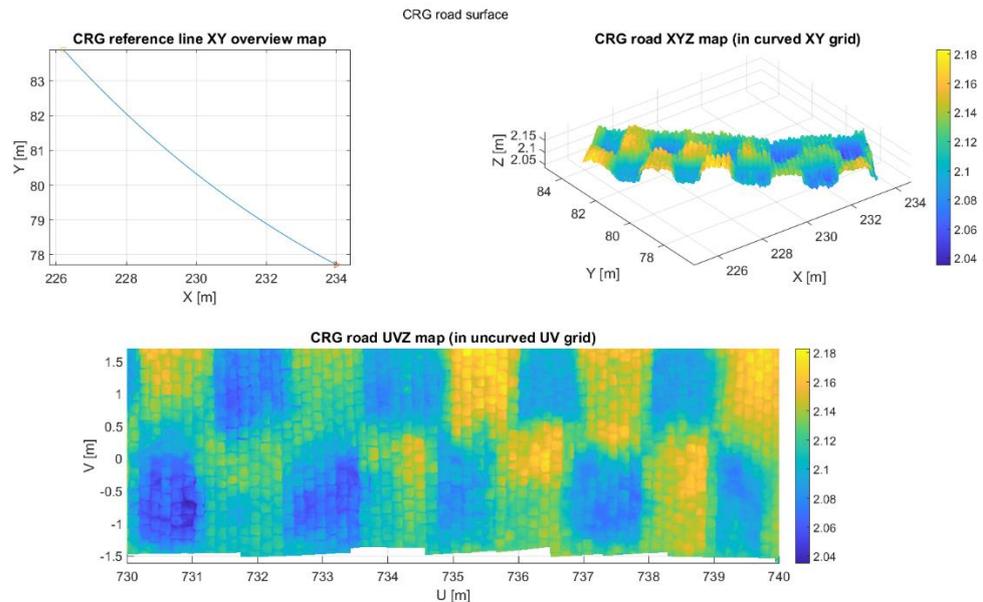


Figura 120 - Esempio utilizzo formato OpenCRG

Uno dei programmi, a libero accesso, per la generazione dei dati in formato crg è ASAM [37], che mette a disposizione un insieme di librerie contenenti codici/funzioni atti a svolgere operazioni su o per file crg.

Le librerie Matlab che si possono utilizzare sono utili a:

- Leggere file formato crg
- Impostare modificatori e opzioni varie su file crg
- Valutare dati OpenCRG
- Manipolare dati OpenCRG
- Generare dati OpenCRG (di interesse in questo lavoro)
- Visualizzare e analizzare dati OpenCRG
- Mappare dati OpenCRG rispetto a dati geografici
- Scrivere file OpenGCR (di interesse)

Le funzioni utilizzate per trasformare una mappa Matlab a una crg, generare e salvare un file formato crg si trovano nelle cartelle presenti al percorso: ASAM_OpenCRG > matlab > lib.

Il file formato crg contiene e comunica tramite una struttura chiamata (di default) 'data', all'interno della quale sono presenti una serie di informazioni, tra cui la dimensione dello scenario in cui simulare (50x100 m, in questo esempio), la suddivisione di tale area in una

griglia con celle di specifica dimensione e le informazioni di elevazione del terreno su ogni cella.

Le operazioni da compiere per l'attuazione di questo passaggio sono:

- 1) Impostare le variabili curvilinee relative alla dimensione della strada u e v (ovvero lo scenario di base) e la dimensione di una singola cella lungo la direzione u e v:

```
%% Generazione struttura [data] utilizzata nei file CRG
% Dimensioni scenario
u = [ 0 100 ]; % [m] lunghezza mappa
v = [ -25 25 ]; % [m] larghezza mappa
inc = [ 1 1]; % [m] dimensione elemento di griglia lungo u e v
```

Figura 121 - Creazione griglia di base scenario

- 2) Creazione elemento formato crg tramite la funzione `crg_gen_csb2crg0` e calcolo dimensioni di griglia nu e nv:

```
% Generazione elemento: passaggio informazioni in formato .crg
data = crg_gen_csb2crg0(inc, u, v);
% Dimensioni della matrice rappresentate la griglia contenente i valori di
% elevazione su ogni cella
[nu nv] = size(data.z);
```

Figura 122 – Creazione dato formato CRG

- 3) Creazione della mappa in Matlab tramite definizione di ostacoli o quote di elevazione, imposte sulle celle della griglia. Si riporta l'esempio degli ostacoli relativi alla prima figura:

```
%% Creazione quote/ostacoli nella mappa
a = 0.2;
x = 0:1:(nu-1);
y = 0:1:(nv-1);
[X,Y] = meshgrid(x,y);
Z = zeros(size(X));
Z(24:26,47:53) = a*ones;
Z(28:32,47:53) = a*ones;
Z(15:20,47:53) = a*ones;
Z(38:41,5:8) = a*ones;
% Z(24:26,60:65) = 2*ones;
Z(27:32,60:65) = a*ones;
Z(12:15,7:10) = a*ones;
Z(23:28,20:25) = a*ones;
Z(40:45,73:78) = a*ones;
%Z(37:42,40:46) = 2*ones;
Z(23:27,85:90) = a*ones;
Z(36:42,30:46) = a*ones;
Z(5:10,28:34) = a*ones;
Z(15:20,80:86) = a*ones;
Z(10:15,78:83) = a*ones;
Z(22:25,32:35) = a*ones;
```

Figura 123 - Creazione ostacoli scenario

- 4) Adattamento matrici alla struct 'data' e riduzione dei dati da formato double a single perché comunicano tramite tale formato:

```
% Traspongo Z per adattare le matrici data.z e Z1
Z1 = Z';
% Riporto, in formato single, le quote dei vari punti della mappa in data.z
data.z(1:nu,:) = single(Z1(1:nu,:));
% Creo il file crg e lo porto nella cartella di IPG-CM desiderata
[data] = crg_single(data); % Trasformo la struct 'data' in formato single
```

Figura 124 - Adattamento formati

- 5) Scelta nome file crg che si vuole generare e scrittura del file nella cartella di lavoro in cui è contenuto lo script che si sta utilizzando tramite la funzione 'crg_write'.

```
filename= 'prova2_303622.crg'; % Scrivo il nome del file .CRG che voglio creare e riportare su IPG_CM
ier = crg_write(data, filename); % Creo il file crg con il nome che voglio nella cartella che sto utilizzando.
% ier se è pari a 0 vuol dire che il file è stato salvato correttamente
% Copio questo file nella cartella di IPG_CM in cui si trovano i tracciati da simulare
path_iniziale = 'C:/Users/Utente/Dropbox (Politecnico Di Torino Studenti)/Il mio PC (DESKTOP-93VTJ2B)/Desktop/TESI LM';
path_destinazione = 'C:/IPG/carmaker/win64-11.0.1/Data/Road/Examples/Measured/TestTrack';
sorgente = fullfile(path_iniziale,filename);
destinazione = fullfile(path_destinazione,filename);
copyfile(sorgente,destinazione)
```

Figura 125 - Salvataggio file con relativo nome

Si è riportato, infine, il file crg nella cartella di IPG-CM in cui si andranno a selezionare gli scenari su cui effettuare la simulazione.

- 6) Una volta che si ha il file crg all'interno della cartella da cui si estrarrà, tramite lo Scenario Editor di IPG-CM, il profilo stradale desiderato, si apre, appunto, lo 'Scenario Editor di IPG andando su: Parameters > Scenario/Road. Una volta che ci si trova all'interno di questo ambiente di creazione di scenari di simulazione si seleziona la casella: Road Segment > File:

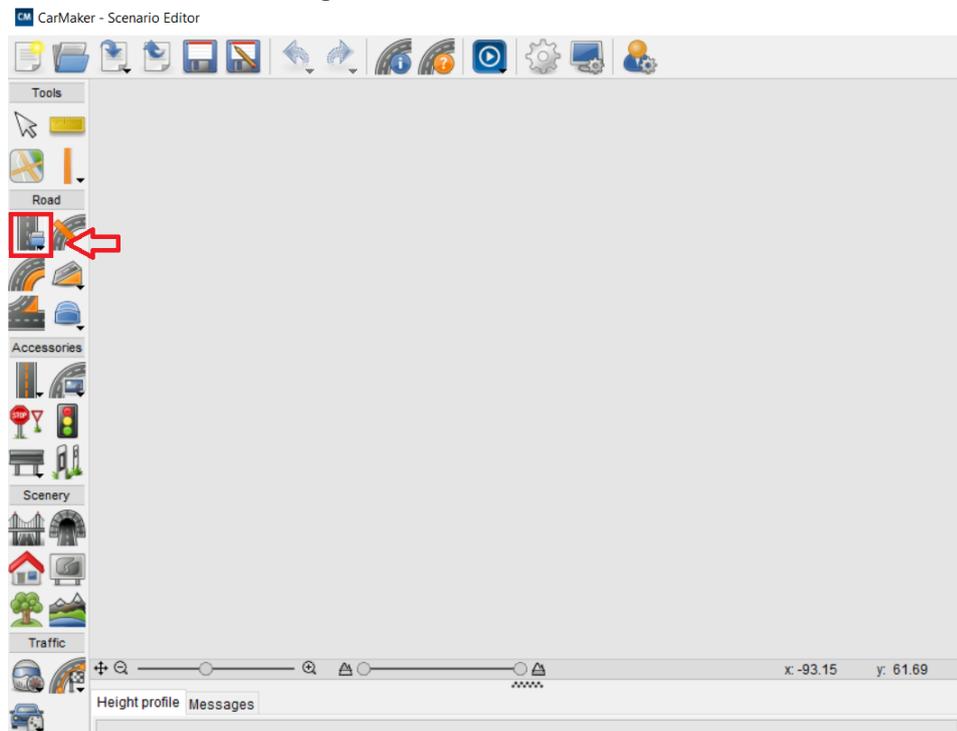


Figura 126 - Interfaccia Scenario Editor

- 7) Con il cursore centrale del mouse si traccia una linea orizzontale che determini l'orientazione originale/ di partenza della strada su cui si vorrà riportare il profilo stradale desiderato in formato crg:

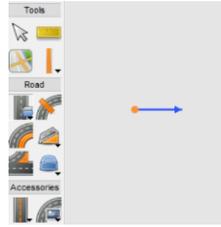


Figura 127 - Direzione asse stradale

- 8) Si aprirà una schermata di dialogo su cui si andrà a selezionare il file crg in cui si è salvato in precedenza lo scenario di simulazione:

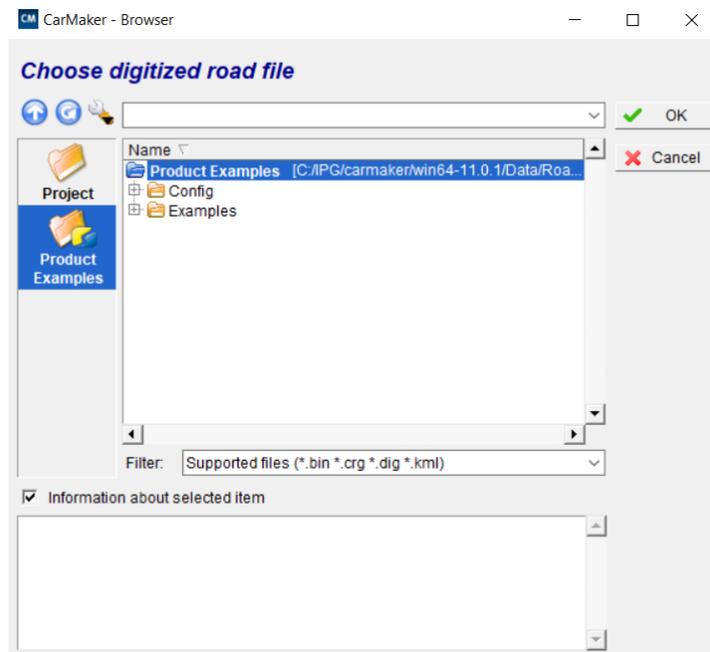


Figura 128 - Scelta tracciato di simulazione

- 9) Selezionando Product Exemples > Examples > Measured > TestTrack > prova2_303622.crg (questo percorso è stato eseguito sulla base della posizione in cui è stato salvato il file crg, ma risulterà differente a seconda della posizione di salvataggio prescelta).

Si aprirà, quindi, la seguente strada sormontata da un profilo ad ostacoli corrispondente a quello generato in Matlab:

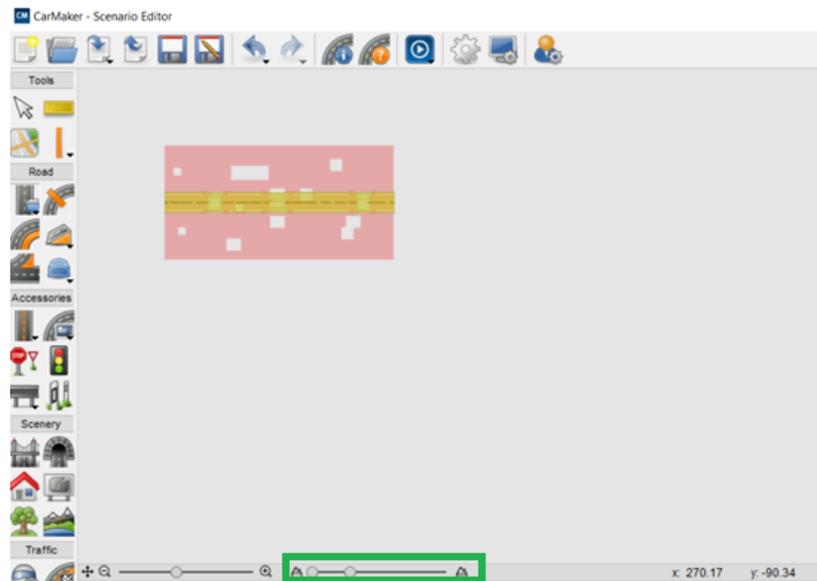


Figura 129 - Profilo stradale riportato nello Scenario Editor

Sono stati messi in evidenza le sommità degli ostacoli, per mostrare come, nonostante la strada standard di default sia più stretta della larghezza complessiva dello scenario che si sta considerando (50 m), è comunque presente il profilo stradale desiderato.

Per ovviare a questo problema si procede all'allargamento delle due carreggiate della strada di sottofondo nella casella di lavoro 'Object List':

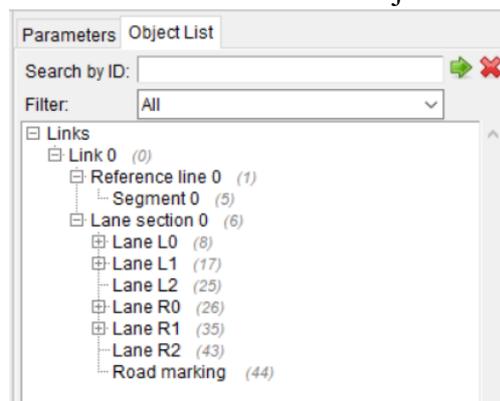


Figura 130 - Object List dello Scenario Editor

Selezionando 'Lane L0' e successivamente 'Lane R0' si può procedere all'allargamento delle carreggiate di sinistra e destra (in questo caso entrambe di 25 m al posto dei 3,5 di default)

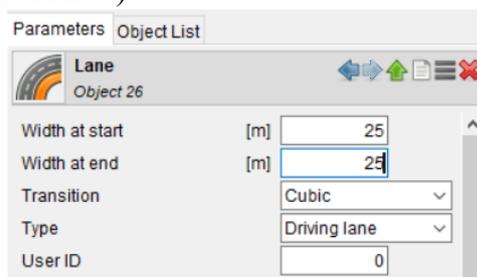


Figura 131 - Allargamento carreggiata

10) Si ottiene la seguente strada e il corrispettivo profilo superficiale:



Figura 132 - Scenario definitivo

Andando a visualizzare il risultato all'interno di una simulazione, si vedranno, in definitiva, gli ostacoli (i dossi in questo caso) stradali desiderati:



Figura 133 - Rappresentazione dello scenario di prova

La larghezza e lunghezza della strada può essere configurata a piacimento all'interno delle impostazioni dello Scenario Editor.

Nella prova finale di simulazione la strada è stata allargata e allungata per evitare eventuali messaggi di errore nel momento in cui il veicolo non aderisse perfettamente al percorso di riferimento.

4.2 Configurazione simulazione finale su IPG CarMaker

Al fine di testare le logiche di path planning e di path tracking (descritte nei capitoli 2 e 3) è stato necessario l'utilizzo del pacchetto 'CarMaker for Simulink' che permette di modificare gli algoritmi alla base del controllo dei veicoli di prova di CarMaker. Nello specifico è stato variato il modello di controllo dell'angolo volante (capitolo 4.3) e dell'acceleratore/freno (capitolo 4.4) per poter eseguire, secondo richiesta, le manovre desiderate.

Di seguito si riporta una descrizione delle variazioni apportate alle impostazioni di CarMaker che hanno permesso l'esecuzione delle manovre da eseguire.

CarMaker for Simulink

Per prima cosa è stato necessario utilizzare l'interfaccia CarMaker for Simulink aprendo la cartella di Matlab: 'C:/CM_Projects/Progetti/src_cm4sl', dove l'indirizzo 'Progetti' è il

Project folder in cui è stato deciso di salvare gli output (in 'SimOutput') delle simulazioni CarMaker. In 'src_cm4sl' si trova di default un file Simulink (formato. mdl) denominato 'generic.mdl'. Aprendolo si ottiene la seguente interfaccia:

CarMaker 11.0.1

Generic Car Model

You may use this model as a starting point.
It already contains everything to run a simple CarMaker simulation.
Build your Simulink model around it.



Figura 134 - Interfaccia modello Simulink-CarMaker

Si possono selezionare le tre caselle:

- La casella a sinistra con il disegno degli ingranaggi, una volta selezionata fa comparire un'interfaccia in cui è necessario inserire sotto il 'Command line arguments' la dicitura '-disablevehiclecontrol' in modo da comunicare al sistema di controllo che degli algoritmi esterni, creati dall'utente, saranno utilizzati nella gestione del veicolo durante le manovre da eseguire.
- Nella casella centrale si trova l'interfaccia Simulink di governo alla base del veicolo CarMaker:

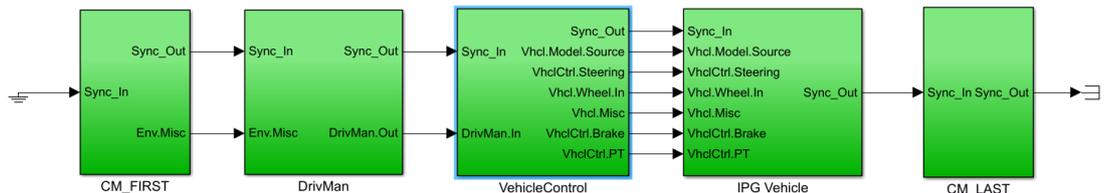


Figura 135 - Interfaccia Simulink CarMaker for Simulink

In particolare, all'interno del sottosistema 'VehicleControl' si andranno a sviluppare le logiche di controllo che verranno descritte nei capitoli 4.3 e 4.4.

- Si può, infine, aprire la main GUI di CarMaker selezionando la casella più a destra. L'interfaccia che si aprirà è quasi identica a quella che si visualizza quando si apre IPG-CarMaker direttamente dal desktop. L'unica differenza è la diversa configurabilità di alcune sue funzionalità che verranno sfruttate nell'implementazione del controllo di sterzo e di velocità.

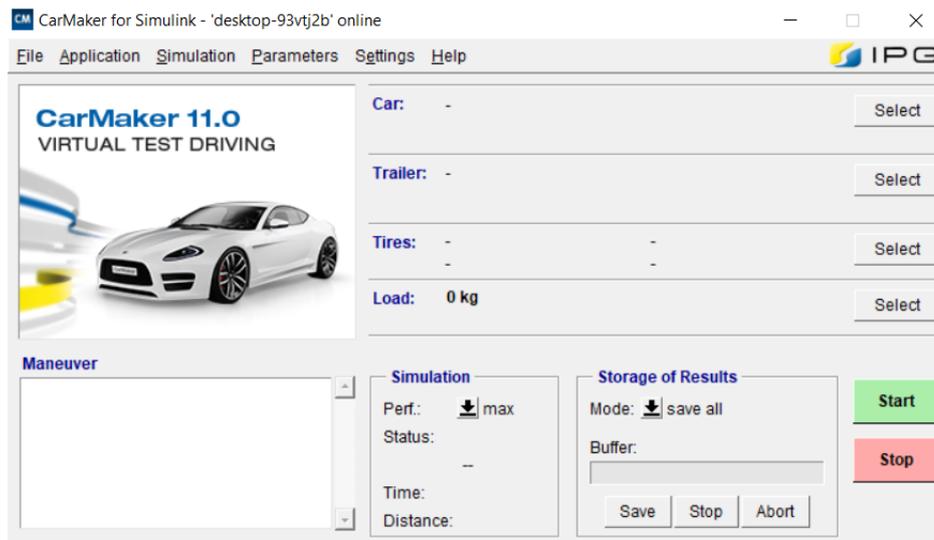


Figura 136 - GUI CarMaker for Simulink

Configurazioni Pre-Simulazione

Nella main GUI, andando su Parameters > Scenario/Road si entra nello scenario editor in cui si può ricostruire, secondo quanto indicato al capitolo 4.1, la mappa di prova degli algoritmi da testare. In questa simulazione finale il piazzale è stato allargato e allungato per evitare un'interruzione della simulazione qualora il veicolo uscisse leggermente dal perimetro di base (100x50 m).

Per permettere libertà di azione alla logica di controllo sterzo si è dovuto evitare di definire un 'Active Route' (una linea di riferimento che coincide, di default, con il centro delle carreggiate), pertanto selezionando 'Scenario settings' nella rotellina icona in alto si sono impostate le seguenti caratteristiche:

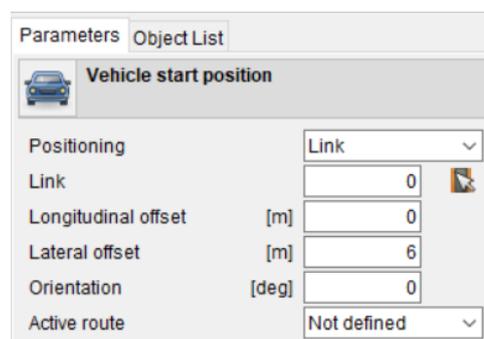


Figura 137 - Deselezionare Active Route

Nota: è importante definire, inizialmente una 'Route' che può coincidere, ad esempio, con il centro della carreggiata di destra. In caso contrario il programma non permette l'esecuzione della manovra. Infine, ricordarsi di selezionare, come descritto nell'immagine soprastante, 'Not defined'.

Ora il veicolo non è costretto a inseguire una traiettoria rettilinea predefinita, ma può sterzare e muoversi liberamente cercando di perseguire la traiettoria di riferimento che gli viene suggerita tramite l'angolo volante (nel modo che verrà descritto nel capitolo 4.3).

Successivamente, sempre nella main GUI, in ‘Maneuver’ si devono impostare le caratteristiche generali di esecuzione della manovra. Nel caso in esame, si vuole testare, tramite la logica di path tracking, precedentemente sviluppato, il tracciato generato dal path planner. I risultati della simulazione saranno riportati nel capitolo 4.5. Pertanto, in Maneuver si deve impostare:

- Le condizioni iniziali della manovra: In ‘Global settings/ preparation:

Start Values			
Velocity	[km/h]	36.58	Steer Angle [deg]
Gear		1	Track Offset [m]
Start Condition			f(x)
Vehicle Operator	↓	IPG Operator	<input checked="" type="checkbox"/> active
Operator State	↓	drive	

Figura 138 - Condizioni iniziali manovra su CarMaker

In questo caso si doveva solo impostare la velocità iniziale desiderata, mentre la marcia (‘Gear’) è la prima poiché è stato utilizzato un veicolo elettrico privo di cambi di velocità.

Al più si può imporre un ‘Track Offset iniziale’, a seconda dell’errore di posizione iniziale che si vuol far testare.

- Il tempo di percorrenza della traiettoria: in questo caso è stato di dieci secondi. Ma si è liberi di scegliere la velocità di percorrenza, quindi il tempo di compimento del tracciato.
- Le caratteristiche del Driver sia in dinamica longitudinale che laterale:

Specification of Maneuver Step		
Label / Attr. (Long/Lat)		
Description		
End Condition		f(x)
Duration (time/dist)	10 s	m Adjust
Longitudinal Dynamics		
↓		
Driver Parameter...		
Lateral Dynamics		
↓		
Driver Parameter...		

Figura 139- - Impostazioni Driver dinamica longitudinale e laterale

È stato necessario lasciare libertà di manovra sia sul longitudinale che sul laterale facendo agire gli algoritmi descritti in 4.3 e 4.4, selezionando ‘not-specified’ tra le diverse possibili opzioni che compaiono cliccando sulle frecce nere visibili nell’immagine soprastante.

Il modello di veicolo scelto per la manovra è la Tesla-S irrigidita secondo quanto descritto nel capitolo 3. Alcune configurazioni di base, relative al controllo dello sterzo e dell’acceleratore sono state modificate nel modo seguente:

Con un doppio clic sul nome del veicolo, nella main GUI, appare il ‘Vehicle Data Set’:

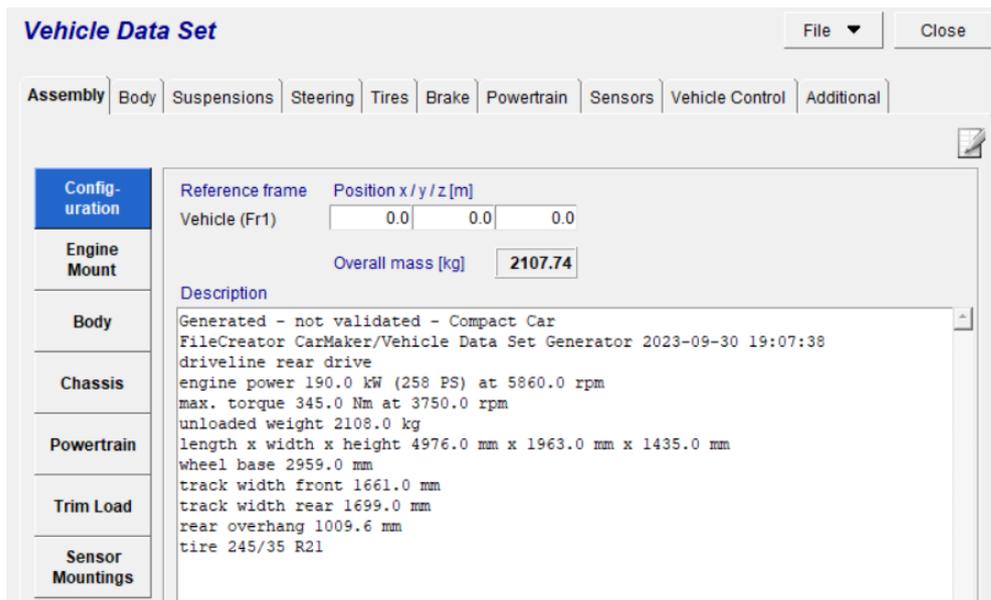


Figura 140 - Vehicle Data Set CarMaker

- Selezionando tra le opzioni in alto ‘Steering’ si ottiene un interfaccia al modello di angolo di sterzo in cui è necessario inserire un modello di sterzo basato sulla generazione dell’angolo volante (Static Steer Ratio – GenAngle) e non sulla coppia di sterzo:

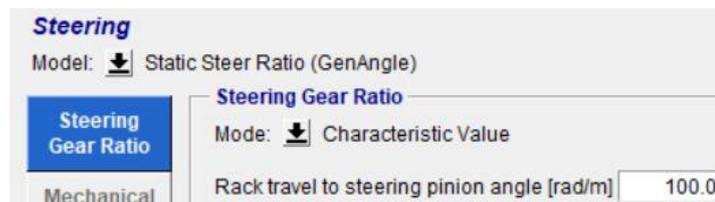


Figura 141 - Steering model

- Selezionando, invece, ‘Powertrain’ in ‘General’ lasciare un modello di convenzionale (‘Conventional’) di propulsore.
- In ‘Drive Source’ > Engine > Model: Look-Up Table
- In ‘Clutch’ > Model: Closed
- In ‘Gearbox’ > Model: Automatic with Converter
- Selezionando, infine, ‘Control Unit’ dare le seguenti impostazioni:
 - In ‘PT Control’ > Model: Generic, e aggiungere ‘Gas interpreter’
 - In ‘ECU’ > Model: Basic
 - In ‘TCU’ > Model: Automatic with Converter
 - In ‘MCU’ e in BCU’ lasciare i modelli di default

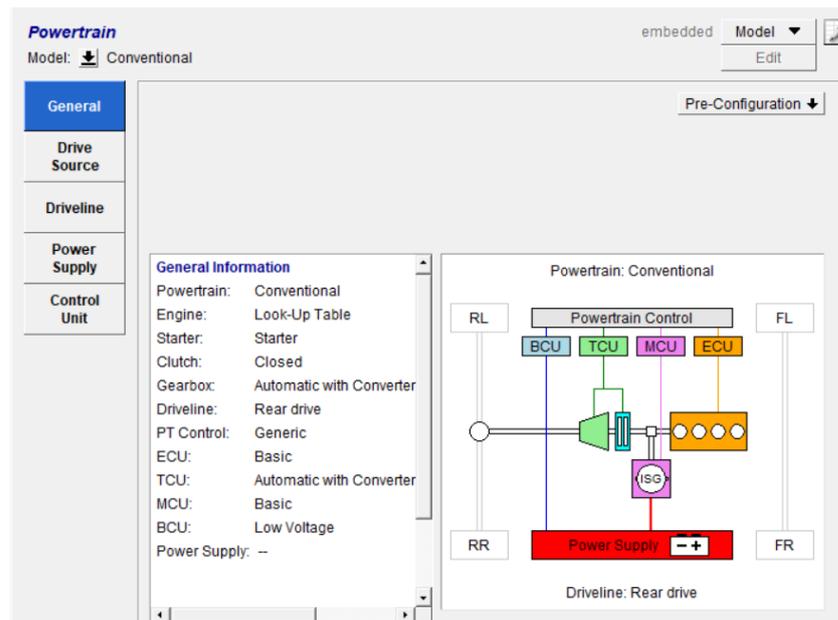


Figura 142 - Interfaccia generale Powertrain CarMaker

Queste impostazioni di base possono essere eseguite direttamente prima o a seguito dell'implementazione dei controlli di sterzo e di velocità descritti nei prossimi due capitoli. Ricordarsi, infine, di salvare il Test Run appena creato per potervi accedere velocemente ogni qualvolta si voglia eseguire la simulazione, senza dover seguire lo stesso iter tutte le volte.

4.3 Implementazione controllo sterzo

Accedendo al modello Simulink del veicolo CarMaker in 'VehicleControl' si trova la seguente struttura:

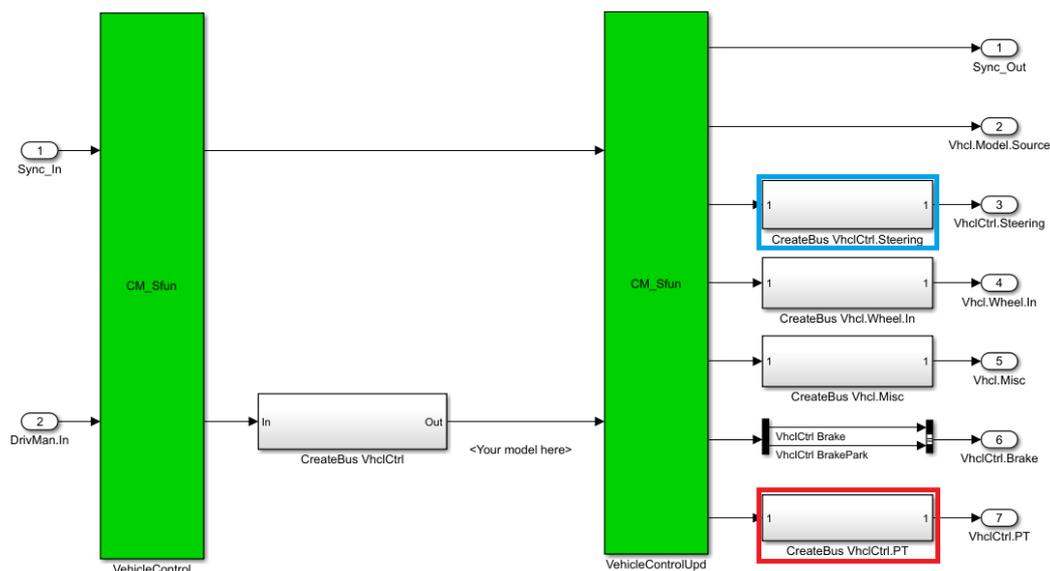


Figura 143 - VehicleControl Subsystem

All'interno del sottosistema 'CreateBus VhclCtrl' oppure in 'CreateBus VhclCtrl.Steering' e 'CreateBus VhclCtrl.PT' si possono implementare il controllo dell'angolo di sterzo e di velocità che si desidera testare. In questo capitolo viene descritto il Controllo di sterzo implementato in 'CreateBus VhclCtrl.Steering' (nel riquadro azzurro). Nel capitolo 4.4 si descriverà il controllo di velocità 'CreateBus VhclCtrl.PT' (nel riquadro rosso). Pertanto, in 'CreateBus VhclCtrl' è stato lasciato tutto com'era inizialmente.

Selezionando 'CreateBus VhclCtrl.Steering' si può osservare il modello di controllo dell'angolo di sterzo e delle sue derivate:

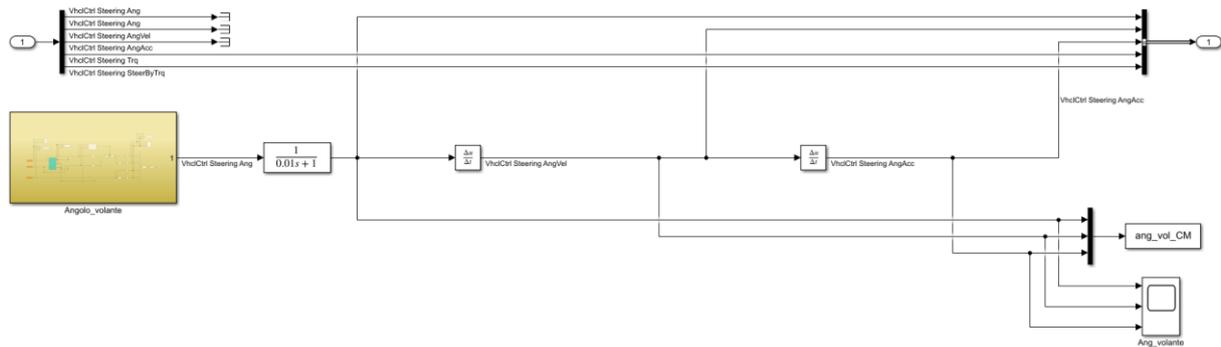


Figura 144 - Controllo angolo volante e derivate

Come si può notare il blocco 'Angolo_volante' (che è il cuore del controllo di sterzo) restituisce l'angolo volante (in radianti) che andranno applicati dal Driver durante la manovra di persecuzione della traiettoria di riferimento. In questa struttura vengono solamente calcolate le derivate di 'VhclCtrl SteeringAng': 'VhclCtrl Steering AngVel' e 'VhclCtrl Steering AngAcc' e riportate nel Workspace di Matlab per poi essere analizzate.

Un filtro è stato posto sull'angolo volante a una frequenza di taglio di 100 Hz per eliminare oscillazioni numeriche indesiderata che obbligavano il driver a dare dei colpi di sterzo immotivati dal punto di vista della guida e irrealizzabili nella pratica.

Sottosistema 'Angolo_volante':

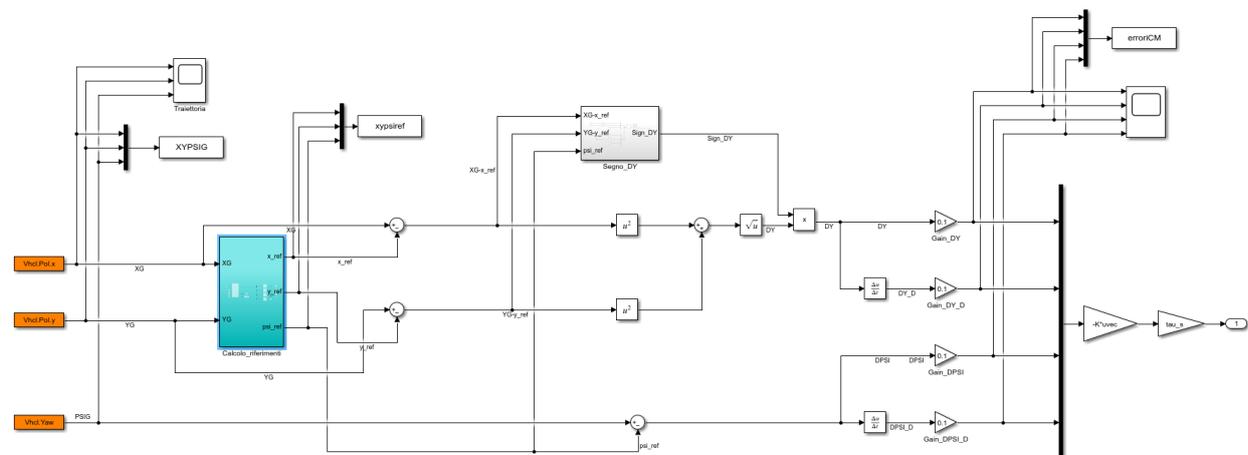


Figura 145 – Simulink generazione angolo volante di controllo

In questo schema viene calcolato l'angolo volante (output del sistema) a partire dalla posizione del veicolo di simulazione su CarMaker. Le coordinate che determinano la

posizione del veicolo ad un dato istante di tempo: $[X_G, Y_G, \psi_G]$ vengono estratte tramite dei blocchi Simulink ‘Read CarMaker Dictionary Variables’ direttamente dal veicolo in simulazione. Tali coordinate sono quelle del baricentro del veicolo, quindi, facendo riferimento alla nomenclatura delle variabili CarMaker:

$$[X_G, Y_G, \psi_G] = [Vhcl.PoI.x, Vhcl.PoI.y, Vhcl.Yaw]$$

Le coordinate X_G e Y_G vengono inviate dentro il sottosistema ‘Calcolo_riferimenti’ identico a quello sviluppato e descritto al paragrafo 3.4.3, in cui, a partire dalla posizione attuale del veicolo, si calcolano le coordinate del punto più vicino appartenente alla traiettoria di riferimento (che è stata generata dal path planner). In uscita ci sono le coordinate di riferimento $[X_{ref}, Y_{ref}, \psi_{ref}]$ che verranno sottratte alle coordinate del baricentro del veicolo per ottenere i vettori errore:

$$[X_G - X_{ref}, Y_G - Y_{ref}, \psi_G - \psi_{ref}]$$

Nel calcolo dell’errore di posizione laterale locale del veicolo rispetto la traiettoria di riferimento si è dovuto tenere conto del segno di tale errore: considerato, per convenzione, positivo se il veicolo si trova alla sinistra della sua proiezione sul tracciato di riferimento, negativo se sulla destra.

Per ricavare il segno di Δy si è preso spunto dal riferimento [27] che utilizza il seguente prodotto vettoriale e ne calcola il segno secondo la seguente formulazione (3.54) e (3.55):

$$\Delta y = \sqrt{(X - X_{ref})^2 + (Y - Y_{ref})^2} \text{Sign}(\Delta y^*)$$

$$\Delta y^* = [(X_{ref} - X) (Y_{ref} - Y) 0] \times [\cos \psi_{ref} \sin \psi_{ref} 0]$$

Dove Δy^* è un vettore lungo la direzione perpendicolare al piano di simulazione $(X - Y)$. Tale prodotto vettoriale è stato eseguito in Simulink tramite un blocco ‘Cross Product’, e il segno del vettore risultante si calcola tramite il blocco ‘Sign’ della sua terza componente (lungo z). Il calcolo è stato implementato nel subsystem ‘Segno_DY’:

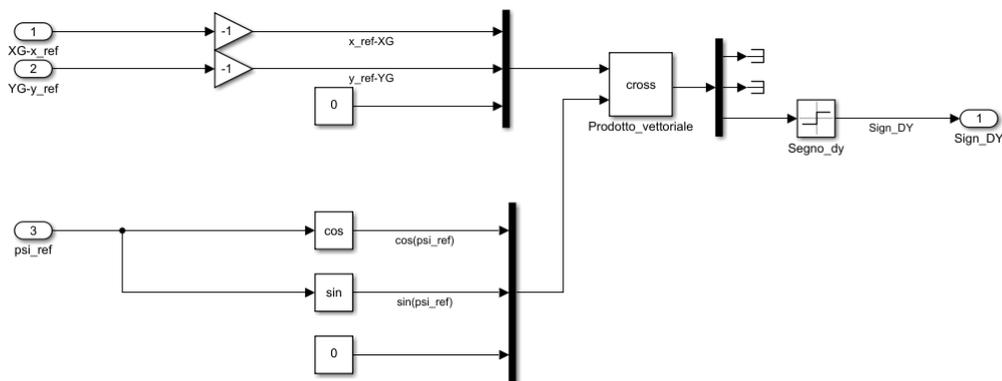


Figura 146 - Calcolo segno errore posizione laterale locale

L’errore Δy è stato quindi derivato per ottenere $\Delta \dot{y}$.

L’errore di angolo di imbardata è stato calcolato semplicemente come:

$$\Delta \psi = \psi_G - \psi_{ref}$$

Ed è stato derivato, ricavando $\Delta\dot{\psi}$.

Il vettore degli stati errore è stato, quindi, moltiplicato per l'inverso della matrice/vettore dei guadagni ottimi K , calibrata nel paragrafo 3.4.5. È stato necessario moltiplicare l'angolo di sterzo così ottenuto per il rapporto di trasmissione dello sterzo, poiché:

$$VhclCtrl SteeringAng = \delta_{SW} = \tau\delta = \tau(-Kx) \quad (4.1)$$

Dove δ_{SW} è l'angolo volante che il Driver deve applicare durante la manovra, τ è il rapporto di sterzo, δ è l'angolo di sterzo, $x = [\Delta y, \Delta\dot{y}, \Delta\psi, \Delta\dot{\psi}]'$ è il vettore degli stati errore.

Guadagni correttivi per la matrice dei guadagni ottimi K

Come si può notare da questo ingrandimento del sistema 'Angolo_volante':

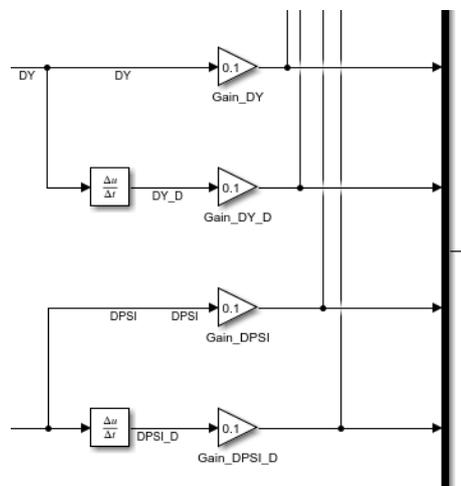


Figura 147 - Guadagno correttivi matrice dei guadagni ottimi LQR

È stato necessario inserire quattro guadagni costanti $[G_{\Delta y}, G_{\Delta\dot{y}}, G_{\Delta\psi}, G_{\Delta\dot{\psi}}]$ per ridurre i guadagni ottimi calcolati con la logica LQR che risultano essere troppo grandi al fine di ottenere una buona aderenza del percorso eseguito al tracciato di riferimento. Senza l'introduzione di essi si otteneva una risposta dello sterzo troppo ampia, aggressiva e oscillatoria.

Eventuali funzioni di trasferimento, con frequenze di taglio opportunamente calibrate, potrebbero portare a risultati ancora migliori di esecuzione dei test di prova.

4.4 Implementazione controllo velocità

4.4.1 Controllo di velocità con riferimento off-line

Inizialmente la manovra voleva essere fatta eseguire tramite il controllo di velocità già implementato in CarMaker 'Speed control', ma si poteva notare una forte oscillazione della velocità di avanzamento. Andando ad analizzare lo sfruttamento del Gas che il veicolo era in grado di dare ci si è accorti del fatto che neanche il 50% della potenzialità del motore veniva utilizzata. Pertanto, un semplice controllo Proporzionale di velocità è stato implementato all'interno del già citato sottosistema 'CreateBus VhclCtrl.PT':

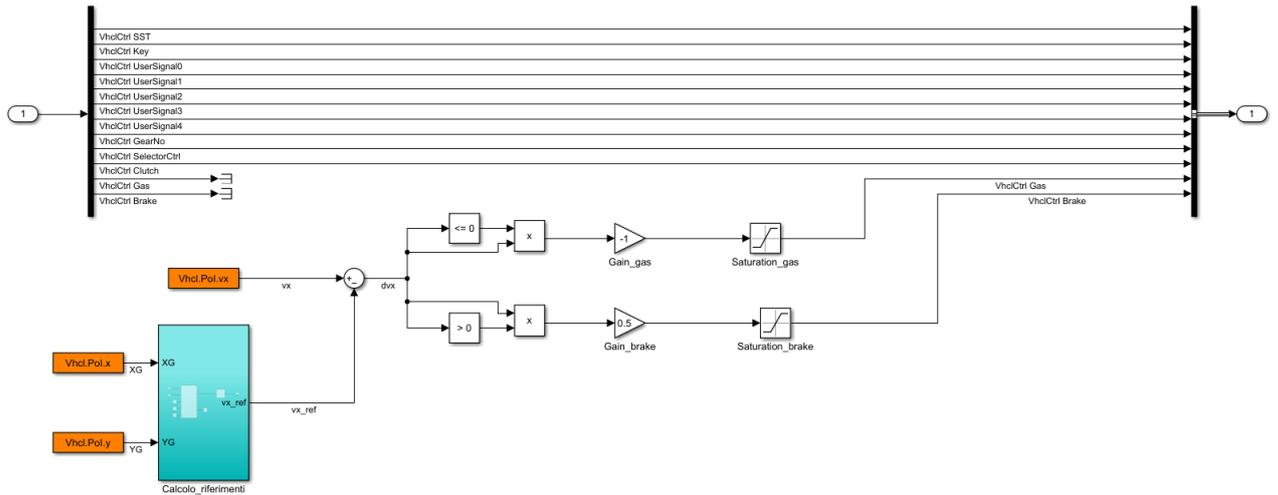


Figura 148 - Controllo proporzionale acceleratore e freno – off-line di velocità

Tramite il sottosistema ‘Calcolo_riferimenti’ è stata calcolata la velocità longitudinale desiderata di riferimento relativa al punto, appartenente alla traiettoria generata dal PP, più vicino alla posizione attuale del veicolo di simulazione $[X_G, Y_G] = [Vhcl.Pol.x, Vhcl.Pol.y]$:

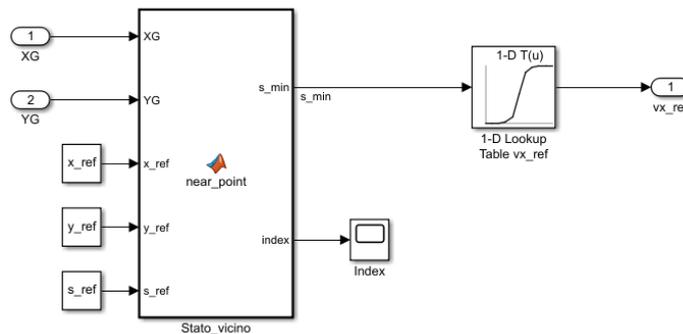


Figura 149 - Calcolo velocità longitudinale di riferimento

Tramite la Look-Up Table $[s_{ref}, vx_{ref}]$ si ottiene la velocità longitudinale desiderata in ogni singolo istante di tempo alla data coordinata curvilinea s_{ref} . La funzione ‘Stato_vicino’ è identica a quella già ampiamente descritta nel capitolo 3.4.3.

Come è noto, il controllo Proporzionale (P) si basa sul calcolo dell’errore della variabile che si vuole controllare, quindi sulla sua moltiplicazione per un opportuno guadagno al fine di generare un’azione di controllo che tenda a minimizzare questa discrepanza. L’errore in questo caso è:

$$\Delta vx = vx_G - vx_{ref} = Vhcl.Pol.vx - vx_{ref} \quad (4.2)$$

Pertanto, se l’errore di velocità longitudinale è positivo, vuol dire che è necessaria un’azione frenante, se, invece, l’errore è negativo, la velocità del veicolo è inferiore a quella desiderata, quindi, ci va un’azione accelerante. Sono stati impostati dei guadagni $[G_{gas}, G_{brake}]$ più o meno elevati, in modo da ottenere un controllo più o meno reattivo a dati scostamenti dalla velocità di riferimento. Nel caso specifico sono stati impostati i seguenti guadagni:

$$\begin{cases} \text{if } \Delta vx \leq 0 \rightarrow Gas = G_{gas} \Delta vx = -1 \Delta vx ; Brake = 0 \\ \text{else if } \Delta vx > 0 \rightarrow Gas = 0 ; Brake = G_{brake} \Delta vx = 0.5 \Delta vx \end{cases} \quad (4.3)$$

$G_{gas} = -1$ solo per il fatto che nel caso in cui è necessario attivare il gas, $\Delta vx < 0$, e i valori di G_{gas} vanno da 0 a 1, quindi è necessario rendere $Gas > 0$.

Sia i valori di Gas che di Brake sono stati saturati tramite dei blocchi ‘Saturation’ poiché essi devono essere compresi tra 0 e 1, dato che rappresentano la frazione di acceleratore e freno, rispettivamente, di gas e freno attuati.

Eventuali aggiustamenti dei guadagni vengono fatti per verificare le differenze nell’esecuzione della traiettoria, ma in generale dei guadagni troppo alti portano a un comportamento eccessivamente oscillatorio della velocità, mentre guadagni troppo bassi rendono il veicolo eccessivamente lento a dare delle risposte di controllo.

Un eventuale controllo PI o PID potrebbe essere costruito al fine di controllare con più accuratezza la velocità del veicolo di simulazione, ma non coinciderebbe con l’obiettivo del presente lavoro, che mira a testare modelli di path planning e path tracking nell’esecuzione di una data traiettoria.

4.4.2 Controllo di velocità con riferimento on-line

Un sistema di controllo della velocità alternativo è stato creato per la persecuzione della traiettoria di riferimento con l’obiettivo di mantenere, non una data velocità preimpostata dal path planner (che potesse essere costante lungo tutta il percorso o variabile), bensì per cercare di mantenere l’accelerazione laterale del veicolo ($a_{y,max}$) sempre sotto un dato limite desiderato. Questo tipo di controllo permette, in teoria, l’esecuzione di traiettorie (nel piano) di qualsiasi curvatura in sicurezza, evitando di superare i limiti fisici di tenuta in laterale del veicolo e garantendo un comfort interno veicolo maggiore. È stato inserito anche un semplicissimo controllo della dinamica in rollio del veicolo, cercando di limitare il LTR (‘Load Transfer Ratio’). Il modello, alternativo, presenta il seguente schema Simulink:

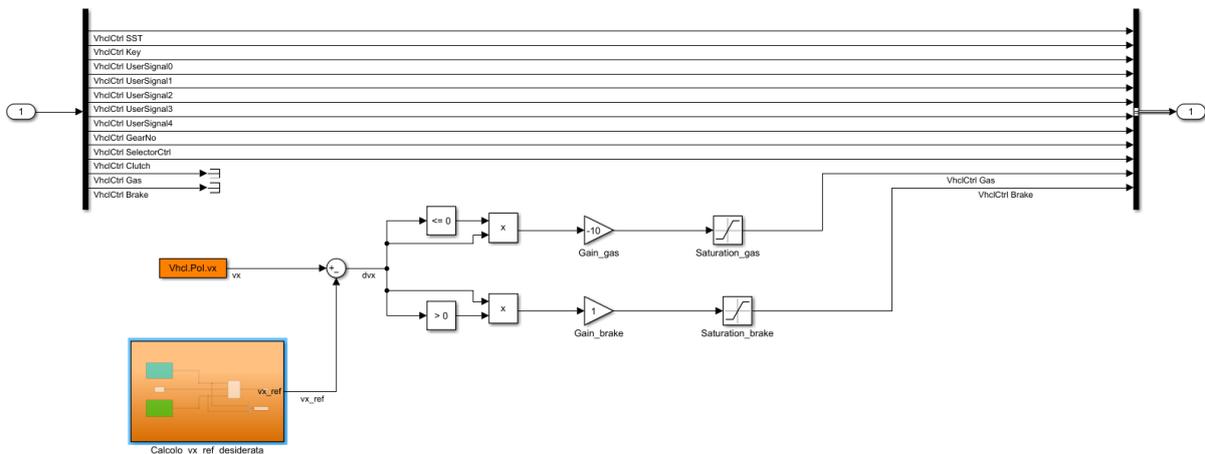


Figura 150 - Nuovo modello controllo off-line/on-line di velocità

Lo schema è analogo a quello presentato nel paragrafo precedente. Solo i guadagni del controllo proporzionale sono stati aumentati al fine di rendere più reattiva la risposta del

veicolo all'errore di velocità longitudinale. Inoltre, in questo modello è possibile selezionare sia il metodo off-line che quello on-line per la determinazione della velocità di riferimento $v_{x_{ref}}$. Si possono osservare i due sottosistemi nella figura seguente aprendo 'Controllo_vx_ref_desiderata':

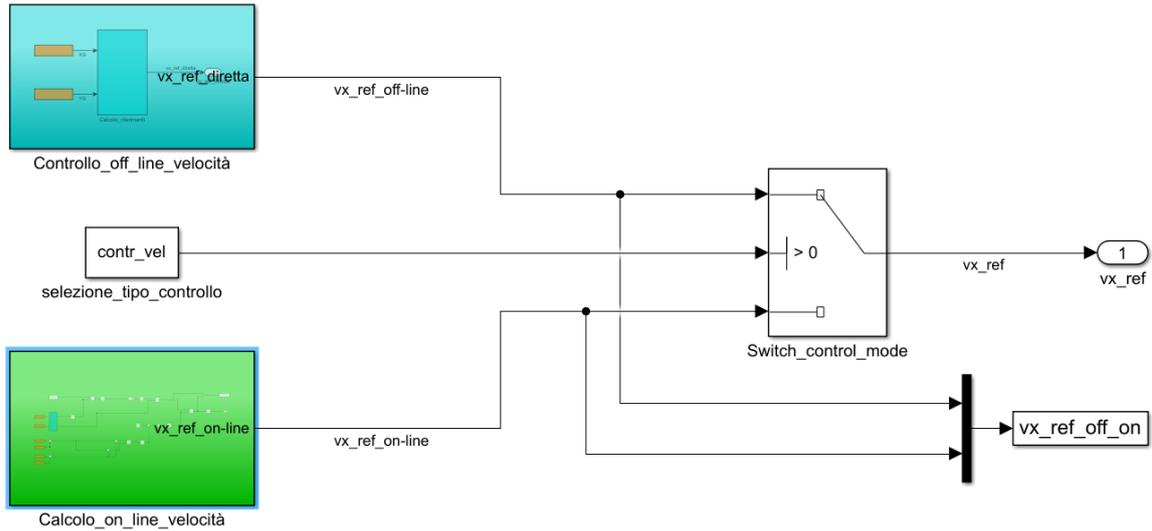


Figura 151 – Calcolo off-line e on-line velocità

Si osserva il 'Controllo_off_line_velocità' che corrisponde al modello descritto nel paragrafo 4.4.1: esso calcola una $v_{x_{ref,off_line}}$. Mentre nel sottosistema 'Calcolo_on_line_velocità' si può apprezzare lo schema per il calcolo on-line della velocità longitudinale di riferimento $v_{x_{ref,on_line}}$:

Tramite uno switch si può scegliere quale dei due controlli utilizzare. Se $contr_vel=1$ (imposto tramite lo script Matlab), quindi $contr_vel > 0$, verrà utilizzato il controllo off-line, mentre se $contr_vel=-1$, quindi $contr_vel < 0$, si sceglierà il metodo on-line.

$$v_{x_{ref}} = \begin{cases} v_{x_{ref,off_line}} & \text{if } contr_{vel} = 1 \\ v_{x_{ref,on_line}} & \text{if } contr_{vel} = -1 \end{cases} \quad (4.4)$$

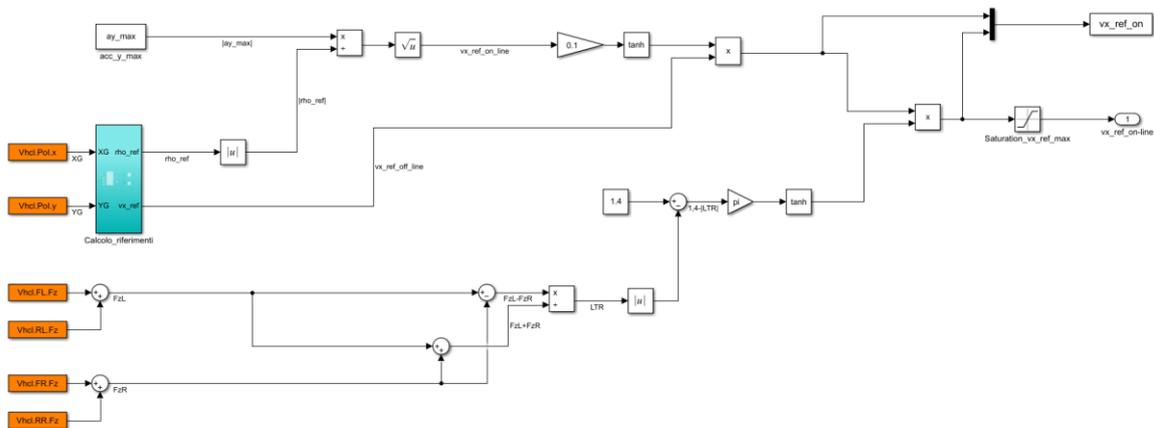


Figura 152 - Calcolo on-line velocità di riferimento

L'obiettivo di questo sistema è quello di calcolare la velocità longitudinale di riferimento, che il veicolo prova ad attuare grazie al controllo proporzionale di velocità (in figura 150), tramite la limitazione della velocità off-line di riferimento che è stata generata nella fase di path planning. Essa viene limitata moltiplicandola per un fattore compreso tra 0 e 1 che dipende dall'accelerazione laterale massima, sul piano, che non si vorrebbe superare, dalla curvatura di riferimento nello specifico punto della traiettoria e dal LTR ('Load Transfer Ratio') che si sviluppa.

$$LTR = \frac{F_{zL} - F_{zR}}{F_{zL} + F_{zR}} \quad (4.5)$$

In primis si calcola la velocità longitudinale massima di riferimento, in funzione dell'accelerazione laterale massima che non si vuole oltrepassare (sul piano) e della curvatura di riferimento del punto più vicino al veicolo:

$$vx_{ref,max} = \sqrt{\frac{a_{y,max}}{|\rho_{ref}|}} \quad (4.6)$$

Dove l'accelerazione massima laterale sul piano che si è imposta è pari a (ed è posta costante durante tutta la simulazione):

$$a_{y,max} = 4 \left[\frac{m}{s^2} \right]$$

Nota: Durante la simulazione questo limite verrà superato in corrispondenza di punti della manovra in cui l'accelerazione laterale che si sviluppa sul veicolo è dovuto per lo più a dinamiche di trasferimento di carico e di rollio causate da attraversamento parziale di dossi.

La ρ_{ref} è la curvatura della traiettoria di riferimento nel punto più vicino al veicolo. Il suo valore viene calcolato, ad ogni iterazione, grazie alla funzione 'Calcolo_riferimenti', ampiamente descritto nel capitolo 3.4.3.

È stato calcolato il valore assoluto della curvatura di riferimento perché essa assume valori positivi o negativi a seconda che il centro della curva punti verso destra o verso sinistra, mentre qui l'obiettivo è di ottenere il modulo della velocità longitudinale massima che si dovrebbe tenere per eseguire lo specifico tratto del percorso rispettando l'accelerazione laterale massima desiderata.

Il LTR è un valore, legato al trasferimento di carico, compreso tra 0 e 1, che assume valore pari a: 0 quando il veicolo distribuisce perfettamente il proprio peso tra il suo lato destro e sinistro, 1 quando il peso è scaricato completamente sul lato sinistro e -1 quando tutto il peso è sul lato destro. Le condizioni per cui $|LTR| = 1$ corrispondono a incipiente distacco del contatto gomma/terreno di un lato specifico del veicolo (da evitare).

Vengono, dunque, introdotti due coefficienti correttivi $k_{ay,max}$ e k_{LTR} . Il primo coefficiente correttivo, il cui valore ricade tra 0 e 1, tiene conto dell'accelerazione laterale massima che non si desidera superare, a causa della velocità longitudinale e della curvatura del percorso:

$$k_{ay,max} = \tanh(0,1vx_{ref,max}) \quad (4.7)$$

Il secondo coefficiente correttivo tiene conto dei trasferimenti di carico del veicolo. Come è noto, i trasferimenti di carico sono causati sia da un'esecuzione ad alta velocità di percorsi a elevata curvatura, sia ad attraversamento di dossi da parte del veicolo.

$$k_{LTR} = \tanh(\pi(1,4 - |LTR|)) \quad (4.8)$$

Quando il $|LTR|$ è pari a 1, la tangente iperbolica restituirà un valore pari a $\tanh(0,4\pi)$, mentre, quando il $|LTR| = 0$, si ottiene $k_{LTR} \approx 1$. Non si desidera mai avere un k_{LTR} uguale a zero, perché, in presenza di un salto trasversale, richiedere una velocità di riferimento che arrivi istantaneamente pari a 0; circostanza inattuabile e molto instabile.

In entrambi i coefficienti correttivi è stata inserita una tangente iperbolica per rendere più graduale il passaggio da una condizione a bassa curvatura e LTR, a condizioni di maggiore sollecitazione dinamica del veicolo.

Pertanto, la velocità di riferimento che viene restituita da questo sistema di generazione 'on-line' (perché limita, in tempo reale, la velocità di riferimento cosiddetta 'off-line') sarà:

$$vx_{ref,on_line} = vx_{ref,off_line}k_{ay,max}k_{LTR} \quad (4.9)$$

Infine, viene inserito un blocco 'Saturazione' impostando il suo limite superiore a 30 [m/s] (circa 100 [km/h]) per non eccedere con la velocità in qualsiasi manovra che si desideri eseguire. Ovviamente i limiti specifici imposti possono essere variati.

Nelle impostazioni di simulazione 'Maneuver', di CarMaker, si inserisce, non un tempo limite di percorrenza della traiettoria (precedentemente uguale a 10 secondi), bensì una distanza finale da percorrere (circa 101.6 metri), poiché, imporre un tempo specifico e una velocità variabile potrebbe portare (con alte probabilità) o al non raggiungimento del punto di destinazione, oppure un sorpasso di esso.

I risultati delle simulazioni che sfruttano sia questa logica di controllo della velocità, che la precedente, sono riportati nel capitolo seguente.

4.5 Risultati simulazioni su modello di veicolo CarMaker

Tramite lo script 'Run_Compl_IPG.m' vengono eseguite in automatico le seguenti operazioni che portano alle simulazioni finali su IPG-CarMaker:

- Lettura del file 'cmenv.m' con cui si mette in comunicazione Matlab e CarMaker
- Caricamento delle variabili di riferimento relative alla traiettoria generata dal path planner: $[X_{ref}, Y_{ref}, \psi_{ref}, \rho_{ref}, s_{ref}, vx_{ref}]$
- Caricamento della matrice dei guadagni ottimi K
- Lancio del modello Simulink 'generic.mdl' alla base della simulazione CarMaker
- Fase di Postprocess in cui si riportano le risposte del sistema nella persecuzione della manovra

Si riporta, dunque, la risposta finale del veicolo a molti gradi di libertà di IPG-CarMaker, in cui sono stati implementati il controllo di sterzo e i due controlli di velocità (cap. 4.4):

4.5.1 Risultati controllo di velocità con riferimento off-line

- 1) La traiettoria realmente eseguita (in rosso) messa a confronto con quella di riferimento:

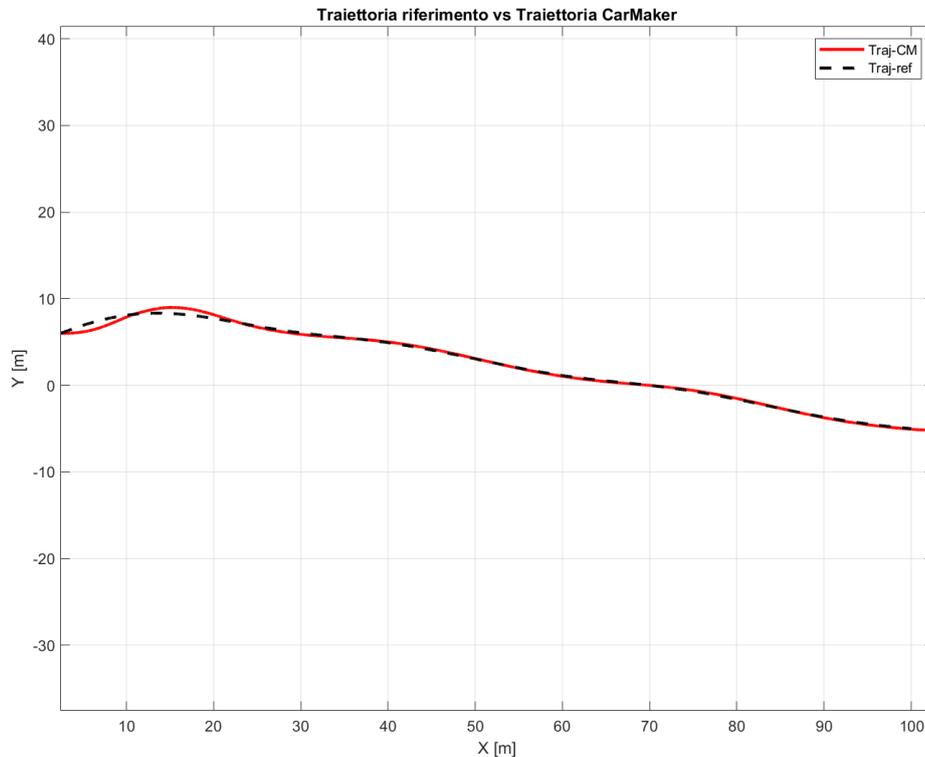


Figura 153 - Risultato persecuzione traiettoria di riferimento su CarMaker

La fedeltà con cui il controllo di sterzo è in grado di inseguire il tracciato di riferimento è molto alta, a parte un piccolo scostamento iniziale dovuto alla presenza di un dosso parzialmente attraversato. A prova della validità dell'algoritmo sviluppato:

- 2) Gli errori di posizione e imbardata commessi durante questa manovra sono:

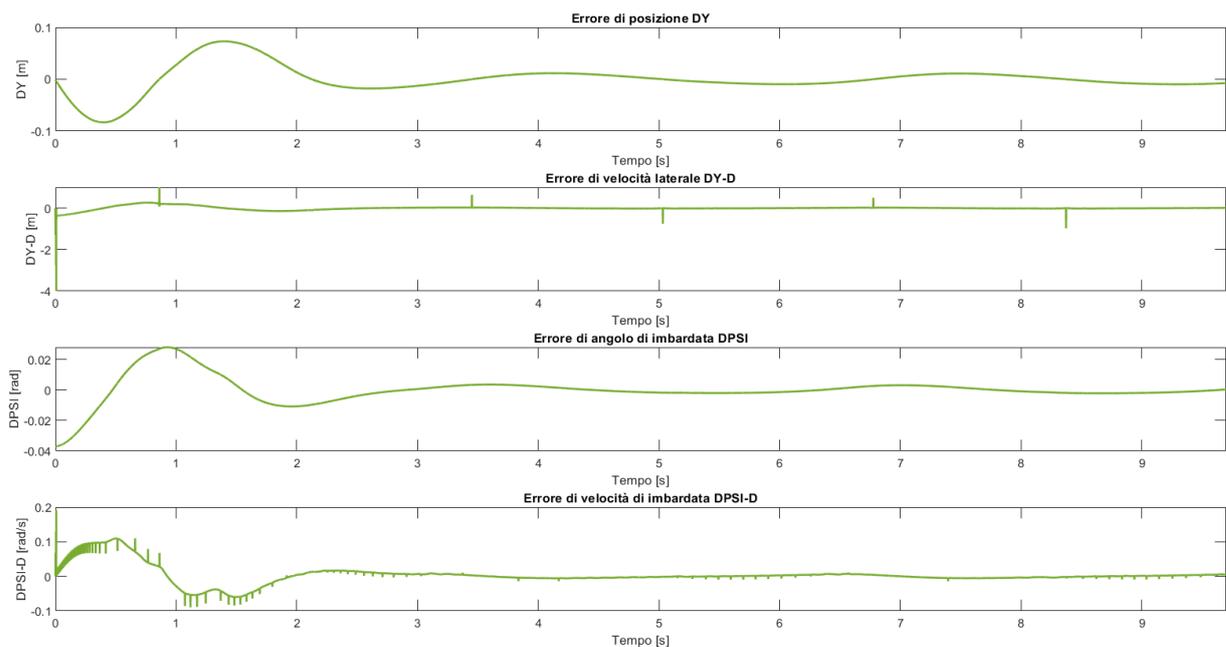


Figura 154 - Errori di posizione laterale e di imbardata manovra CarMaker

Si noti come l'errore di posizione laterale non superi mai gli 0,1 [m], e l'errore di angolo di imbardata gli 0,04 [rad] in modulo. Si può concludere che il controllo di sterzo implementato è di assoluta validità. Si possono osservare delle oscillazioni sull'errore di velocità di imbardata, dovute a oscillazioni numeriche. Queste ultime si potrebbero tranquillamente eliminare inserendo una funzione di trasferimento (filtro) con opportuna frequenza di taglio. Questo non è stato fatto qui direttamente per evitare che il segnale avesse un ritardo, che si sarebbe propagato sull'angolo volante conseguentemente calcolato.

3) L'angolo volante con le rispettive derivate:

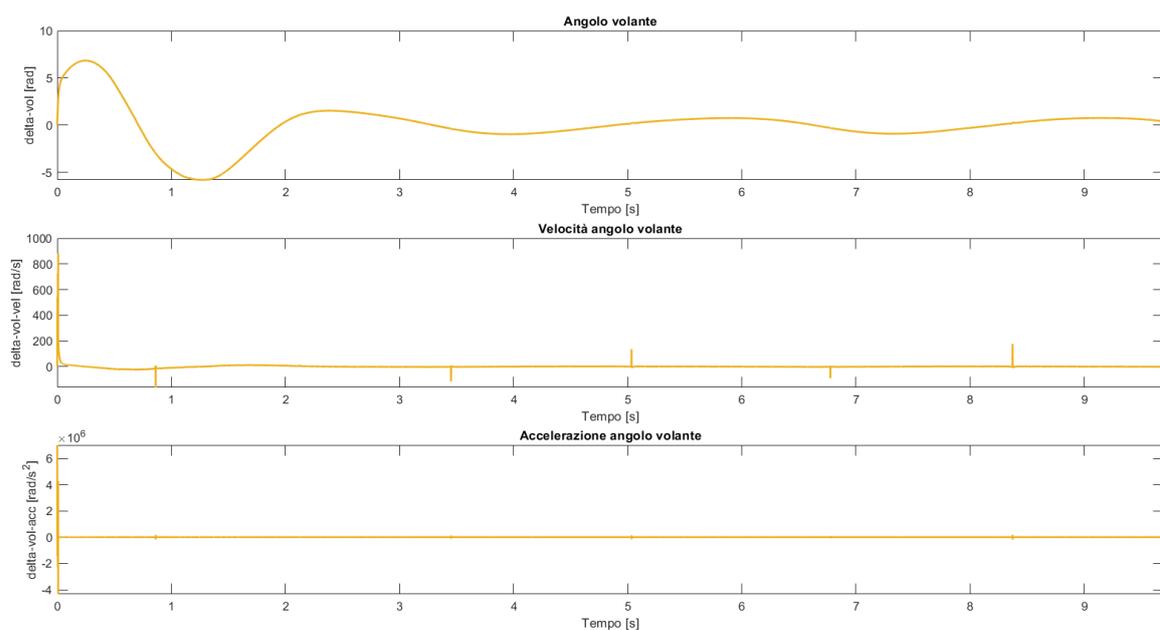


Figura 155 - Angolo volante manovra CarMaker

Anche in questo caso le derivate dell'angolo volante presentano, a causa di approssimazioni numeriche generate durante la derivazione, delle oscillazioni repentine (poco visibili perché l'asse delle ordinate presenta valori molto grandi a causa di picchi elevati in corrispondenza dell'inizio della simulazione). Questo non rappresenta un problema ai fini della manovra, dal momento in cui quello che conta è l'angolo volante stesso e non le sue derivate.

- 4) L'acceleratore/Gas, il freno/Brake e la velocità longitudinale effettuate durante il percorso:

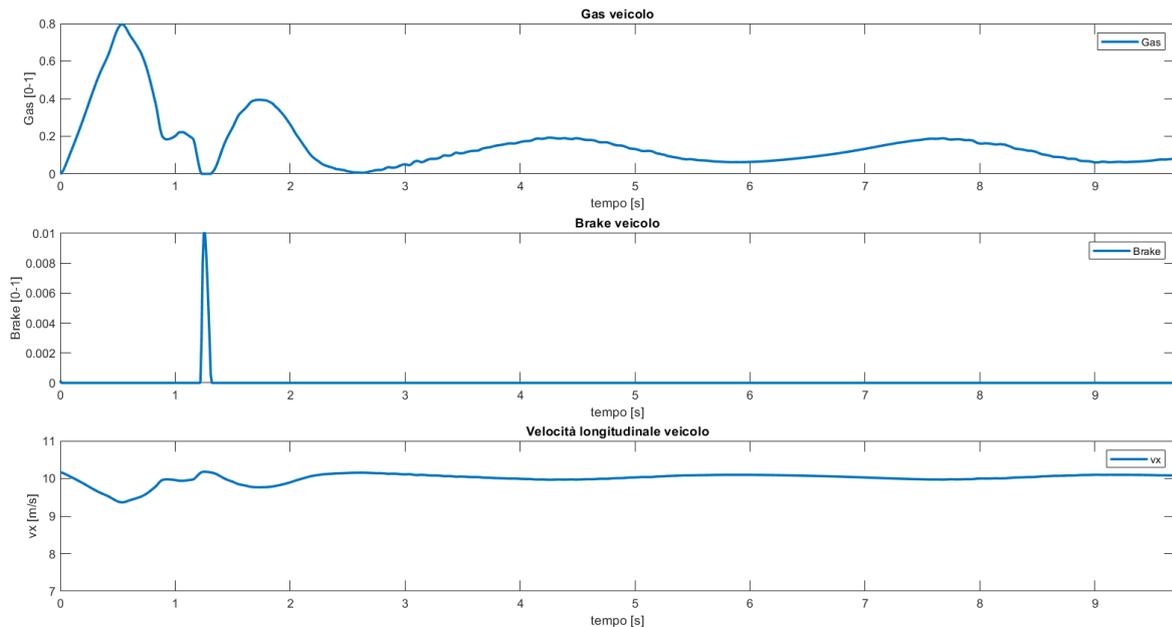


Figura 156 - Controllo velocità longitudinale CarMaker off-line

Si nota un utilizzo del Gas e del Freno da parte del Driver maggiore rispetto a quello che faceva il Driver di default di CarMaker. Il risultato è un mantenimento sufficientemente preciso della velocità di riferimento. Osservando lo 'Scope', nel modello Simulink, dell'errore di velocità longitudinale rispetto a quella di riferimento, si potrebbe osservare che il massimo scostamento è di circa 1 [m/s] (in modulo), valore abbastanza contenuto considerando che il percorso eseguito prevede l'attraversamento di dossi e curve che tendono naturalmente a rallentare il veicolo. Un controllo più aggressivo potrebbe essere implementato aumentando i guadagni descritti al capitolo 4.4, ma questo porterebbe anche a più ampie oscillazioni dei segnali, dunque, più 'sporcizia numerica'.

La traiettoria ottenuta su CarMaker non persegue così fedelmente il path di riferimento come il modello a quattro stati (anche se molto preciso in termini assoluti). Questo è naturale, dal momento in cui un sistema a minor gradi di libertà è più rigido nell'esecuzione di una manovra, mentre un modello a molti gradi di libertà presenta maggior flessibilità di azione. Un'altra spiegazione è che nel modello a quattro stati i guadagni in K sono dieci volte più grandi rispetto a quelli definitivamente utilizzati su CarMaker. Questo porta a più reattività nel modello lineare (a quattro stati), ma causa instabilità in un modello fortemente caratterizzato dalle non linearità.

Di seguito si riporta un'analisi della risposta dinamica del veicolo, nell'esecuzione della manovra, tramite alcune variabili:

- Accelerazione longitudinale a_x
- Accelerazione laterale a_y
- Angolo di assetto β e sua derivata $\dot{\beta}$
- Angolo di beccheggio θ e sua derivata $\dot{\theta}$
- Angolo di rollio φ e sua derivata prima $\dot{\varphi}$ e seconda $\ddot{\varphi}$
- Forze verticali sugli pneumatici $F_{zfl}, F_{zfr}, F_{zrl}, F_{zrr}$ e corrispondente LTR (Load Transfer Ratio)

Il LTR è definito coerentemente a quanto descritto in [30]:

$$LTR = \frac{F_{zL} - F_{zR}}{F_{zL} + F_{zR}}$$

$$F_{zL} = F_{zfl} + F_{zrl} \text{ e } F_{zR} = F_{zfr} + F_{zrr}$$

Grafici Accelerazione longitudinale, Accelerazione laterale, Angolo di assetto e sua derivata: Si osservi che l'accelerazione laterale raggiunge i 10 [m/s²] nella prima fase di attraversamento parziale di un avvallamento.

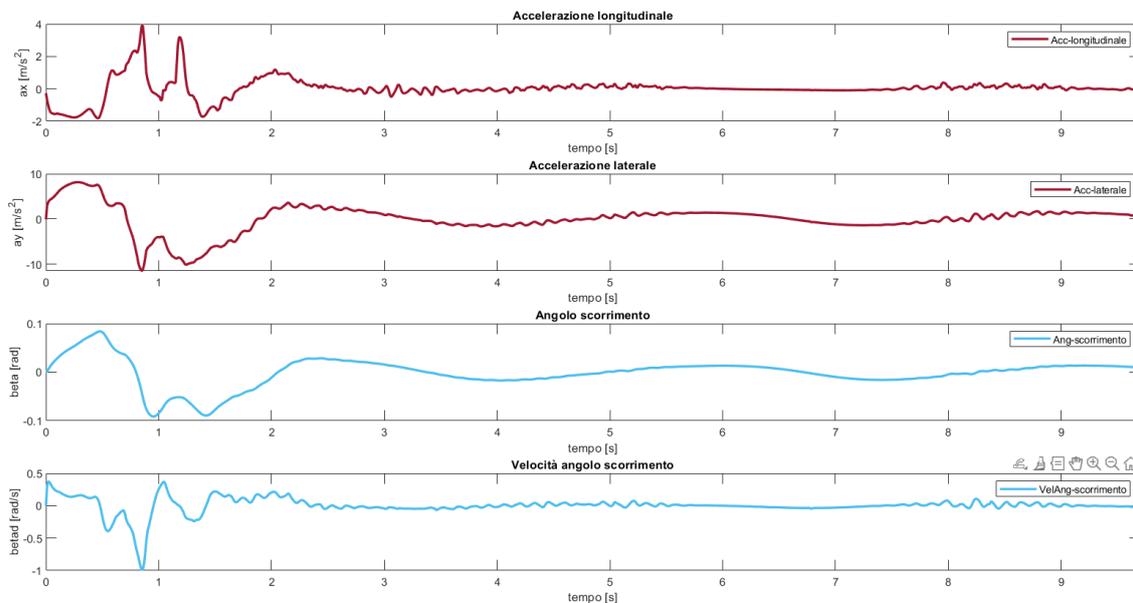


Figura 157 - Risposte dinamiche controllo velocità off-line 1

Grafici Angolo di beccheggio e sua derivata, Angolo di rollio e sue derivate prima e seconda:

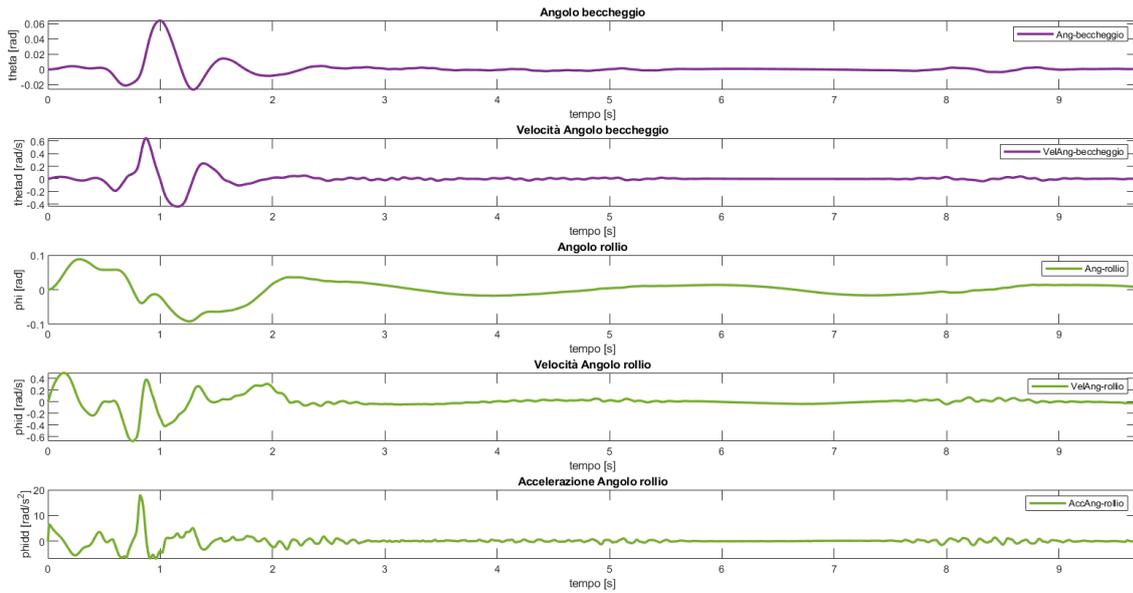


Figura 158 - Risposte dinamiche controllo velocità off-line 2

Grafici Forze verticali e LTR:

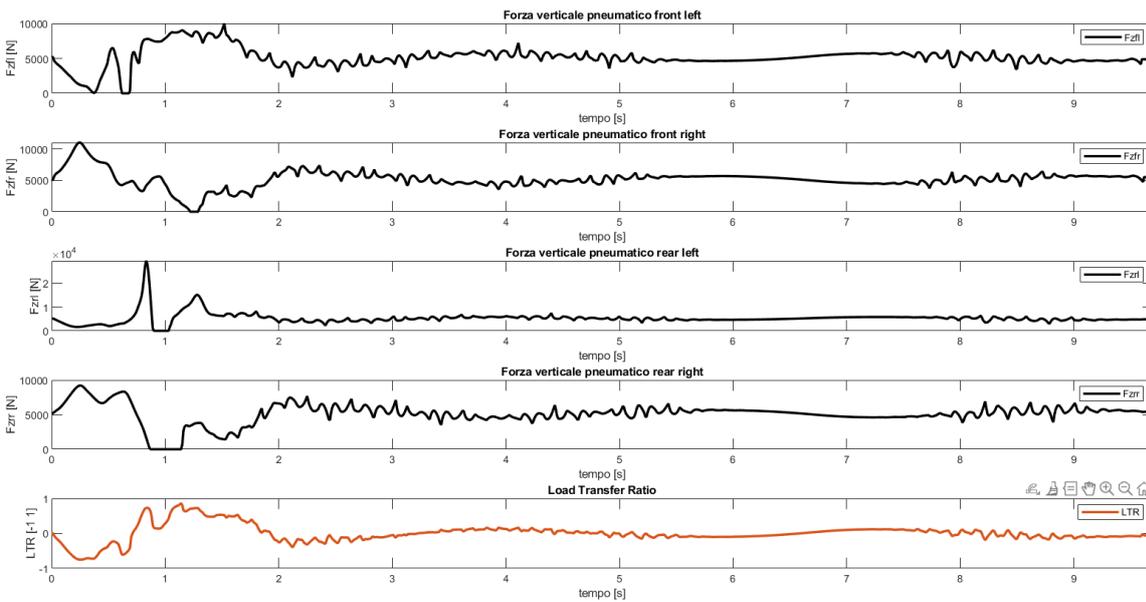


Figura 159 - Risposte dinamiche controllo velocità off-line 3

Come si può notare, principalmente dal grafico relativo all'accelerazione laterale che si sviluppa sul veicolo e dal LTR, i trasferimenti di carico sono molto importanti, e il limite indicativo di 10 m/s^2 di aderenza al suolo di un veicolo comune (come quello che è stato utilizzato) viene superato durante le prime manovre, che, come si può vedere dall'animazione dell'IPGMovie di CarMaker (in allegato), sono molto repentine. Pertanto, è stato implementato il controllo 'on-line' di velocità (capitolo 4.4.2) che limita l'accelerazione laterale, adeguando la velocità del veicolo in base alla curvatura che esso sta eseguendo e in base ai trasferimenti di carico che si presentano sul veicolo.

4.5.2 Risultati controllo di velocità con riferimento on-line

Vengono analizzati gli stessi risultati che sono stati riportati con il generatore 'off-line' della velocità di riferimento. Anche in questo caso, al fine di controllare la velocità del mezzo, si è agito sull'apertura delle valvole di alimentazione motore e sulla percentuale di utilizzo dell'impianto frenante.

Nella figura che segue si osserva come il tracciato di riferimento si stato eseguito più fedelmente con il controllo 'on-line' di velocità rispetto al veicolo soggetto a un controllo 'off-line' di velocità. Questo è dovuto al fatto che il metodo on-line abbassa la velocità del veicolo in corrispondenza di zone critiche in cui è necessario avvicinarsi con un'andatura inferiore tratti ad elevata curvatura o con ostacoli da superare (con conseguenti importanti trasferimenti di carico).

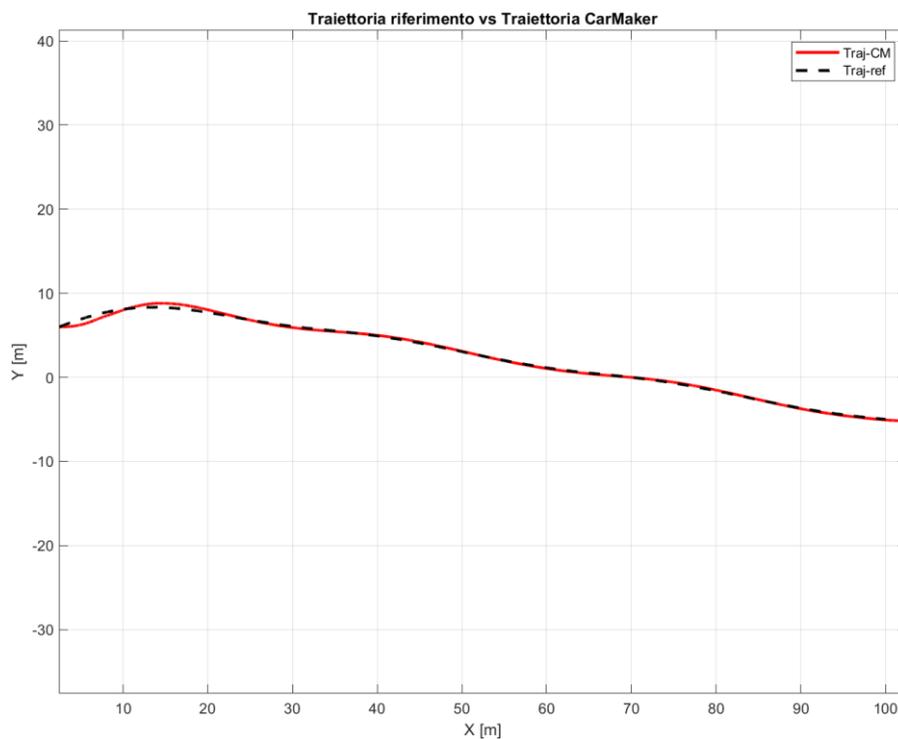


Figura 160 - Risultato persecuzione traiettoria di riferimento su CarMaker

Gli errori di posizione trasversale, di angolo di imbardata e relative derivate presentano una serie di differenze sostanziali rispetto al caso con controllo off-line:

- Gli estremi massimi, in termini assoluti, dell'errore di posizione sono stati ridotti
- L'errore di angolo di imbardata è molto simile
- Le derivate degli errori presentano dei picchi inferiori ma molte oscillazioni. Il motivo di ciò risiede nel fatto che il primo picco, che si presentava in precedenza, qui è scomparso, per lasciar spazio a un'apparente maggior quantità di oscillazioni numeriche. Gli estremi massimi sono, tuttavia, stati ridotti

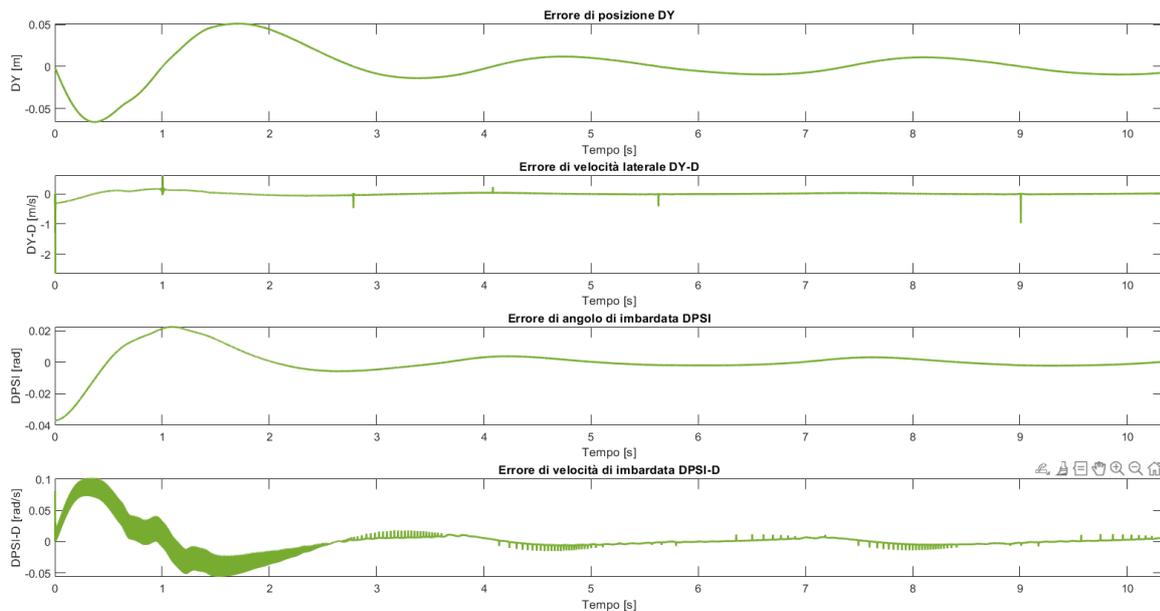


Figura 161 - Errori di posizione laterale e di imbardata manovra CarMaker

L'angolo volante presenta un picco superiore rispetto al caso precedente, a favore, però, di una migliore persecuzione della traiettoria. Non sono scomparse le oscillazioni numeriche sulle derivate.

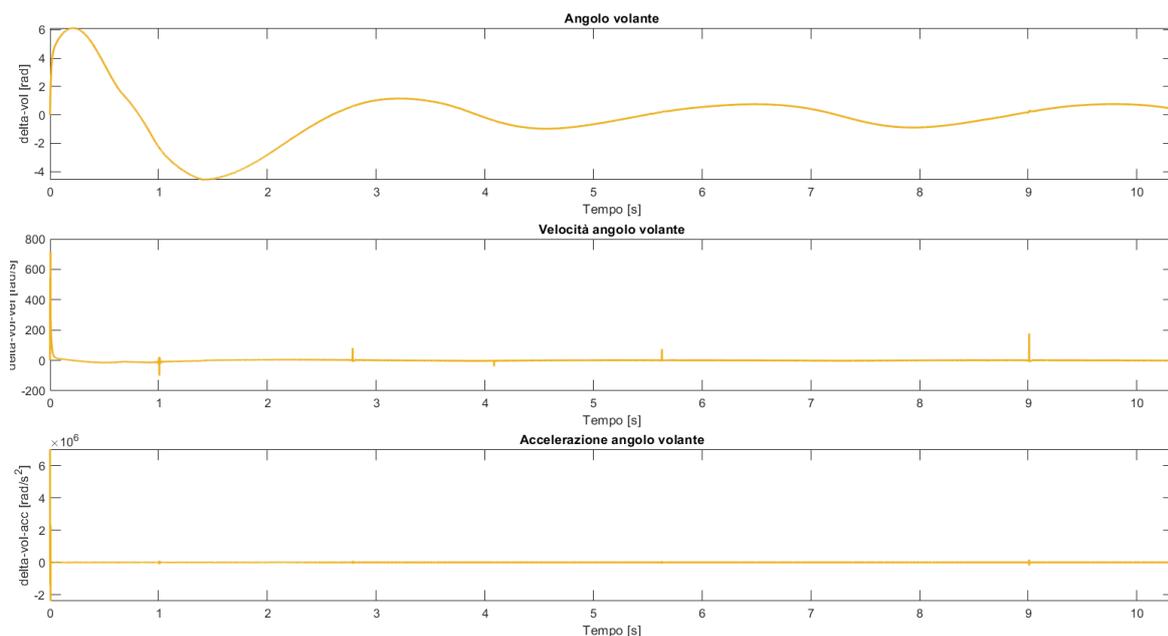


Figura 162 - Angolo volante manovra CarMaker

Una grossa differenza si può, invece, osservare nello sfruttamento del pedale del Gas e del Freno. Sono presenti forti oscillazioni dovute principalmente al fatto che qui la velocità di riferimento da inseguire non è più una costante, ma altamente variabile in base alla curvatura che il mezzo deve affrontare istante per istante. $a_{y,max} \& \rho_{ref} \rightarrow vx_{ref,max} \rightarrow vx_{ref}$.

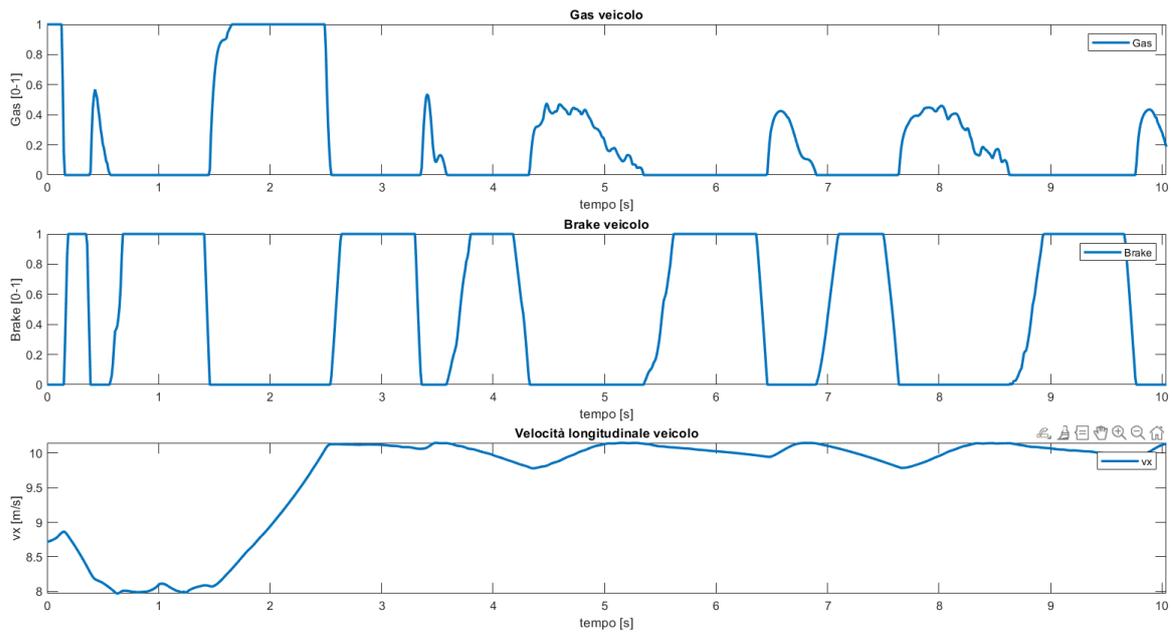


Figura 163 - Controllo velocità longitudinale CarMaker on-line

Come ci si aspettava la velocità longitudinale del veicolo aumenta e diminuisce per far fronte a elevate curvature e trasferimenti di carico. Una forte diminuzione della velocità si presenta, infatti, in prossimità dell'attraversamento del primo dosso (tra gli 0,6 e i 2 secondi circa)

Nei grafici che seguono si osserva che l'accelerazione massima laterale, in modulo, è scesa di circa $2/3$ [m/s^2]: sono presenti dei picchi elevati iniziali di circa 8 [m/s^2] dovuti al superamento parziale di un ostacolo che genera forti trasferimenti di carico, quindi elevata accelerazione laterale. Questi picchi, purtroppo non si possono limitare con un controllo di velocità di questo tipo, che si accorge di dover rallentare solamente quando si trova già in un punto ad elevata curvatura o, peggio ancora, ad elevato LTR; servirebbe, bensì, un controllo di velocità con 'Preview', ovvero in grado di prevedere quale sarà la curvatura e il LTR prima che ci si trovi in un dato punto critico, e adattare, di conseguenza, la velocità per avvicinare più lentamente possibile lo specifico ostacolo o curva.

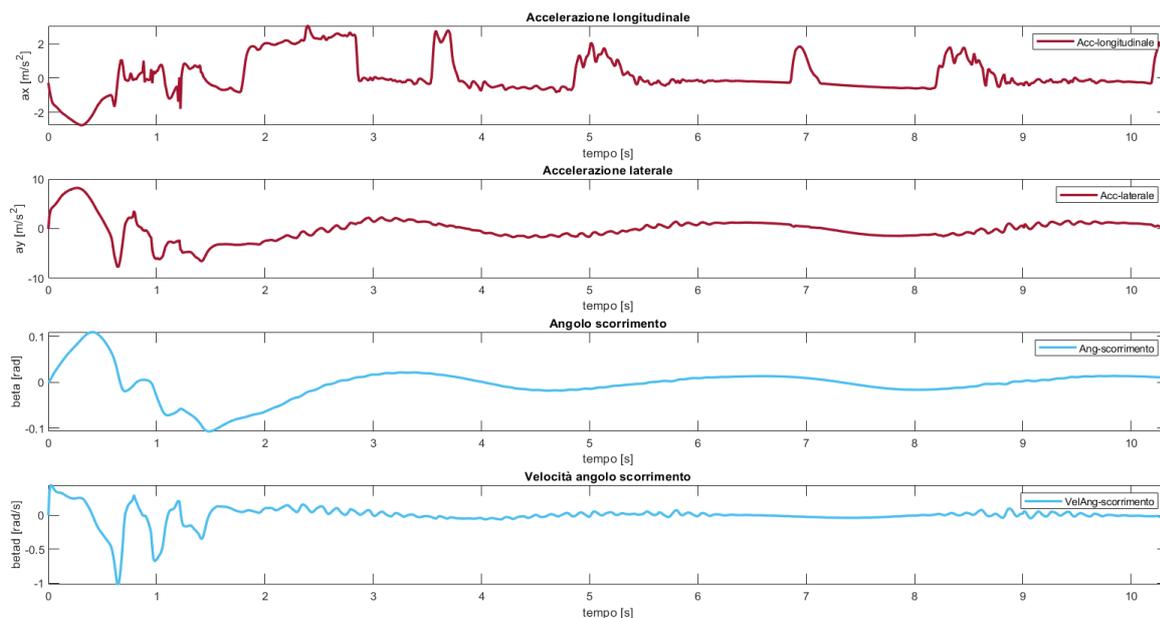


Figura 164 - Risposte dinamiche controllo velocità on-line 1

Volendo analizzare più da vicino l'accelerazione centripeta e quella tangenziale sviluppate sul piano di avanzamento del veicolo, si osserva con più chiarezza di come a_y massima sviluppata sia di circa $7/8 \text{ [m/s}^2]$, superiore ai 4 desiderati, ma limitabili solo con preview o con velocità molto più basse di attraversamento.

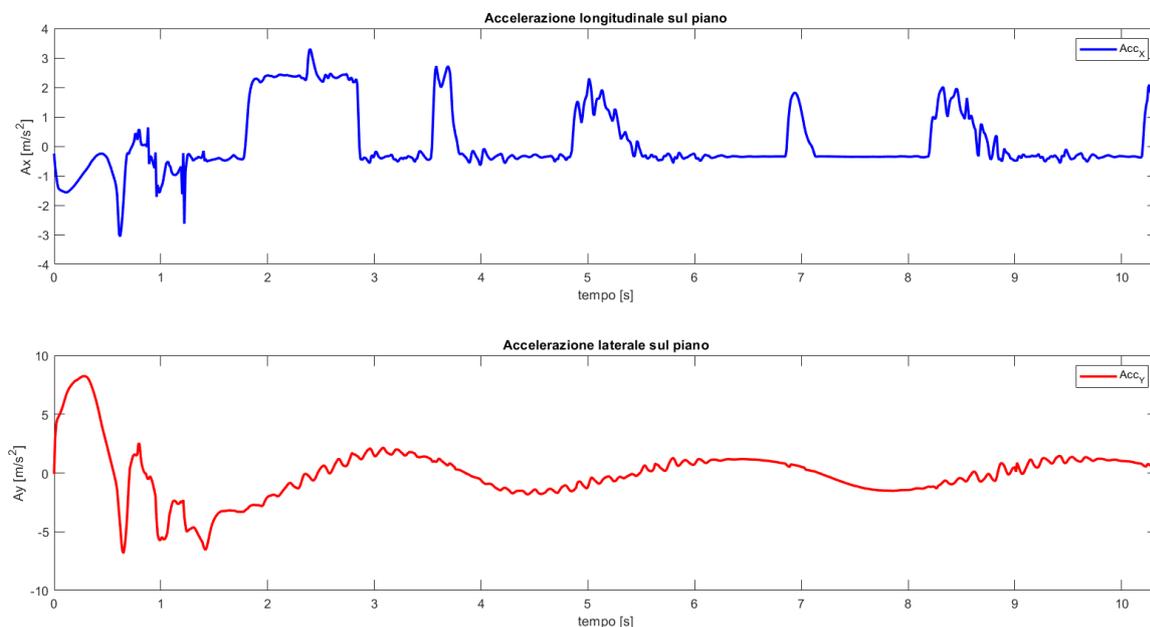


Figura 165 - Accelerazioni tangenziali e centripete con controllo on-line

Si riportano, per completezza, i grafici degli angoli di beccheggio e di rollio del veicolo. Gli angoli, velocità e accelerazioni di rollio (in verde) sono state abbassate, proprio grazie al sistema di generazione on-line della velocità di riferimento.

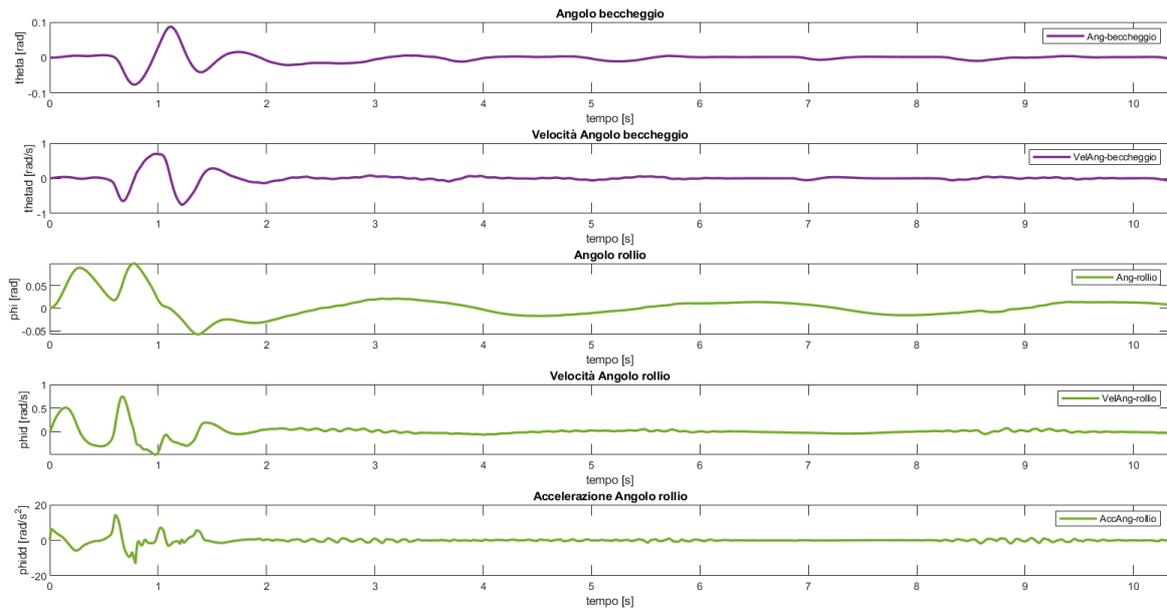


Figura 166 - Risposte dinamiche controllo velocità on-line 2

Osservando il $|LTR|$ è evidente come esso non assuma mai un valore pari a 1. Risultato sperato grazie all'introduzione del fattore correttivo k_{LTR} presentato al capitolo 4.4.2.

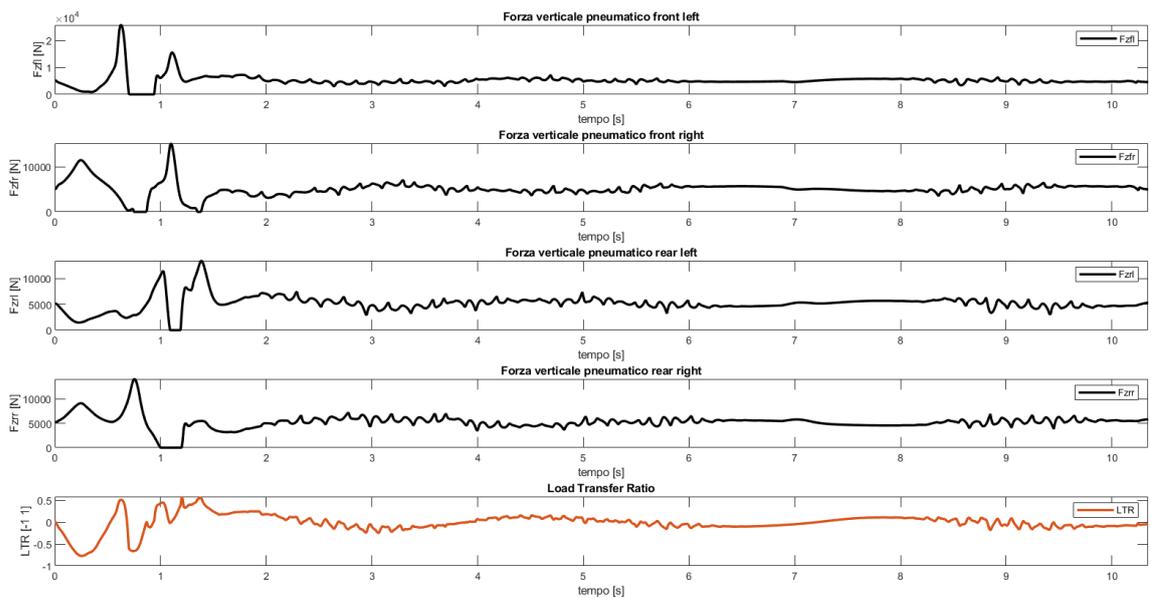


Figura 167 - Risposte dinamiche controllo velocità on-line 3

4.5.3 Confronto velocità di riferimento off-line vs on-line

In quest'ultima sezione vengono riportate una serie di analisi sulla velocità di riferimento generata dal controllo off e on line:

- Confronto velocità longitudinali di riferimento restituite dai due modelli off-line (capitolo 4.4.1) e da quello on-line (capitolo 4.4.2)
- Confronto tra velocità di riferimento on-line generate in tre passaggi legati alle costanti $k_{ay,max}$, k_{LTR} e blocco saturazione

- Confronto velocità di riferimento on-line e velocità realmente eseguita dal veicolo di simulazione

Velocità longitudinale di riferimento off-line (in blu) vs on-line (in rosso). In questo caso la velocità off-line è stata impostata costante, facilmente inseguibile dal controllo proporzionale. La velocità di riferimento on-line, di più difficile inseguimento, presenta una serie di oscillazioni dovute ai fattori correttivi legati alla curvatura e al LTR:

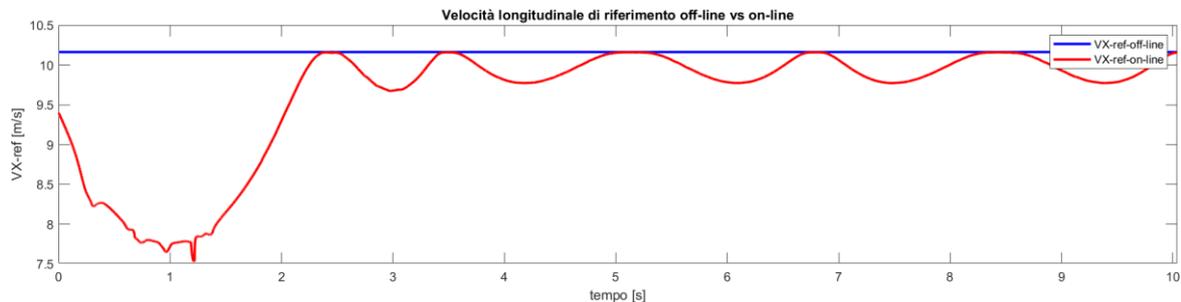


Figura 168 - Confronto velocità longitudinale di riferimento off-line vs on-line

È Presente una notevole riduzione della velocità di riferimento longitudinale tra off-line (priva di controllo di feedback) e on-line (con controllo feedback).

Confronto tra le velocità longitudinale di riferimento on-line tenendo conto:

- Della sola dinamica in laterale, quindi della curvatura, sul piano (in giallo)
- Integrando la componente legata al trasferimento di carico, quindi al contributo del LTR (in verde)
- Saturando la velocità longitudinale di riferimento on-line a un massimo di 30 [m/s].

Si vede una graduale attenuazione della velocità longitudinale di riferimento on-line. L'effetto della saturazione non è evidente (poiché non viene mai raggiunta la velocità di 30 [m/s]); si vedrebbe qualora si decidesse di eseguire il percorso a una velocità più sostenuta. La curva verde, quindi, coincide con quella rossa. Si noti, però, la forte attenuazione legata al k_{LTR} in corrispondenza dell'attraversamento del primo ostacolo (tra gli 0,2 e i 2 secondi circa)

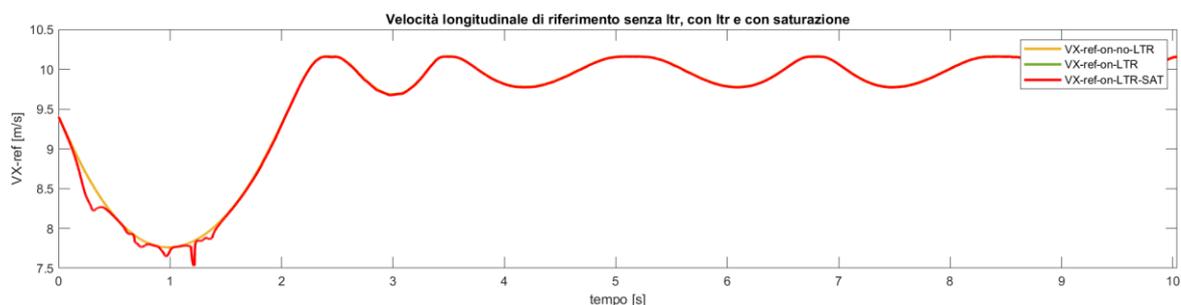


Figura 169 - Confronto velocità di riferimento on-line con coefficienti aggiunti

Infine: confronto velocità longitudinale di riferimento on-line vs velocità longitudinale realmente eseguita dal veicolo. Si vede una sostanziale discrepanza tra le due velocità: di target (in rosso) e la velocità che il veicolo riesce ad attuare (in blu). Questo risultato è atteso dal momento in cui la velocità longitudinale di riferimento è molto oscillante e non

attuabile da parte di alcun freno o acceleratore reale. Come già anticipato, solo un controllo predittivo potrebbe limitare questi effetti dinamici indesiderati, oltre all'inserimento di funzioni di calcolo della velocità di riferimento molto più gradualmente.

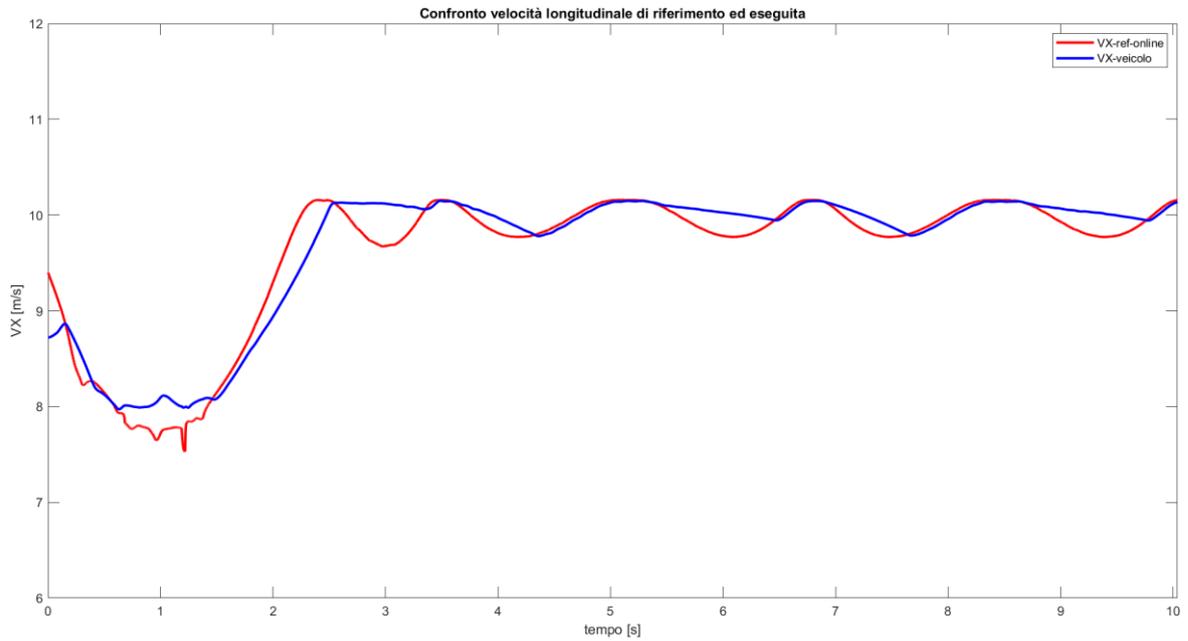


Figura 170 - Confronto velocità di riferimento on-line e velocità eseguita dal veicolo di simulazione

Si noti, tuttavia, le corrette 'accelerate' e 'frenate' in base al Δvx presente.

5. Conclusioni

Il presente lavoro di tesi si era posto l'obiettivo ambizioso di sviluppare logiche di pianificazione e controllo di traiettorie per veicoli passeggeri, eventualmente utilizzabili in ambito off-road. I modelli presenti in letteratura sono molto numerosi, ma relativi ad ambienti strutturati quali autostrade, città e parcheggi. Essi vanno dunque studiati a fondo e opportunamente adattati alle necessità specifiche di uno scenario complesso com'è la natura.

Nonostante la meta da raggiungere fosse lontana e alquanto complessa per uno studente, come il sottoscritto, alcuni passi verso una soluzione preliminare sono stati eseguiti con successo. Si è passato in rassegna il sistema alla base di un qualsiasi veicolo autonomo (percezione, elaborazione e controllo), per poi approfondire il livello tattico di pianificazione della traiettoria, che rimane, secondo parere personale, quello più sconosciuto tuttora, ma sicuramente il più necessario per rendere efficace ed efficiente un veicolo autonomo. Tramite la sua ottimizzazione passa la reale capacità del VA di navigare all'interno di una qualsiasi situazione, più o meno difficile da affrontare. Ci si è spostati dallo studio di alcuni modelli di Artificial Potential Field, per giungere a una soluzione Curves Motion Planning in una Occupancy Grid Map sviluppata e qui ottimizzata in ambiente Matlab. La traiettoria di riferimento generata, cinematicamente fattibile e priva di collisioni, è stata perseguita grazie a un algoritmo di path tracking sviluppato sulla base di un modello a quattro stati errore (su single-track model) e grazie a una logica LQR che ha permesso di trovare la matrice di guadagni ottimi con cui calcolare un'azione di feedback di angolo volante. Il problema fondamentale legato alla modellazione e interpretazione degli ostacoli naturali (buche e avvallamenti) è stato scavallato, semplificando e approssimando tali entità con delle zone off-limits rettangolari, sulla mappa 2D, in cui il veicolo doveva muoversi, senza transitare interamente sopra di essi.

Si è passati, infine, alla validazione sperimentale dell'intero algoritmo, per mezzo del software IPG-CarMaker che mette a disposizione veicoli virtuali di simulazione a molti gradi di libertà (pressoché reali), prendendo il comando dell'angolo di sterzo, del gas e dell'acceleratore, tramite opportuni controlli sviluppati in parte nel modello a quattro stati o direttamente nella logica Simulink che governa i mezzi CM. I risultati dinamici ottenuti, nell'esecuzione delle manovre, hanno dato conferma dell'assoluta validità dei modelli sviluppati, lasciando spazio aperto ad eventuali ottimizzazioni basate su logiche predittive (quali MPC o semplici preview aggiuntive).

In conclusione, ciò che è stato fatto deve essere il punto di lancio verso un mondo di modelli intelligenti, che vanno studiati ed approfonditi, per poter dare, in futuro, un reale contributo innovativo al mondo STEM (Science, Technology, Engineering, Mathematics).

Ringraziamenti

Sto per giungere al termine la prima fase fondamentale di crescita che ho avuto la fortuna di vivere con serenità. Tutto ebbe inizio ventiquattro anni fa; da allora un susseguirsi di eventi casuali o predeterminati mi ha portato ad essere la persona che sono oggi. Non a tutti capita di trovarsi nel posto giusto del mondo al momento giusto. Non so se io faccia parte di quelle persone, ma così mi sento, e questo è quello che conta, quello che, forse, fa la differenza: avere la giusta prospettiva! Sia nei momenti belli, ma anche in quelli più difficili. Ora sembrerà tutto fantastico, anche se sappiamo bene che così non sempre è, ma credo che ognuno di noi abbia a disposizione delle carte da sfruttare, e un consiglio che mi sento di dare è di giocarle con semplicità e spontaneità d'animo. Lasciare spazio alle idee e alle proprie passioni è il più grande regalo che possiamo farci. Tutte bello; ma, mettiamo caso, di trovarci per qualche giorno da soli su questo pianeta o semplicemente abbandonati da tutti: cosa saremmo in grado di fare? Probabilmente ognuno di noi darebbe una risposta diversa, o forse la stessa, chi lo sa? Allora mettiamo caso, per la seconda volta, che la domanda fosse posta a me. Beh, non avrei dubbi su cosa dire:

Se non fosse per i miei genitori, non sarei qua a scrivere e a poter dire con assoluta lucidità e libertà che sono contento di tutto quello che ho vissuto finora. Grazie a loro ho avuto la possibilità di sviluppare un pensiero autonomo, personale, svincolato da ogni forma di imposizione autoritaria. Direi che è un bel regalo.

Se non fosse per mio fratello e per i miei familiari, sia quelli che ci sono che quelli che si trovano in un altro luogo, probabilmente non avrei avuto i giusti esempi con cui sviluppare la mia naturale identità.

Se non fosse per i miei amici del 'Poli': Cecilia, Francesco e Giorgio, tra lezioni ed esami, non avrei avuto la possibilità di affrontare con lo stesso spirito di squadra le svariate prove che ci venivano poste, e che siamo riusciti a superare egregiamente.

Se non fosse per i miei ex compagni, e attuali amici: Davide e Nuccio, non avrei avuto qualcuno da cui prendere ispirazione. Siete davvero dei fenomeni!

Se non fosse per la mia compagna di camminate preesame: Lu (il nostro cane), non avrei trovato la calma che solo un animale è in grado di trasmettere (nonostante sia molto agitata).

Se non fosse per i miei due amici di sempre non sarei la persona che sono oggi. Come vi ho già detto, moltissimo di quello che sono lo devo a voi!

Se non fosse per i miei due 'nuovi migliori amici', probabilmente, non mi sarei reso conto del fatto che è sempre possibile trovare qualcuno con cui coltivare un nuovo rapporto di questo tipo.

Infine, senza di te, che hai deciso di condividere con me i momenti migliori e quelli peggiori della vita, che sei la persona che ripone più fiducia in me, non sarei ogni giorno disposto a fare sempre meglio, a spingermi un passo sempre oltre quello che prima neanche potevo immaginare.

Grazie, ancora, di cuore a tutti voi!

Ringraziamento ai professori

Un ringraziamento particolare va ai professori: Mauro Velardocchia, Antonio Tota e Luca Dimauro che mi hanno accompagnato, credendo sempre in me e spronandomi con la leggerezza che li contraddistingue, senza mai farmi sentire il peso del lavoro.

A presto!

Bibliografia

- [1 A. Tota, *Integration of Active Systems for a Global Chassis Control Design*, Torino: Politecnico di Torino, 2016.
- [2 S. Aldo, *Lezione: Introduction to automated driving*, Torino, 2023.
- [3 T. Ardi, M. Tabet e e. al, «A Survey of End-to-End Driving: Architectures and Training Methods,» *IEEE Transactions on Neural Networks and Learning Systems*, 2022.
- [4 SAE, «Normativa SAE J3061_202104,» 2021. [Online]. Available: https://www.sae.org/standards/content/j3016_202104/.
- [5 L. Cereda, «QUATTORRUOTE,» 06 05 2021. [Online]. Available: https://www.quattoruote.it/news/tecnologia/2021/05/06/guida_autonoma_la_sae_ridefinisce_ilivelli_di_automazione_per_fare_chiarezza.html.
- [6 «Plan 3D Paths for Drones | Motion Planning Hands-on Using RRT Algorithm, Part 4,» [Online]. Available: <https://it.mathworks.com/videos/motion-planning-with-the-rrt-algorithm-part-4-plan-3d-paths-for-drones-1657611815362.html>.
- [7 S. T. M. M. J. D. Dmitri Dolgov, «Path Planning for Autonomous Vehicles in Unknown Semi-structured Environments,» *The International Journal of Robotics Research* , 2010.
- [8 J. T. S. L. Jiayi Sun, «Collision Avoidance for Cooperative UAVs With Optimized Artificial Potential Field Algorithm,» *IEEE Access*, 2017.
- [9 R. N. B. Y. X. Z. R. B. S. Z. Chen Hua, «A Global Path Planning Method for Unmanned Ground Vehicles in Off-Road Environments Based on Mobility Prediction,» *MDPI machines*, 2022.
- [1 «Introduction to Motion Planning Algorithms | Motion Planning Hands-on Using RRT
0] Algorithm, Part 1,» [Online]. Available: <https://it.mathworks.com/videos/motion-planning-with-the-rrt-algorithm-part-1-introduction-to-motion-planning-algorithms-1620626888580.html>.
- [1 J. P. V. M. F. N. David Gonzalez Bautista, «A Review of Motion Planning Techniques
1] for Automated Vehicles,» *IEEE Transactions on Intelligent Transportation Systems*, 2015.
- [1 J. S. K. S. S. L. K. H. P. Joo Young Hwang, «A Fast Path Planning by Path Graph
2] Optimization,» *IEEE Transactions on systems, man, and cybernetics*, 2003.
- [1 D. D. S. T. a. o. Micheal Montemerlo, «Junior: The Stanford Entry in the Urban
3] Challenge,» 2007.

- [1 G. L. N. L. Yunlong Bai, «Motion Planning and Tracking Control of Autonomous
4] Vehicle Based on Improved A* Algorithm,» *Hindawi, Journal of Advanced Transportation*, 2022.
- [1 P. S. J.-C. L. M. H. O. Lidia E. Kavraki, «Probabilistic Roadmaps for Path Planning in
5] High-Dimensional Configuration Spaces,» *IEEE Transactions on Robotics and Automation*, 1996.
- [1 R. Z. W. H. W. J. a. o. Jiajia Chen, «Path Planning for Autonomous Vehicle Based on a
6] Two-Layered Planning Model in Complex Environment,» *Hindawi, Journal of Advanced Transportation*, 2020.
- [1 A. K. S.-K. C. B. L. Yadollah Rasehipour, «A Potential Field-Based Model Predictive
7] Path-Planning Controller for Autonomous Road Vehicles,» *IEEE Transactions on intelligent transportation systems*, 2016.
- [1 Z. C. S. Y. Z. B. G. F. Hu Hongyu, «An Improved Artificial Potential Field Model
8] Considering Vehicle Velocity for Autonomous Driving,» *ScienceDirect - IFAC Papers Inline*, 2018.
- [1 L. G. N. N. L. Y. G. J. W. H. HE Bing, «A Route Planning Method Based on Improved
9] Artificial Potential Field Algorithm,» *IEEE*, 2011.
- [2 P. M. S. V. F. A. F. G. Julien Moreau, «Path planning with fractional potential fields for
0] autonomous vehicles,» *ScienceDirect - IFAC Papers Online - ELSEVIER*, 2017.
- [2 T. T. K. E. Rafal Szczepanski, «Energy Efficient Local Path Planning Algorithm Based
1] on Predictive Artificial Potential Field,» *IEEE Vehicular Technology Society Section*, 2022.
- [2 H. W. A. K. H. D. K. Y. Y. Q. Yanjun Huang, «A Novel Local Motion Planning
2] Framework for Autonomous Vehicles based on resistance network and model predictive control,» *IEEE Transactions on Vehicular Technology*, 2019.
- [2 U. M. M. D. D. O. T. M. A. M. S. F. Shilp Dixit, «Trajectory Planning for Autonomous
3] High-Speed Overtaking in Structured Environments Using Robust MPC,» *IEEE Transactions on Intelligent Transportation Systems*, 2019.
- [2 T. Y. R. Z. Yingjie Liu, «Trajectory Tracking Model Predictive Controller Design for
4] Autonomous Vehicles with Updating Constraints of Tire Characteristics,» *MDPI - World Electric Vehicle Journal*, 2023.
- [2 MathWorks, «trajectoryOptimalFrenet - Matworks,» 2019. [Online]. Available:
5] <https://it.mathworks.com/help/nav/ref/trajectoryoptimalfrenet.html>.
- [2 M. V. L. G. Antonio Tota, «Path Tracking Control for Autonomous Driving
6] Applications,» *Mechanisms and Machine Science*, 2018.
- [2 E. G. M. V. M. C. Antonio Tota, «ARTICULATED STEERING CONTROL DESIGN
FOR AUTONOMOUS TRACKED VEHICLES,» *International Journal of Mechanics*

- 7] *and Control*, 2021.
- [2 A. S. P. G. M. Z. D. W. B. B. Christoforos Chatzikomis, «Comparison of Path Tracking
8] and Torque-Vectoring Controllers for Autonomous Electric Vehicles,» *IEEE
Transactions on Intelligent vehicles*, 2018.
- [2 J. M. Snider, «Automatic Steering Methods for Autonomous Automobile Path
9] Tracking,» Carneige Mellon University - Pittsburgh, Pennsylvania, 2009.
- [3 A. T. Mauro Velardocchia, *Appunti e Dispense di Meccanica del Veicolo*, Torino:
0] Politecnico di Torino, 2022.
- [3 S. IPG-CarMaker, Manuali IPG-CarMaker: QuickStartGuide, UserGuide,
1] ReferenceManual, ProgrammersGuide.
- [3 E. Ostertag, *Mono and Multivariable Control and Estimation*, Berlin: Springer, 2011.
2]
- [3 I. P. Sigurd Skogestad, *Multivariable feedback control - Analysis and design*, JOHN
3] WILEY & SONS.
- [3 B. Douglas, «What Is LQR Optimal Control?|State Space, Part 4,» MathWorks, 2023.
4] [Online]. Available: <https://it.mathworks.com/videos/state-space-part-4-what-is-lqr-control-1551955957637.html>.
- [3 B. Douglas, «Why the Ricatti Equation Is important for LQR Control,» MathWorks,
5] 2023. [Online]. Available: <https://it.mathworks.com/videos/why-the-riccati-equation-is-important-for-lqr-control-1689235310468.html>.
- [3 S. IPG-CarMaker, «Vehicle Dynamics,» IPG-CarMaker AUTOMOTIVE, [Online].
6] Available: <https://ipg-automotive.com/en/applications/vehicle-dynamics/>.
- [3 S. ASAM, «ASAM OpenCRG Version 1.2.0,» ASAM, 2020. [Online]. Available:
7] https://www.asam.net/index.php?eID=dumpFile&t=f&f=3950&token=21a7ae456ec0eb0f9ec3aee5bae3e8c9ebaea140#_usage_of_the_c_api.
- [3 MathWorks, «Plan 3D Paths for UAVs with RRT Algorithm,» MathWorks, 2023.
8] [Online]. Available: <https://it.mathworks.com/help/uav/ug/motion-planning-with-rrt-for-fixed-wing-uav.html>.