ΕΘΝΙΚΟ ΜΕΤΣΟΒΙΟ ΠΟΛΥΤΕΧΝΕΙΟ
ΤΜΗΜΑ ΗΛΕΚΤΡΟΛΟΓΩΝ ΜΗΧΑΝΙΚΩΝ ΚΑΙ ΜΗΧΑΝΙΚΩΝ ΥΠΟΛΟΓΙΣΤΩΝ

ΤΟΜΕΑΣ ΗΛΕΚΤΡΙΚΩΝ ΒΙΟΜΗΧΑΝΙΚΩΝ ΔΙΑΤΑΞΕΩΝ ΚΑΙ ΣΥΣΤΗΜΑΤΩΝ ΑΠΟΦΑΣΕΩΝ
ΕΡΓΑΣΤΗΡΙΟ ΥΨΗΛΩΝ ΤΑΣΕΩΝ ΚΑΙ ΗΛΕΚΤΡΙΚΩΝ ΜΕΤΡΗΣΕΩΝ

Υπολογιστική Προσομοίωση Συστήματος Κατανομής Ροπής

ΔΙΠΛΩΜΑΤΙΚΗ ΕΡΓΑΣΙΑ

του

ΒΑΣΙΛΕΙΟΥ Γ. ΔΗΜΗΤΡΟΠΟΥΛΟΥ

Επιβλέπων: Ιωάννης Φ. Γκόνος
Καθηγητής Ε.Μ.Π

Αθήνα, Οκτώβριος 2023

ΕΘΝΙΚΟ ΜΕΤΣΟΒΙΟ ΠΟΛΥΤΕΧΝΕΙΟ
ΤΜΗΜΑ ΗΛΕΚΤΡΟΛΟΓΩΝ ΜΗΧΑΝΙΚΩΝ ΚΑΙ ΜΗΧΑΝΙΚΩΝ ΥΠΟΛΟΓΙΣΤΩΝ

ΤΟΜΕΑΣ ΗΛΕΚΤΡΙΚΩΝ ΒΙΟΜΗΧΑΝΙΚΩΝ ΔΙΑΤΑΞΕΩΝ ΚΑΙ ΣΥΣΤΗΜΑΤΩΝ ΑΠΟΦΑΣΕΩΝ
ΕΡΓΑΣΤΗΡΙΟ ΥΨΗΛΩΝ ΤΑΣΕΩΝ ΚΑΙ ΗΛΕΚΤΡΙΚΩΝ ΜΕΤΡΗΣΕΩΝ

Υπολογιστική Προσομοίωση Συστήματος Κατανομής Ροπής

ΔΙΠΛΩΜΑΤΙΚΗ ΕΡΓΑΣΙΑ

του

ΒΑΣΙΛΕΙΟΥ Γ. ΔΗΜΗΤΡΟΠΟΥΛΟΥ

Επιβλέπων: Ιωάννης Φ. Γκόνος
Καθηγητής Ε.Μ.Π

Εγκρίθηκε από την τριμελή εξεταστική επιτροπή την 31η Οκτωβρίου 2023.

.....	.....	.....
Ιωάννης Φ. Γκόνος Καθηγητής Ε.Μ.Π	Ευάγγελος Χριστοφόρου Καθηγητής Ε.Μ.Π	Δημήτριος Κουλοχέρης Αν. Καθηγητής Ε.Μ.Π

Αθήνα, Οκτώβριος 2023.

.....

Βασίλειος Γ. Δημητρόπουλος
Διπλωματούχος Ηλεκτρολόγος Μηχανικός και Μηχανικός Υπολογιστών Ε.Μ.Π.

Περίληψη

Αυτή η Διπλωματική Εργασία περιλαμβάνει την ανάπτυξη υπολογιστικού κώδικα προσομοίωσης ενός αλγορίθμου συστήματος κατανομής ροπής (Torque Vectoring) και την αξιολόγηση αυτού στη δυναμική συμπεριφορά του οχήματος. Για την ανάπτυξη του υπολογιστικού κώδικα χρησιμοποιήθηκε η προγραμματιστική γλώσσα Python. Το σύστημα του Torque Vectoring (TV) παίζει καίριο ρόλο στη βελτίωση της σταθερότητας, του ελέγχου και της ασφάλειας των οχημάτων. Η παρούσα έρευνα εξετάζει τα θεωρητικά θεμέλια του TV και την πρακτική του εφαρμογή στον τομέα της δυναμικής οχημάτων.

Η μελέτη ξεκινά με την παρουσίαση ενός βασικού μοντέλου οχήματος, χρησιμοποιώντας το ως θεμελιώδη πλατφόρμα για όλες τις ακόλουθες διερευνήσεις. Έπειτα ακολουθεί εξέταση των αρχών και της μηχανικής του TV, συμπεριλαμβανομένων των πλεονεκτημάτων του και της πρακτικής εφαρμογής του στα σύγχρονα οχήματα. Σκοπός είναι η ανάλυση της φιλοσοφίας πίσω από το σύστημα κατανομής ροπής και των τρόπων με τους οποίους βελτιώνεται η κατανομή της διαθέσιμης ροπής του κινητήρα μεταξύ των τροχών για να ενισχύσει το κράτημα του οχήματος, τη σταθερότητα και τη συνολική απόδοση.

Η έρευνα επεκτείνει τις συνεισφορές της, αναπτύσσοντας έναν αλγόριθμο TV και ενσωματώνοντας τον στο μοντέλο του οχήματος. Το αποτέλεσμα αυτής της προσπάθειας είναι ένα αποδοτικό εργαλείο που έχει τη δυνατότητα να βελτιώσει τη συμπεριφορά των αυτοκινήτων σε ένα εύρος οδηγικών συνθηκών. Η αποτελεσματικότητα του αλγορίθμου δοκιμάζεται σε ελιγμούς με επιταχυνόμενη κίνηση, όπως στροφές και αποφυγή εμποδίων, καθένας εκ των οποίων εκτελείται με διάφορες αρχικές ταχύτητες.

Τα αποτελέσματα αυτής της Διπλωματικής Εργασίας αποκαλύπτουν ότι η υλοποίηση του αλγορίθμου βελτιώνει σημαντικά τη δυνατότητα του οχήματος να αντιμετωπίζει προκλητικές και επικίνδυνες καταστάσεις, ενισχύοντας όχι μόνο τη σταθερότητα και την ευελιξία, αλλά και τη συνολική ασφάλεια. Καθώς εισερχόμαστε σε μια εποχή εξελισσόμενων τεχνολογιών στα οχήματα και αυξημένων απαιτήσεων για ασφάλεια, η έρευνα που παρουσιάζεται σε αυτή τη διατριβή κατέχει σημαντικό ρόλο στον προσδιορισμό του μέλλοντος της αυτοκινητοβιομηχανίας.

Εν κατακλείδι, η παρούσα έρευνα αποτελεί ακόμα ένα βήμα προς την προσπάθεια να αξιοποιηθεί πλήρως το δυναμικό των αλγορίθμων κατανομής ροπής για το όφελος της αυτοκινητοβιομηχανίας και του γενικού κοινού, οδηγώντας τελικά σε πιο ασφαλείς, αποδοτικές και ευχάριστες εμπειρίες οδήγησης.

Λέξεις - κλειδιά

Κατανομή Ροπής Διανομή Ροπής, Αλγόριθμος Κατανομής Ροπής, Δυναμική Οχημάτων, Μηχανική Οχημάτων, Αλγόριθμοι Διανομής Ισχύος, Βελτιστοποίηση Δυναμικής Οχήματος, Έλεγχος Ευστάθειας, Βελτίωση Επιδόσεων, Αποφυγή Εμποδίου, Ευελιξία, Ασφάλεια Οχήματος, Προγραμματιστική Γλώσσα Python.

Abstract

This Diploma Thesis is a development of a computational simulation code of a torque distribution algorithm (Torque Vectoring) and its evaluation in the dynamic behavior of the vehicle. The programming language Python was used for this development. Torque vectoring plays a pivotal role in enhancing stability, control, and safety. The research herein delves into the theoretical underpinnings of torque vectoring and its practical application within the domain of vehicle dynamics.

The study begins with the presentation of a baseline vehicle model, employed as the foundational platform for all subsequent investigations. An examination of the principles and mechanics of torque vectoring, inclusive of its inherent advantages and real-world applicability, follows. The research endeavors to analyze how Torque Vectoring optimizes the distribution of torque between the wheels to enhance the vehicle's cornering capabilities, stability, and overall performance.

This research extends its contributions by developing a Torque Vectoring algorithm and implementing it into the vehicle model. The outcome of this endeavor is an efficient tool that has the potential to revolutionize vehicular dynamics. The algorithm's effectiveness is tested across a range of scenarios, spanning maneuvers like accelerating left turns, step-steer maneuvers, and obstacle avoidance, each executed at varying initial velocities.

The findings of this Diploma Thesis reveal the profound influence of torque vectoring on vehicle dynamics. The algorithm's implementation significantly improves the vehicle's ability to navigate through challenging scenarios, enhancing not only its stability but also its maneuverability and overall safety. As we step into an era of evolving vehicle technologies and heightened safety concerns, the research presented in this thesis holds substantial importance in shaping the future of automotive engineering.

In conclusion, this thesis represents a vital step in the ongoing quest to harness the full potential of Torque Vectoring to benefit both the automotive industry and the general public, ultimately leading to safer, more efficient, and more enjoyable driving experiences.

Keywords

Torque Vectoring, Torque Vectoring Algorithm, Vehicle Dynamics, Automotive Engineering, Torque Distribution, Power Distribution Algorithms, Enhanced Vehicle Handling, Vehicle Dynamics Optimization, Stability Control, Cornering Performance, Obstacle Avoidance, Maneuverability, Vehicle Safety, Multi-Body Vehicle Model, Python.

Ευχαριστίες

Αρχικά, θα ήθελα να εκφράσω τις ευχαριστίες μου στους καθηγητές μου και στη μεταδιδακτορική ερευνήτρια κα. Κλειώ Βόσου, οι οποίοι με καθοδήγησαν και με στήριξαν καθ' όλη τη διάρκεια της συγγραφής αυτής της διατριβής. Οι τεχνικές τους γνώσεις, οι συμβουλές τους και η ακλόνητη υπομονή τους στην αντιμετώπιση της πολυπλοκότητας του θέματος που ερευνάται υπήρξαν μεγάλη βοήθεια.

Δεν υπάρχει γραπτός τρόπος να εκφράσω το βάθος της απέραντης ευγνωμοσύνης προς τους γονείς μου, Ελένη και Γιώργο. Η αμέριστη υποστήριξή τους, τόσο ψυχολογικά όσο και οικονομικά, αποτέλεσε την κινητήρια δύναμη του ταξιδιού μου και οι συμβουλές τους αποτέλεσαν πηγή ανεκτίμητης σοφίας. Χάρη στην αδιάκοπη αγάπη και υποστήριξή τους μπόρεσα να ξεκινήσω αυτή την ακαδημαϊκή προσπάθεια και να φτάσω στο σημείο που βρίσκομαι σήμερα.

Θα ήθελα επίσης να εκφράσω την εκτίμησή μου στον αδελφό μου, Παναγιώτη, του οποίου το ταλέντο στο να με εκνευρίζει φαίνεται να ανταγωνίζεται την πολυπλοκότητα των αλγορίθμων κατανομής ροπής. Σαν ένα απρόβλεπτο σφάλμα λογισμικού, είναι πάντα εκεί, εισάγοντας μια δόση χάους στον κώδικα της ζωής μου. Αλλά ακριβώς όπως η κατανομή ροπής βελτιώνει την απόδοση του οχήματος, έτσι και τα καμώματά του με έχουν διδάξει να αντιμετωπίζω τις ανωμαλίες στο δρόμο της ζωής με χιούμορ και χάρη.

Αφιερώνω την παρούσα Διπλωματική Εργασία στους γονείς μου, Ελένη και Γιώργο, ως έναν μικρό τρόπο να εκφράσω την απέραντη αγάπη και ευγνωμοσύνη μου για όλα όσα μου έχουν δώσει.

Περιεχόμενα

Κατάλογος σχημάτων

Κατάλογος πινάκων

Κατάλογος κώδικα

Κεφάλαιο 1

Εισαγωγή

1.1 Εισαγωγή στο Torque Vectoring: Έννοια και συνοπτική περιγραφή

Η κατανομή ροπής (Torque Vectoring) αποτελεί μια σημαντική τεχνολογική προσέγγιση στον τομέα της οδικής συμπεριφοράς και της δυναμικής των οχημάτων. Στον πυρήνα της τεχνολογίας αυτής βρίσκεται η ικανότητα ενός οχήματος να ελέγχει και να ρυθμίζει την κατανομή της ροπής μεταξύ των τροχών, προσφέροντας έτσι βελτιωμένο κράτημα κατά την οδήγηση. Αυτή η τεχνολογία αποτελεί το κέντρο του εξελισσόμενου κλάδου των αυτοκινήτων και διευρύνει τα όρια της οδικής ασφάλειας, της ευελιξίας και της απόδοσης.

Ο βασικός σκοπός του Torque Vectoring (TV) είναι η κατανομή της ροπής μεταξύ των τροχών, εξασφαλίζοντας ότι ο καθένας επιλαμβάνεται του κατάλληλου μεριδίου ανάλογα με την κατάσταση του οδοστρώματος και τις απαιτήσεις της οδήγησης. Αυτό επιτυγχάνεται μέσω της προηγμένης χρήσης των διαφορικών, των φρένων και των κινητήρων, καθώς και μέσω συστημάτων ελέγχου και αισθητήρων. Το TV μπορεί να προσαρμοστεί δυναμικά κατά την οδήγηση, λαμβάνοντας υπόψη παράγοντες όπως την ταχύτητα, την επιτάχυνση, το ρυθμό εκτροπής του οχήματος και άλλες παραμέτρους.

Το σύστημα αυτό βρίσκει εφαρμογή σε ποικιλία οχημάτων, από τα καθημερινά, μέχρι τα αυτοκίνητα υψηλής απόδοσης και τα αγωνιστικά και μπορεί να βελτιώσει την πρόσφυση και την απόδοση στις στροφές. Σε συνθήκες κακοκαιρίας ή απώλειας ελέγχου, μπορεί να συμβάλλει στην ευστάθεια και στην αποφυγή επικίνδυνων καταστάσεων.

Η τεχνολογία της κατανομής ροπής συνεχίζει να εξελίσσεται, προσφέροντας νέες δυνατότητες για τη βελτίωση των επιδόσεων των οχημάτων και την αύξηση της οδικής ασφάλειας. Η κατανόηση της λειτουργίας της αποτελεί κρίσιμη προϋπόθεση για την εξέλιξη και την εφαρμογή της στην επόμενη γενιά των οχημάτων.

1.2 Σκοπός και Στόχοι

Σκοπός της παρούσας Διπλωματικής Εργασίας είναι η προσομοίωση της επιρροής του TV στη δυναμική συμπεριφορά του αυτοκινήτου κατά τη διάρκεια ελιγμών.

Στόχος είναι η εξ' ολοκλήρου αναπαράσταση της δυναμικής ενός μοντέλου οχήματος σε προγραμματιστική γλώσσα Python και έπειτα η δημιουργία ενός συστήματος αυτομάτου ελέγχου, το οποίο θα προσαρμοστεί σε αυτό το μοντέλο. Η λειτουργία του συστήματος αυτού έγκειται στη δυναμική κατανομή της διαθέσιμης ροπής του κινητήρα σε κάθε τροχό, λαμβάνοντας υπόψη παράγοντες που θα περιγραφούν αναλυτικά σε επόμενο κεφάλαιο. Κριτήριο αξιολόγησης του συστήματος θα είναι οι επιδόσεις και η ασφάλεια κατά τη διάρκεια των ελιγμών.

1.3 Συμβολή

Η παρούσα Διπλωματική Εργασία συνεισφέρει παρέχοντας γνώση σχετικά με τον έλεγχο της δυναμικής των οχημάτων και παρουσιάζει τρόπους με τους οποίους μπορούν να χρησιμοποιηθούν στρατηγικές ελέγχου για τη βελτίωση της οδηγικής συμπεριφοράς τους. Συμβάλλει επίσης στην εμβάθυνση της κατανόησης του τρόπου με τον οποίο το TV επηρεάζει την οδηγική συμπεριφορά του αυτοκινήτου σε διάφορες καταστάσεις και αναδεικνύει τον αντίκτυπο που έχουν η υποστροφή (under-steer) και η υπερστροφή (over-steer) στις επιδόσεις κατά τη διάρκεια ελιγμών. Η σημαντικότερη ίσως εφαρμογή των παραπάνω έγκειται στη δημιουργία ασφαλέστερων οχημάτων προς όφελος της κοινωνίας. Η συμβολή επεκτείνεται μέσω της χρήσης της προγραμματιστικής γλώσσας Python, η οποία είναι open-source, δηλαδή ο κώδικας διατίθεται ελεύθερα και μπορεί να αναδιανεμηθεί και να τροποποιηθεί. Επομένως, η παρούσα Διπλωματική Εργασία αποτελεί σημαντική βάση για περαιτέρω ανάλυση του θέματος που ερευνάται.

Κεφάλαιο 2

Μοντελοποίηση οχήματος

Στην παρούσα Διπλωματική Εργασία θα χρησιμοποιηθεί κώδικας μοντελοποίησης του αυτοκινήτου από το αποθετήριο CommonRoad [1], το οποίο παρέχει υλοποιήσεις της συμπεριφοράς διάφορων οχημάτων και διαδικασίες για το μετασχηματισμό των αρχικών καταστάσεων.

Το αποθετήριο CommonRoad παρέχει παραμέτρους για τέσσερα είδη αυτοκινήτων:

- μικρού μεγέθους (Ford Escort; vehicle ID: 1)
- μεσαίου μεγέθους (BMW 320i; vehicle ID: 2)
- van (VW Vanagon; vehicle ID: 3)
- ημιρυμουλκούμενο φορτηγό (vehicle ID: 4)

Λεπτομερείς παράμετροι αυτών των οχημάτων έχουν ληφθεί από το [2], Παράρτημα Α και από άλλες διαθέσιμες στο διαδίκτυο βάσεις δεδομένων οχημάτων.

Στην παρούσα έρευνα θα χρησιμοποιηθεί το όχημα 2, ένα BMW 320i. Για το συγκεκριμένο όχημα παρέχονται τα ακόλουθα μοντέλα αναπαράστασης δυναμικής:

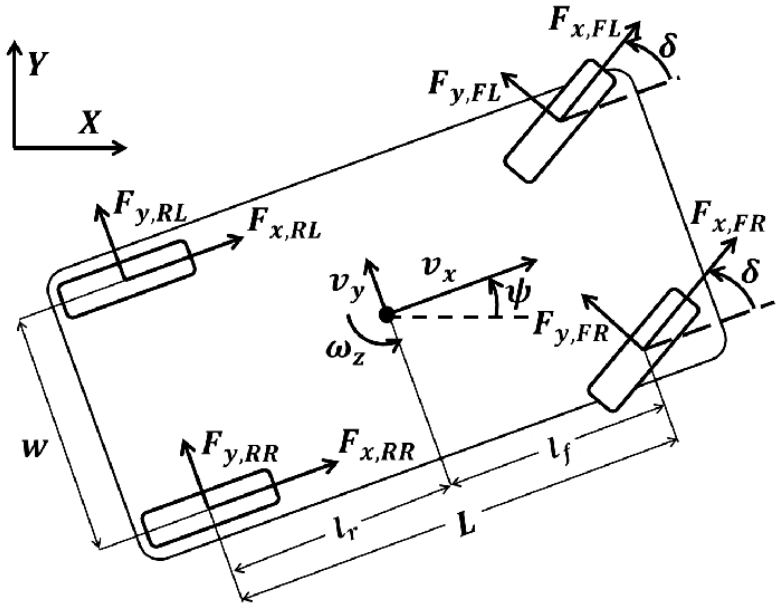
- μοντέλο Point-Mass (PM)
- μοντέλο Kinematic Single-Track (KS)
- μοντέλο Single-Track (ST)
- μοντέλο Single-Track Drift (STD)
- μοντέλο Multi-Body (MB)

Το μοντέλο που θα χρησιμοποιηθεί για την εκτέλεση και την αξιολόγηση των ελιγμών πριν και μετά την εφαρμογή του TV είναι το Multi-Body (μοντέλο πλήρους οχήματος), καθώς είναι το πιο περίπλοκο όλων και αναπαριστά με μεγάλη ακρίβεια τη συμπεριφορά του οχήματος σε πραγματικές συνθήκες οδήγησης. Θα γίνει ωστόσο σύντομη χρήση και του μοντέλου Single-Track Drift (μοντέλο μισού οχήματος - bicycle) για τη δημιουργία

της ιδανικής τροχιάς του αυτοκινήτου κατά τη διάρκεια ενός από τους ελιγμούς που θα εκτελεσθούν.

Το συγκεκριμένο μοντέλο προέρχεται από το [2], Παράρτημα Α, το οποίο αποτελεί μία από τις λίγες λεπτομερείς και προσβάσιμες περιγραφές δυναμικής πολυσωμάτων οχημάτων. Για ευκολία χρήσης, έχει γίνει μετάφραση των εξισώσεων από το [2], Παράρτημα Α σε ένα μοντέλο χώρου καταστάσεων, το οποίο είναι πιο κατάλληλο για την υλοποίηση σε κλασσικούς επιλυτές διαφορικών εξισώσεων.

Η δυναμική του μοντέλου Multi-body (MB) περιγράφεται από 2 μάζες: Τη μη αναρτώμενη και την αναρτώμενη μάζα του εμπρόσθιου και του οπίσθιου άξονα. Οι δυνάμεις μεταξύ αυτών των μαζών περιγράφονται από τη δυναμική της ανάρτησης και το μοντέλο του ελαστικού. Λαμβάνονται υπόψη όλες οι δυνάμεις της ανάρτησης που προέρχονται από ελατήρια, αποσβεστήρες και αντιστρεπτικές μπάρες, όπως ακριβώς στο [2], Παράρτημα Α. Δε λαμβάνεται υπόψη η ελαστικότητα στο σύστημα διεύθυνσης, τα bump stops, και οι δυνάμεις squat/lift που προκαλούνται από τη γεωμετρία της ανάρτησης. Το εξεταζόμενο όχημα διαθέτει ανεξάρτητη ανάρτηση, οπότε δεν παρουσιάζονται οι εξισώσεις για στερεούς άξονες. Για την αναπαράσταση της δυναμικής των ελαστικών επιλέχθηκε το μοντέλο PAC2002 Magic-Formula, το οποίο χρησιμοποιείται ευρέως στη βιομηχανία [3]. Οι συνδυασμένες πλευρικές και διαμήκεις δυνάμεις του ελαστικού υπολογίζονται από τη γωνία ολίσθησης, τη γωνία κάμψης και την κάθετη δύναμη του ελαστικού που περιγράφεται στο [2], Παράρτημα Α. Οι παράμετροι των ελαστικών για όλους τους τροχούς προέρχονται από το παράδειγμα αρχείου ιδιοτήτων ελαστικού PAC2002 στο [3]. Η μετάφραση όλων των εξισώσεων ως μοντέλο χώρου καταστάσεων οδηγεί σε 29 μεταβλητές κατάστασης. Όλες οι μεταβλητές κατάστασης, συμπεριλαμβανομένων των αρχικών τους τιμών, αναφέρονται στον Πίνακα 2.1, όπου τα ζεύγη LF, RF, LR, RR υποδεικνύουν [αριστερά/δεξιά] και [μπροστά/πίσω]. Σε σύγκριση με το [2], Παράρτημα Α, οι εξισώσεις παρουσιάζονται με τέτοιο τρόπο ώστε να εξαρτώνται από τα προηγουμένως υπολογισμένα αποτελέσματα, καθιστώντας δυνατή την άμεση υλοποίησή τους. Παρακάτω φαίνεται ένα μοντέλο πλήρους οχήματος.



Σχήμα 2.1: Μοντέλο πλήρους οχήματος

2.1 Μεταβλητές κατάστασης

Το μοντέλο MB του αποθετηρίου Commonroad που θα χρησιμοποιηθεί, ομαδοποιεί τις μεταβλητές κατάστασης σε *σώμα οχήματος, εμπρόσθιος άξονας, οπίσθιος άξονας, τροχοί* και *βοηθητικές*.

Σώμα οχήματος

$x_1 = s_x$ (x-θέση στο σύστημα συντεταγμένων),
$x_2 = s_y$ (y-θέση στο σύστημα συντεταγμένων),
$x_3 = \delta$ (γωνία διεύθυνσης των εμπρόσθιων τροχών),
$x_4 = u_x$ (ταχύτητα στη διαμήκη διεύθυνση),
$x_5 = \Psi$ (γωνία εκτροπής),
$x_6 = \dot{\Psi}$ (ρυθμός εκτροπής),
$x_7 = \Phi_S$ (γωνία roll),
$x_8 = \dot{\Phi}_S$ (ρυθμός roll),
$x_9 = \Theta_S$ (γωνία pitch),
$x_{10} = \dot{\Theta}_S$ (ρυθμός pitch),
$x_{11} = u_y$ (ταχύτητα στην πλευρική διεύθυνση),
$x_{12} = s_z$ (z-θέση (ύψος) από το έδαφος),
$x_{13} = u_z$ (ταχύτητα στην κατακόρυφη διεύθυνση, κάθετη στο επίπεδο του δρόμου),

Εμπρόσθιος άξονας

$x_{14} = \Phi_{UF}$ (εμπρόσθια γωνία roll),
$x_{15} = \dot{\Phi}_{UF}$ (εμπρόσθιος ρυθμός roll),
$x_{16} = v_{y,UF}$ (εμπρόσθια ταχύτητα στην y-διεύθυνση),
$x_{17} = s_{z,UF}$ (εμπρόσθια z-θέση),
$x_{18} = v_{z,UF}$ (εμπρόσθια ταχύτητα στην z-διεύθυνση),

Οπίσθιος άξονας

$x_{19} = \Phi_{UR}$ (οπίσθια γωνία roll),
$x_{20} = \dot{\Phi}_{UR}$ (οπίσθιος ρυθμός roll),
$x_{21} = v_{y,UR}$ (οπίσθια ταχύτητα στην y-διεύθυνση),
$x_{22} = s_{z,UR}$ (οπίσθια z-θέση),
$x_{23} = v_{z,UR}$ (οπίσθια ταχύτητα στην z-διεύθυνση),

Τροχοί

$x_{24} = \omega_{LF}$ (γωνιακή ταχύτητα εμπρόσθιου αριστερού τροχού),
$x_{25} = \omega_{RF}$ (γωνιακή ταχύτητα εμπρόσθιου δεξιού τροχού),
$x_{26} = \omega_{LR}$ (γωνιακή ταχύτητα οπίσθιου αριστερού τροχού),
$x_{27} = \omega_{RR}$ (γωνιακή ταχύτητα οπίσθιου δεξιού τροχού),

$x_{28} = \delta_{y,f}$ (εμπρόσθια πλευρική μετατόπιση αναρτώμενης μάζας λόγω roll),
$x_{29} = \delta_{y,r}$ (οπίσθια πλευρική μετατόπιση αναρτώμενης μάζας λόγω roll).

2.2 Βοηθητικές μεταβλητές

Γωνία ολίσθησης και ταχύτητα στο κέντρο βάρους

Αυτές οι εξισώσεις προέρχονται από το CommonRoad:

$$\beta = \arctan\left(\frac{x_{11}}{x_4}\right)$$
$$v_{CG} = \sqrt{x_4^2 + x_{11}^2}$$

Κατακόρυφες δυνάμεις ελαστικών

Αυτές οι εξισώσεις λαμβάνονται από [2], εξ. A48-A51:

$$F_{z,LF} = \left(x_{17} + R_w\left(\cos\left(x_{14}\right) - 1\right) - \frac{1}{2}T_f \sin\left(x_{14}\right)\right) K_{zt}$$
$$F_{z,RF} = \left(x_{17} + R_w\left(\cos\left(x_{14}\right) - 1\right) + \frac{1}{2}T_f \sin\left(x_{14}\right)\right) K_{zt}$$
$$F_{z,LR} = \left(x_{22} + R_w\left(\cos\left(x_{19}\right) - 1\right) - \frac{1}{2}T_r \sin\left(x_{19}\right)\right) K_{zt}$$
$$F_{z,RR} = \left(x_{22} + R_w\left(\cos\left(x_{19}\right) - 1\right) + \frac{1}{2}T_r \sin\left(x_{19}\right)\right) K_{zt}$$

Μεμονωμένες ταχύτητες τροχών

Αυτές οι εξισώσεις προέρχονται από [2], εξ. A56-A59 υποθέτοντας ότι οι πίσω τροχοί δε μπορούν να κατευθύνουν το όχημα και χρησιμοποιώντας $x_4 \tan(\beta) = x_{11}$ από [2], σελ. A45:

$$u_{w,LF} = \left(x_4 + \tfrac{1}{2}T_f x_6\right)\cos\left(x_3\right) + \left(x_{11} + l_f x_6\right)\sin\left(x_3\right)$$
$$u_{w,RF} = \left(x_4 - \tfrac{1}{2}T_f x_6\right)\cos\left(x_3\right) + \left(x_{11} + l_f x_6\right)\sin\left(x_3\right)$$
$$u_{w,LR} = x_4 + \tfrac{1}{2}T_r x_6$$
$$u_{w,RR} = x_4 - \tfrac{1}{2}T_r x_6$$

Διαμήκης ολίσθηση

Αυτές οι εξισώσεις λαμβάνονται από [2], εξ. A60:

$$
\begin{aligned}
s_{LF} &= 1 - \frac{R_w x_{24}}{u_{w,LF}} \\
s_{RF} &= 1 - \frac{R_w x_{25}}{u_{w,RF}} \\
s_{LR} &= 1 - \frac{R_w x_{26}}{u_{w,LR}} \\
s_{RR} &= 1 - \frac{R_w x_{27}}{u_{w,RR}}
\end{aligned}
$$

Πλευρικές γωνίες ολίσθησης

Αυτές οι εξισώσεις προέρχονται από [2], εξ. A42-A45, υποθέτοντας ότι οι οπίσθιοι τροχοί δε μπορούν να κατευθυνθούν:

$$
\begin{aligned}
\alpha_{LF} &= \arctan\left(\frac{x_{11} + l_f x_6 - x_{15}\left(R_w - x_{17}\right)}{x_4 + \frac{1}{2}T_f x_6}\right) - x_3 \\
\alpha_{RF} &= \arctan\left(\frac{x_{11} + l_f x_6 - x_{15}\left(R_w - x_{17}\right)}{x_4 - \frac{1}{2}T_f x_6}\right) - x_3 \\
\alpha_{LR} &= \arctan\left(\frac{x_{11} - l_r x_6 - x_{20}\left(R_w - x_{22}\right)}{x_4 + \frac{1}{2}T_r x_6}\right) \\
\alpha_{RR} &= \arctan\left(\frac{x_{11} - l_r x_6 - x_{20}\left(R_w - x_{22}\right)}{x_4 - \frac{1}{2}T_r x_6}\right)
\end{aligned}
$$

Βοηθητική κίνηση ανάρτησης

Αυτές οι εξισώσεις λαμβάνονται από [2], εξ. A23a-A26a και [2], εξ. A23b-A26b:

$$z_{S,LF} = \frac{h_s - R_w + x_{17} - x_{12}}{\cos(x_7)} - h_s + R_w + l_f x_9 + \frac{1}{2}(x_7 - x_{14})\, T_f$$
$$z_{S,RF} = \frac{h_s - R_w + x_{17} - x_{12}}{\cos(x_7)} - h_s + R_w + l_f x_9 - \frac{1}{2}(x_7 - x_{14})\, T_f$$
$$z_{S,LR} = \frac{h_s - R_w + x_{22} - x_{12}}{\cos(x_7)} - h_s + R_w - l_r x_9 + \frac{1}{2}(x_7 - x_{19})\, T_r$$
$$z_{S,RR} = \frac{h_s - R_w + x_{22} - x_{12}}{\cos(x_7)} - h_s + R_w - l_r x_9 - \frac{1}{2}(x_7 - x_{19})\, T_r$$

$\dot{z}_{S,LF} = x_{18} - x_{13} + l_f x_{10} + \frac{1}{2}(x_8 - x_{15})\, T_f$

$\dot{z}_{S,RF} = x_{18} - x_{13} + l_f x_{10} - \frac{1}{2}(x_8 - x_{15})\, T_f$

$\dot{z}_{S,LR} = x_{23} - x_{13} - l_r x_{10} + \frac{1}{2}(x_8 - x_{20})\, T_r$

$\dot{z}_{S,RR} = x_{23} - x_{13} - l_r x_{10} - \frac{1}{2}(x_8 - x_{20})\, T_r$

(Το '-' άλλαξε σε '+' σε σύγκριση με το [2], εξ. A26b)

Γωνίες Camber

Αυτές οι εξισώσεις λαμβάνονται από [2], εξ. A27-A30:

$$\gamma_{LF} = x_7 + D_f z_{S,LF} + E_f (z_{S,LF})^2$$
$$\gamma_{RF} = x_7 - D_f z_{S,RF} - E_f (z_{S,RF})^2$$
$$\gamma_{LR} = x_7 + D_r z_{S,LR} + E_r (z_{S,LR})^2$$
$$\gamma_{RR} = x_7 - D_r z_{S,RR} - E_r (z_{S,RR})^2$$

Βοηθητικές κινήσεις/δυνάμεις για τις εξισώσεις ευέλικτων αρθρώσεων

Αυτές οι εξισώσεις λαμβάνονται από [2], εξ. A61-A68:

$$
\begin{aligned}
&\Delta z_F = h_s - R_w + x_{17} - x_{12} \\
&\Delta z_R = h_s - R_w + x_{22} - x_{12} \\
&\Delta \phi_F = x_7 - x_{14} \\
&\Delta \phi_R = x_7 - x_{19} \\
&\Delta \dot{\phi}_F = x_8 - x_{15} \\
&\Delta \dot{\phi}_R = x_8 - x_{20} \\
&\Delta \dot{z}_F = x_{18} - x_{13} \\
&\Delta \dot{z}_R = x_{23} - x_{13} \\
&\Delta \dot{y}_F = x_{11} + l_f x_6 - x_{16} \\
&\Delta \dot{y}_R = x_{11} - l_r x_6 - x_{21} \\
&\Delta_F = \Delta z_F \sin(x_7) - x_{28} \cos(x_7) - (h_{RAF} - R_w) \sin(\Delta \phi_F) \\
&\Delta_R = \Delta z_R \sin(x_7) - x_{29} \cos(x_7) - (h_{RAR} - R_w) \sin(\Delta \phi_R) \\
&\dot{\Delta}_F = (\Delta z_F \cos(x_7) + x_{28} \sin(x_7))\, x_8 + \Delta \dot{z}_F \sin(x_7) - \Delta \dot{y}_F \cos(x_7) \\
&- (h_{RAF} - R_w) \cos(\Delta \phi_F)\, \Delta \dot{\phi}_F \\
&\dot{\Delta}_R = (\Delta z_R \cos(x_7) + x_{29} \sin(x_7))\, x_8 + \Delta \dot{z}_R \sin(x_7) - \Delta \dot{y}_R \cos(x_7) \\
&- (h_{RAR} - R_w) \cos(\Delta \phi_R)\, \Delta \dot{\phi}_R \\
&F_{RAF} = \Delta_F K_{RAS} + \dot{\Delta}_F K_{RAD} \\
&F_{RAR} = \Delta_R K_{RAS} + \dot{\Delta}_R K_{RAD}
\end{aligned}
$$

Βοηθητικές δυνάμεις ανάρτησης (αγνοούνται οι δυνάμεις squat/lift και το bump stop)

Αυτές οι εξισώσεις λαμβάνονται από το [2], εξ. A23-A26 και σελ. A51:

$$
\begin{aligned}
F_{S,LF} &= \frac{m_s g l_r}{2\left(l_f + l_r\right)} - z_{S,LF} K_{S,F} - \dot{z}_{S,LF} K_{SD,F} + \frac{\left(x_7 - x_{14}\right) K_{TS,F}}{T_f} \\
F_{S,RF} &= \frac{m_s g l_r}{2\left(l_f + l_r\right)} - z_{S,RF} K_{S,F} - \dot{z}_{S,RF} K_{SD,F} - \frac{\left(x_7 - x_{14}\right) K_{TS,F}}{T_f} \\
F_{S,LR} &= \frac{m_s g l_f}{2\left(l_f + l_r\right)} - z_{S,LR} K_{S,R} - \dot{z}_{S,LR} K_{SD,R} + \frac{\left(x_7 - x_{19}\right) K_{TS,R}}{T_r} \\
F_{S,RR} &= \frac{m_s g l_f}{2\left(l_f + l_r\right)} - z_{S,RR} K_{S,R} - \dot{z}_{S,RR} K_{SD,R} - \frac{\left(x_7 - x_{19}\right) K_{TS,R}}{T_r}
\end{aligned}
$$

Βοηθητικές μεταβλητές - Αναρτώμενη μάζα

Αυτές οι εξισώσεις λαμβάνονται από [2], εξ. A7-A12:

$$
\begin{aligned}
\sum X =& F_{x,LR} + F_{x,RR} + \left(F_{x,LF} + F_{x,RF}\right)\cos\left(x_3\right) - \left(F_{y,LF} + F_{y,RF}\right)\sin\left(x_3\right) \\
\sum N =& \left(F_{y,LF} + F_{y,RF}\right) l_f \cos\left(x_3\right) + \left(F_{x,LF} + F_{x,RF}\right) l_f \sin\left(x_3\right) \\
&+ \left(F_{y,RF} - F_{y,LF}\right) \frac{1}{2} T_f \sin\left(x_3\right) + \left(F_{x,LF} - F_{x,RF}\right) \frac{1}{2} T_f \cos\left(x_3\right) \\
&+ \left(F_{x,LR} - F_{x,RR}\right) \frac{1}{2} T_r - \left(F_{y,LR} + F_{y,RR}\right) l_r \\
\sum Y_s =& \left(F_{RAF} + F_{RAR}\right)\cos\left(x_7\right) + \left(F_{S,LF} + F_{S,LR} + F_{S,RF} + F_{S,RR}\right)\sin\left(x_7\right) \\
\sum L =& \frac{1}{2} F_{S,LF} T_f + \frac{1}{2} F_{S,LR} T_r - \frac{1}{2} F_{S,RF} T_f - \frac{1}{2} F_{S,RR} T_r \\
&- \frac{F_{RAF}}{\cos\left(x_7\right)} \left(h_s - x_{12} - R_w + x_{17} - \left(h_{RAF} - R_w\right)\cos\left(x_{14}\right)\right) \\
&- \frac{F_{RAR}}{\cos\left(x_7\right)} \left(h_s - x_{12} - R_w + x_{22} - \left(h_{RAR} - R_w\right)\cos\left(x_{19}\right)\right) \\
\sum Z_s =& \left(F_{S,LF} + F_{S,LR} + F_{S,RF} + F_{S,RR}\right)\cos\left(x_7\right) - \left(F_{RAF} + F_{RAR}\right)\sin\left(x_7\right) \\
\sum M_s =& l_f \left(F_{S,LF} + F_{S,RF}\right) - l_r \left(F_{S,LR} + F_{S,RR}\right) + \left(\left(F_{x,LF} + F_{x,RF}\right)\cos\left(x_3\right)\right. \\
&\left. - \left(F_{y,LF} + F_{y,RF}\right)\sin\left(x_3\right) + F_{x,LR} + F_{x,RR}\right)\left(h_s - x_{12}\right)
\end{aligned}
$$

Βοηθητικές μεταβλητές - Μη αναρτώμενη μάζα

Αυτές οι εξισώσεις λαμβάνονται από [2], εξ. A20-A22, υποθέτοντας ότι μόνο οι εμπρόσθιοι τροχοί μπορούν να κατευθυνθούν:

$$
\begin{aligned}
\sum L_{uf} =& \frac{1}{2}F_{S,RF}T_f - \frac{1}{2}F_{S,LF}T_f - F_{RAF}\left(h_{RAF} - R_w\right) \\
&+ F_{z,LF}\left(R_w \sin\left(x_{14}\right) + \frac{1}{2}T_f \cos\left(x_{14}\right) - K_{LT}F_{y,LF}\right) \\
&- F_{z,RF}\left(-R_w \sin\left(x_{14}\right) + \frac{1}{2}T_f \cos\left(x_{14}\right) + K_{LT}F_{y,RF}\right) \\
&- \left(\left(F_{y,LF} + F_{y,RF}\right)\cos\left(x_3\right) + \left(F_{x,LF} + F_{x,RF}\right)\sin\left(x_3\right)\right)\left(R_w - x_{17}\right) \\
\sum L_{ur} =& \frac{1}{2}F_{S,RR}T_r - \frac{1}{2}F_{S,LR}T_r - F_{RAR}\left(h_{RAR} - R_w\right) \\
&+ F_{z,LR}\left(R_w \sin\left(x_{19}\right) + \frac{1}{2}T_r \cos\left(x_{19}\right) - K_{LT}F_{y,LR}\right) \\
&- F_{z,RR}\left(-R_w \sin\left(x_{19}\right) + \frac{1}{2}T_r \cos\left(x_{19}\right) + K_{LT}F_{y,RR}\right) \\
&- \left(F_{y,LR} + F_{y,RR}\right)\left(R_w - x_{22}\right) \\
\sum Z_{uf} =& F_{z,LF} + F_{z,RF} + F_{RAF}\sin\left(x_7\right) - \left(F_{S,LF} + F_{S,RF}\right)\cos\left(x_7\right) \\
\sum Z_{ur} =& F_{z,LR} + F_{z,RR} + F_{RAR}\sin\left(x_7\right) - \left(F_{S,LR} + F_{S,RR}\right)\cos\left(x_7\right) \\
\sum Y_{uf} =& \left(F_{y,LF} + F_{y,RF}\right)\cos\left(x_3\right) + \left(F_{x,LF} + F_{x,RF}\right)\sin\left(x_3\right) \\
&- F_{RAF}\cos\left(x_7\right) - \left(F_{S,LF} + F_{S,RF}\right)\sin\left(x_7\right) \\
\sum Y_{ur} =& \left(F_{y,LR} + F_{y,RR}\right) \\
&- F_{RAR}\cos\left(x_7\right) - \left(F_{S,LR} + F_{S,RR}\right)\sin\left(x_7\right)
\end{aligned}
$$

Διάνυσμα εισόδου

Το διάνυσμα εισόδου είναι:

$$u = (u_1, u_2),\ \text{όπου}\ u_1 = v_\delta\ \text{και}\ u_2 = a_{\text{long}} \tag{2.1}$$

2.3 Μοντέλο ελαστικών

Ο κώδικας του CommonRoad χρησιμοποιεί το μοντέλο ελαστικών Pacejka 2002 [4], το οποίο είναι ένα από τα πιο δημοφιλή μοντέλα ελαστικών. Οι ακριβείς παράμετροι που θα χρησιμοποιηθούν στην παρούσα Διπλωματική Εργασία για την αναπαράσταση ενός ρεαλιστικού ελαστικού έχουν αποσπαστεί από το [3]. Για το μοντέλο γίνονται οι ακόλουθες παραδοχές:

- Η ολίσθηση της στροφής (turn slip) αμελείται, έτσι ώστε $\forall i : \xi_i = 1$
- Η επίδραση της αύξησης του φορτίου αγνοείται, έτσι ώστε $df_z = 0$ (βλ. [3], PAC2002, εξ. 16)
- Όλοι οι συντελεστές κλιμάκωσης ορίζονται ως $\forall i : \lambda_i = 1$.

Διαμήκεις δυνάμεις ελαστικών χρησιμοποιώντας τη Magic Formula για καθαρή ολίσθηση (pure slip) $\forall i \in \{LF, RF, LR, RR\}$:

$$S_{Hx} = p_{Hx1} \quad \text{(βλ. [3], PAC2002, εξ. 27)}$$

$$S_{Vx,i} = F_{z,i}\, p_{Vx1} \quad \text{(βλ. [3], PAC2002, εξ. 28)}$$

$$\kappa_i = -s_i \quad \text{(βλ. [1])}$$

$$\kappa_{x,i} = \kappa_i + S_{Hx} \quad \text{(βλ. [3], PAC2002, εξ. 19)}$$

$$\mu_{x,i} = p_{Dx1}\left(1 - p_{Dx3}\gamma_i^2\right) \quad \text{(βλ. [3], PAC2002, εξ. 23)}$$

$$C_x = p_{Cx1} \quad \text{(see [8, PAC2002, εξ. 21)}$$

$$D_{x,i} = \mu_x F_{z,i} \quad \text{(βλ. [3], PAC2002, εξ. 22)}$$

$$E_x = p_{Ex1} \quad \text{(βλ. [3], PAC2002, εξ. 24)}$$

$$K_{x,i} = F_{z,i}\, p_{Kx1} \quad \text{(βλ. [3], PAC2002, εξ. 25)}$$

$$B_{x,i} = \frac{K_{x,i}}{C_x D_{x,i}} \quad \text{(βλ. [3], PAC2002, εξ. 26])}$$

$$F_{x0,i} = D_{x,i} \sin\left(C_x \arctan\left(B_{x,i}\kappa_{x,i} - E_x\left(B_{x,i}\kappa_{x,i} - \arctan\left(B_{x,i}\kappa_{x,i}\right)\right)\right)\right) + S_{Vx,i} \quad \text{(βλ. [3], PAC2002, εξ. 18)}$$

Πλευρικές δυνάμεις ελαστικών χρησιμοποιώντας τη Magic Formula για καθαρή ολίσθηση (pure slip) $\forall i \in \{LF, RF, LR, RR\}$:

$$
\begin{aligned}
S_{Hy,i} &= \operatorname{sgn}(\gamma_i)\,(p_{Hy1} + p_{Hy3}\operatorname{abs}(\gamma_i)) && \text{(βλ. [3], PAC2002, εξ. 40)}\\
S_{Vy,i} &= \operatorname{sgn}(\gamma_i)\,F_{z,i}\,(p_{Vy1} + p_{Vy3}\operatorname{abs}(\gamma_i)) && \text{(βλ. [3], PAC2002, εξ. 41)}\\
\alpha_{y,i} &= \alpha_i + S_{Hy,i} && \text{(βλ. [3], PAC2002, εξ. 31)}\\
\mu_{y,i} &= p_{Dy1}\left(1 - p_{Dy3}\gamma_i^2\right) && \text{(βλ. [3], PAC2002, εξ. 35)}\\
C_y &= p_{Cy1} && \text{(see [8, PAC2002, εξ. 33)}\\
D_{y,i} &= \mu_{y,i}F_{z,i} && \text{(βλ. [3], PAC2002, εξ. 34)}\\
E_y &= p_{Ey1} && \text{(βλ. [3], PAC2002, εξ. 36)}\\
K_{y,i} &= F_{z,i}\,p_{Ky1} \quad (\text{ simplified } K_{y0} \text{ to } p_{Ky1}F_{z,i}) && \text{(βλ. [3], PAC2002, εξ. 38)}\\
B_{y,i} &= \frac{K_{y,i}}{C_y D_{y,i}} && \text{(βλ. [3], PAC2002, εξ. 39)}\\
F_{y0,i} &= D_{y,i}\sin(C_y \arctan(B_{y,i}\alpha_{y,i} - E_y(B_{y,i}\alpha_{y,i}\\
&\quad - \arctan(B_{y,i}\alpha_{y,i})))) + S_{Vy,i} && \text{(βλ. [3], PAC2002, εξ. 30) (1)}
\end{aligned}
$$

Διαμήκεις δυνάμεις ελαστικών για συνδυασμένη ολίσθηση (combined slip) $\forall i \in \{LF, RF, LR, RR\}$:

$$
\begin{aligned}
S_{Hx\alpha} &= r_{Hx1}\\
\alpha_{s,i} &= \alpha_i + S_{Hx\alpha}\\
B_{x\alpha,i} &= r_{Bx1}\cos(\arctan(r_{Bx2}\kappa_i))\\
C_{x\alpha} &= r_{Cx1}\\
E_{x\alpha} &= r_{Ex1}\\
D_{x\alpha,i} &= F_{x0,i}/\cos(C_{x\alpha}\arctan(B_{x\alpha,i}S_{Hx\alpha} - E_{x\alpha}(B_{x\alpha,i}S_{Hx\alpha} - \arctan(B_{x\alpha,i}S_{Hx\alpha}))))\\
F_{x,i} &= D_{x\alpha,i}\cos(C_{x\alpha}\arctan(B_{x\alpha,i}\alpha_{s,i} - E_{x\alpha}(B_{x\alpha}\alpha_{s,i} - \arctan(B_{x\alpha,i}\alpha_{s,i}))))
\end{aligned}
$$

Για τις παραπάνω εξισώσεις βλ. [3], PAC2002, εξ. 65, 60, 61, 62, 64, 63 και 59 αντίστοιχα.

Πλευρικές δυνάμεις ελαστικών για συνδυασμένη ολίσθηση (combined slip) $\forall i \in \{LF, RF, LR, RR\}$:

$$
\begin{aligned}
S_{Hy\kappa} &= r_{Hy1} \\
\kappa_{s,i} &= \kappa_i + S_{Hy\kappa} \\
B_{y\kappa,i} &= r_{By1} \cos\left(\arctan\left(r_{By2}(\alpha_i - r_{By3})\right)\right) \\
C_{y\kappa} &= r_{Cy1} \\
E_{y\kappa} &= r_{Ey1} \\
D_{y\kappa} &= \frac{F_{y0,i}}{\cos(C_{y\kappa} \arctan(B_{y\kappa,i} S_{Hy\kappa} - E_{y\kappa}(B_{y\kappa} S_{Hy\kappa} - \arctan(B_{y\kappa,i} S_{Hy\kappa}))))} \\
D_{vy\kappa,i} &= \mu_{y,i} F_{z,i}(r_{Vy1} + r_{Vy3}\gamma_i)\cos(\arctan(r_{Vy4}\alpha_i)) \\
S_{vy\kappa,i} &= D_{vy\kappa,i} \sin(r_{Vy5} \arctan(r_{Vy6}\kappa_i) \\
F_{y,i} &= D_{y\kappa} \cos(C_{y\kappa} \arctan(B_{y\kappa,i}\kappa_{s,i} - E_{y\kappa}(B_{y\kappa,i}\kappa_{s,i} - \arctan(B_{y\kappa,i}\kappa_{s,i})))) + S_{Vy\kappa}
\end{aligned}
$$

Για τις παραπάνω εξισώσεις βλ. [3], PAC2002, εξ. 74, 69, 70, 71, 73, 72, 76, 75 και 68 αντίστοιχα.

Η υλοποίηση του μοντέλου ελαστικών σε Python φαίνεται στο Παράρτημα Α, στο κομμάτι κώδικα Α.1.

2.4 Δυναμική οχήματος

Στο CommonRoad θέτονται αρχικά περιορισμοί όσον αφορά το σύστημα διεύθυνσης και την επιτάχυνση που μπορεί να λάβει ως είσοδο ο προσομοιωτής. Προκειμένου να διατυπωθούν οι περιορισμοί, εισάγονται οι έννοιες γωνία διεύθυνσης δ, ταχύτητα αύξησης της γωνίας διεύθυνσης v_δ, ταχύτητα του οχήματος v, και η παράμετρος v_S που περιγράφει την ταχύτητα πάνω από την οποία η ισχύς του κινητήρα δεν είναι επαρκής να προκαλέσει ολίσθηση τροχών. Συμβολίζουμε με $_{\min}$ την ελάχιστη δυνατή τιμή, με $_{\max}$ τη μέγιστη δυνατή τιμή, με $_{\text{lat}}$ την τιμή μιας μεταβλητής στην πλάγια διεύθυνση και με $_{\text{long}}$ την τιμή μιας μεταβλητής στη διαμήκη διεύθυνση.

Οι περιορισμοί φαίνονται παρακάτω:

$$v_\delta \in [v_{\delta_{\min}}, v_{\delta_{\max}}], \delta \in [\delta_{\min}, \delta_{\max}], v \in [v_{\min}, v_{\max}]$$

Λαμβάνοντας υπόψη ότι η ισχύς του κινητήρα και η ισχύς πέδησης είναι περιορισμένες, προκύπτει η ακόλουθη συνθήκη, όπως περιγράφεται λεπτομερώς στο [5], Εν. III.B:

$$a_{\text{long}} \in [a_{\min}, a_{\max}(v)], a_{\max}(v) = \begin{cases} a_{\max}\frac{v_S}{v} & \text{για } v > v_S, \\ a_{\max} & \text{αλλού.} \end{cases}$$

Τέλος, θεωρείται ο κύκλος τριβής (γνωστός και ως κύκλος του Kamm) που περιορίζει την απόλυτη επιτάχυνση:

$$\sqrt{a_{\text{long}}^2 + (v\dot{\Psi})^2} \leq a_{\max} \ (a_{\text{lat}} = v\dot{\Psi})$$

Οι περιορισμοί στη γωνία διεύθυνσης, την ταχύτητα και την επιτάχυνση μπορούν να ληφθούν άμεσα υπόψη με την εισαγωγή μιας επιθυμητής ταχύτητας αύξησης γωνίας διεύθυνσης $v_{\delta,\text{d}}$ και μιας επιθυμητής επιτάχυνσης $a_{\text{long,d}}$ και επιλέγοντας:

$$v_\delta = f_{\text{steer}}(\delta, v_{\delta,\text{d}}) = \begin{cases} 0 & \text{για } (\delta \leq \delta_{\min} \wedge v_{\delta,\text{d}} \leq 0) \vee (\delta \geq \delta_{\max} \wedge v_{\delta,\text{d}} \geq 0) \quad (C1), \\ v_{\delta_{\min}} & \text{για } \neg C1 \wedge v_{\delta,\text{d}} \leq v_{\delta_{\min}}, \\ v_{\delta_{\max}} & \text{για } \neg C1 \wedge v_{\delta,\text{d}} \geq v_{\delta_{\max}}, \\ v_{\delta,\text{d}} & \text{αλλού.} \end{cases}$$

Η υλοποίηση των περιορισμών της ταχύτητας αύξησης της γωνίας διεύθυνσης σε Python φαίνεται στο Παράρτημα Α, στο κομμάτι κώδικα Α.2.

$$a_{\text{long}} = f_{\text{acc}}(v, a_{\text{long,d}}) = \begin{cases} 0 & \text{για } (v \leq v_{\min} \wedge a_{\text{long,d}} \leq 0) \vee (v \geq v_{\max} \wedge a_{\text{long,d}} \geq 0)(C2), \\ a_{\min} & \text{για } \neg C2 \wedge a_{\text{long,d}} \leq a_{\min}, \\ a_{\max}(v) & \text{για } \neg C2 \wedge a_{\text{long,d}} \geq a_{\max}(v), \\ a_{\text{long,d}} & \text{αλλού.} \end{cases}$$

Η υλοποίηση των περιορισμών της επιτάχυνσης σε Python φαίνεται στο Παράρτημα Α, στο κομμάτι κώδικα Α.3.

Με βάση τις βοηθητικές μεταβλητές από την Ενότητα 2.2, τις δυνάμεις των ελαστικών από την Ενότητα 2.3 και τους περιορισμούς του συστήματος διεύθυνσης, υπολογίζεται το δεξιό μέρος της δυναμικής του οχήματος, δηλαδή το $\dot{x} = f(x, u)$:

Εξισώσεις δυναμικής κοινές με το μοντέλο Single-Track

Περιγραφή του μοντέλου Single-Track μπορεί να βρεθεί στο [1], σελ. 8-11.

$$\dot{x}_1 = v_{CG} \cos(\beta + x_5) \tag{2.2}$$
$$\dot{x}_2 = v_{CG} \sin(\beta + x_5) \tag{2.3}$$
$$\dot{x}_3 = f_{\text{steer}}(x_3, u_1) \tag{2.4}$$
$$\dot{x}_4 = \frac{1}{m} \sum X + x_6 x_{11} \quad (\text{από [2], εξ. A1}) \tag{2.5}$$
$$\dot{x}_5 = x_6 \tag{2.6}$$
$$\dot{x}_6 = \frac{1}{I_z - \frac{I_{xz,s}^2}{I_{\phi,s}}} \left(\sum N + \frac{I_{xz,s}}{I_{\phi,s}} \sum L_s \right) \quad (\text{δείτε παρακάτω}) \tag{2.7}$$

Παραγώγιση του $\dot{x_6}$:

$$I_z \dot{x_6} - I_{xz,s} \dot{x_8} = \sum N \quad (\text{από [2], εξ. A2}) \tag{2.8}$$
$$I_{\phi,s} \dot{x_8} - I_{xz,s} \dot{x_6} = \sum L_s \quad (\text{από [2], εξ. A4}]) \tag{2.9}$$

Πολλαπλασιάζοντας την εξίσωση (2.9) με $\frac{I_{xz,s}}{I_{\phi,s}}$ και προσθέτοντας το αποτέλεσμα στην εξίσωση (2.8) προκύπτει:

$$\left(I_z - \frac{I_{xz,s}^2}{I_{\phi,s}} \right) \dot{x_6} = \sum N + \frac{I_{xz,s}}{I_{\phi,s}} \sum L_s$$

Υπολειπόμενες εξισώσεις δυναμικής αναρτώμενης μάζας

$$\dot{x}_7 = x_8 \tag{2.10}$$

$$\dot{x}_8 = \frac{1}{\left(I_{\phi,s} - \frac{I_{xz,s}^2}{I_z}\right)} \left(\frac{I_{xz,s}}{I_z} \sum N + \sum L\right) \quad \text{(δείτε παρακάτω)} \tag{2.11}$$

$$\dot{x}_9 = x_{10} \tag{2.12}$$

$$\dot{x}_{10} = \frac{\sum Ms}{I_{y,s}} \quad \text{(από [2], εξ. A6)} \tag{2.13}$$

$$\dot{x}_{11} = \frac{1}{m_s} \sum Y_s - x_6 x_4 \quad \text{(δείτε παρακάτω)} \tag{2.14}$$

$$\dot{x}_{12} = x_{13} \tag{2.15}$$

$$\dot{x}_{13} = g - \frac{1}{m_s} \sum Z_s \quad \text{(από [2], εξ. A5)} \tag{2.16}$$

Παραγώγιση του $\dot{x_8}$:
Πολλαπλασιάζοντας την εξίσωση (2.8) με $\frac{I_{xz,s}}{I_z}$ και προσθέτοντας το αποτέλεσμα στην εξίσωση (2.9) προκύπτει:

$$\left(I_{\phi,s} - \frac{I_{xz,s}^2}{I_z}\right) \dot{x_8} = \frac{I_{xz,s}}{I_z} \sum N + \sum L_s$$

Παραγώγιση του $\dot{x_{11}}$:
Χρησιμοποιώντας το $a_y = \dot{x_{11}} + x_6 x_4$ από [2], εξ. A46 και εισάγοντας το στην [2], εξ. A3 προκύπτει:

$$m_s \left(\dot{x_{11}} + x_6 x_4\right) = \sum Y_s$$

Δυναμική εμπρόσθιας μη αναρτώμενης μάζας

$$\dot{x}_{14} = x_{15} \tag{2.17}$$

$$\dot{x}_{15} = \frac{\sum L_{uf}}{I_{u,f}} \quad (\text{από } [2], \text{ εξ. A17}) \tag{2.18}$$

$$\dot{x}_{16} = \frac{\sum Y_{uf}}{m_{u,f}} - x_6 x_4 \quad (\text{από την παραγώγιση του } \dot{x_{11}} \text{ και } [2], \text{ εξ. A19}) \tag{2.19}$$

$$\dot{x}_{17} = x_{18} \tag{2.20}$$

$$\dot{x}_{18} = g - \frac{\sum Z_{uf}}{m_{u,f}} \quad (\text{από } [2], \text{ εξ. A18}) \tag{2.21}$$

Δυναμική οπίσθιας μη αναρτώμενης μάζας

$$\dot{x}_{19} = x_{20} \tag{2.22}$$

$$\dot{x}_{20} = \frac{\sum L_{ur}}{I_{u,r}} \quad (\text{από } [2], \text{ εξ. A17}) \tag{2.23}$$

$$\dot{x}_{21} = \frac{\sum Y_{ur}}{m_{u,r}} - x_6 x_4 \quad (\text{από την παραγώγιση του } \dot{x_{11}} \text{ και } [2], \text{ εξ. A19}) \tag{2.24}$$

$$\dot{x}_{22} = x_{23} \tag{2.25}$$

$$\dot{x}_{23} = g - \frac{\sum Z_{ur}}{m_{u,r}} \quad (\text{από } [2], \text{ εξ. A18}) \tag{2.26}$$

Μετατροπή της επιτάχυνσης του διανύσματος εισόδου σε ροπή φρένου και κινητήρα

Αυτή είναι μια προσθήκη στο [2], Παράρτημα A, η οποία δημιουργεί ροπή κινητήρα όταν η επιτάχυνση που επιλέγεται στο διάνυσμα εισόδου είναι θετική και ροπή φρένου όταν είναι αρνητική. Ωστόσο, λαμβάνονται υπόψη και οι περιορισμοί που αφορούν τη μέγιστη ταχύτητα και τη μέγιστη ισχύ του κινητήρα (βλ. Παράρτημα A, A.5) χρησιμοποιώντας την $f_{\text{acc}}(x_4, u_2)$:

$$u_2 := f_{\text{acc}}(x_4, u_2) \tag{2.27}$$

$$T_B = \begin{cases} 0, & \text{για } u_2 > 0 \\ m \cdot R_w \cdot u_2, & \text{διαφορετικά} \end{cases} \tag{2.28}$$

$$T_E = \begin{cases} m \cdot R_w \cdot u_2, & \text{για } u_2 > 0 \\ 0, & \text{διαφορετικά} \end{cases} \tag{2.29}$$

Δυναμική των τροχών

Η ροπή φρένου T_B, όπως και στο [2], εξ. A55, μοιράζεται μεταξύ του εμπρόσθιου και του οπίσθιου άξονα σύμφωνα με την παράμετρο $T_{s,b}$ (torque split, brake, βλ. A.5, γραμμή 116). Ομοίως, η ροπή του κινητήρα T_E μοιράζεται μεταξύ του εμπρόσθιου και του οπίσθιου άξονα σύμφωνα με την παράμετρο $T_{s,e}$ (torque split, engine, βλ. A.5, γραμμή 117):

$$\dot{x}_{24} = \frac{1}{I_{y,w}} \left(-R_w \cdot F_{x,LF} + \frac{1}{2} T_{s,b} T_B + \frac{1}{2} T_{s,e} T_E \right) \tag{2.30}$$

$$\dot{x}_{25} = \frac{1}{I_{y,w}} \left(-R_w \cdot F_{x,RF} + \frac{1}{2} T_{s,b} T_B + \frac{1}{2} T_{s,e} T_E \right) \tag{2.31}$$

$$\dot{x}_{26} = \frac{1}{I_{y,w}} \left(-R_w \cdot F_{x,LR} + \frac{1}{2} (1 - T_{s,b}) T_B + \frac{1}{2} (1 - T_{s,e}) T_E \right) \tag{2.32}$$

$$\dot{x}_{27} = \frac{1}{I_{y,w}} \left(-R_w \cdot F_{x,RR} + \frac{1}{2} (1 - T_{s,b}) T_B + \frac{1}{2} (1 - T_{s,e}) T_E \right) \tag{2.33}$$

Συνεπώς, όταν $T_{s,e} : 1$, το αυτοκίνητο είναι προσθιοκίνητο, όταν $T_{s,e} : 0$, το αυτοκίνητο είναι πισωκίνητο, ενώ όταν $T_{s,e} : 0.5$, το αυτοκίνητο είναι τετρακίνητο.

Απαγόρευση αρνητικής κύλισης τροχών

Πρόκειται για μια προσθήκη στο [2], Παράρτημα Α, που απαγορεύει την αρνητική κύλιση των τροχών. Όταν χρησιμοποιείται ροπή φρένου, οι τροχοί παραμένουν ακίνητοι σε περίπτωση που σταματήσουν να κινούνται, αντί να επιταχύνουν προς τα πίσω:

$$\forall i \in [24, \ldots, 27] : \dot{xi} = 0 \text{ για } xi < 0,\ xi := 0 \text{ για } xi < 0$$

Εξισώσεις ευέλικτων αρθρώσεων

$$\dot{x}_{28} = \Delta\dot{y}F \tag{2.34}$$
$$\dot{x}_{29} = \Delta\dot{y}R \tag{2.35}$$

Μικρές ταχύτητες

Το μοντέλο που θα χρησιμοποιηθεί γίνεται μοναδιαίο για πολύ μικρές ταχύτητες. Συνεπώς, στην περίπτωση αυτή χρησιμοποιείται το μοντέλο Single-Track για την εξαγωγή των μεταβλητών $\dot{x}_1$-$\dot{x}_6$, όπως ακριβώς παρουσιάζεται στο [1], σελ. 10. Επιπλέον, όλες οι γωνίες ολίσθησης ορίζονται σε μηδέν:

$$s_{LF} = s_{RF} = s_{LR} = s_{RR} = \alpha_{LF} = \alpha_{RF} = \alpha_{LR} = \alpha_{RR} = 0$$

Η υλοποίηση της δυναμικής του οχήματος σε Python φαίνεται στο Παράρτημα Α, στο κομμάτι κώδικα Α.4.

2.5 Παράμετροι

Το μοντέλο που περιγράφηκε απαιτεί συνολικά 69 παραμέτρους, εκ των οποίων 37 προσδιορίζουν το όχημα και 32 τα ελαστικά. Οι παράμετροι του οχήματος βρίσκονται στο Παράρτημα Α, Α.5, ενώ οι παράμετροι του μοντέλου ελαστικών στο Α.6.

2.6 Δοκιμαστική προσομοίωση μοντέλου

Ακολουθεί η πρώτη δοκιμαστική προσομοίωση του μοντέλου αυτοκινήτου, επάνω στο οποίο θα αναπτυχθεί αλγόριθμος Torque Vectoring (TV).

Διαμόρφωση αρχικών καταστάσεων του μοντέλου

Λόγω της πολυπλοκότητας του μοντέλου Multi-Body, το CommonRoad [1] προτείνει αρχικοποίηση καταστάσεων χρησιμοποιώντας τις ακόλουθες βοηθητικές μεταβλητές:

- $\omega_0 = \frac{v_{x,0}}{R}$ (χωρίς ολίσθηση τροχών, R: ωφέλιμη ακτίνα ελαστικού),
- $v_{x,0} = \cos(-\beta_0)v_0$ (ταχύτητα στη διαμήκη διεύθυνση, υπολογιζόμενη από τη γωνία ολίσθησης β),
- $v_{y,0} = \sin(-\beta_0)v_0$ (ταχύτητα στην πλάγια διεύθυνση, υπολογιζόμενη από τη γωνία ολίσθησης β),
- $v_{yf,0} = v_{y,0} + lf\dot{\Psi}_0$ (ταχύτητα στην πλάγια διεύθυνση στον εμπρόσθιο άξονα, υπολογιζόμενη από την ταχύτητα στο κέντρο βάρους και από το ρυθμό εκτροπής),
- $v_{yr,0} = v_{y,0} - lr\dot{\Psi}_0$ (ταχύτητα στην πλάγια διεύθυνση στον οπίσθιο άξονα, υπολογιζόμενη από την ταχύτητα στο κέντρο βάρους και από το ρυθμό εκτροπής),
- $z_{i,0} = \frac{F_{zi,0}}{2K_{zt}}$ (ύψος από το έδαφος έτσι ώστε τα ελατήρια να υποστηρίζουν το βάρος), όπου $i \in \{f, r\}$.

Πίνακας 2.1: Αρχικές συνθήκες μοντέλου

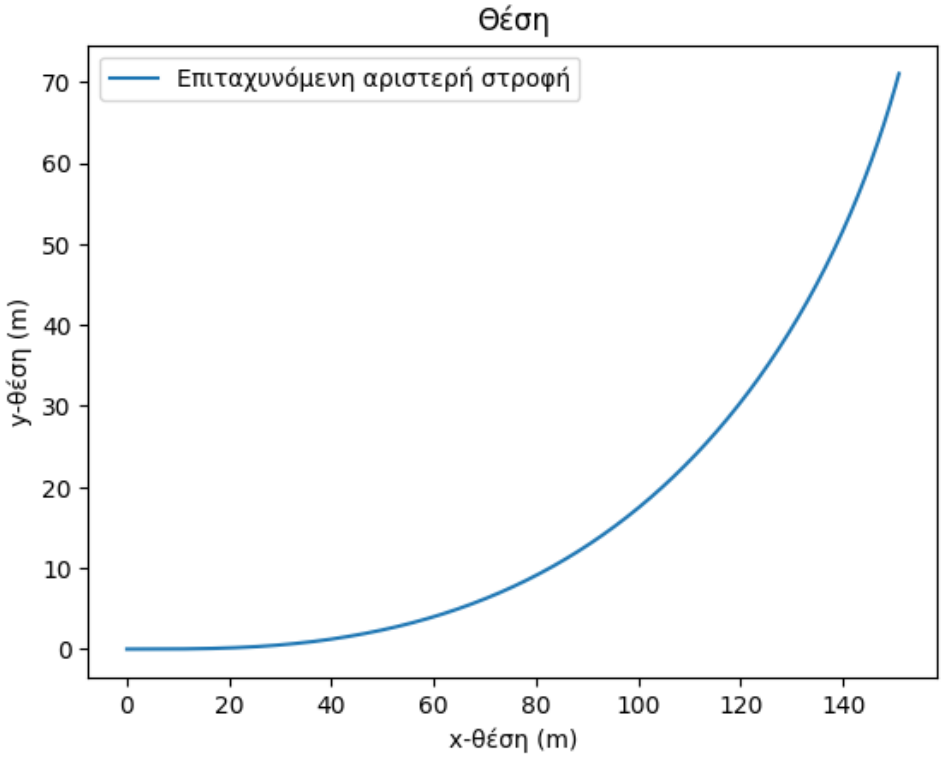
Αναρτ. μάζα ονομασία	**συμβ.**	**τιμή**	**Μη αναρτ. μάζα ονομασία**	**συμβ.**	**τιμή**	**Άλλα ονομασία**	**συμβ.**	**τιμή**
yaw ang.	$x5,0$	Ψ_0	roll ang. (f)	$x14,0$	0	wheel speed (LF)	$x24,0$	ω_0
yaw rate	$x6,0$	$\dot{\Psi}_0$	roll rate (f)	$x15,0$	0	wheel speed (RF)	$x25,0$	ω_0
roll angle	$x7,0$	0	roll ang. (r)	$x19,0$	0	wheel speed (LR)	$x26,0$	ω_0
roll rate	$x8,0$	0	roll rate (r)	$x20,0$	0	wheel speed (RR)	$x27,0$	ω_0
pitch ang.	$x9,0$	0	y-vel. (f)	$x16,0$	$vyf,0$	pin joint diff. (f)	$x28,0$	0
pitch rate	$x10,0$	0	y-vel. (r)	$x21,0$	$vyr,0$	pin joint diff. (r)	$x29,0$	0
x-velocity	$x4,0$	$vx,0$	z-pos. (f)	$x17,0$	$zf,0$	x-position	$x1,0$	$sx,0$
y-velocity	$x11,0$	$vy,0$	z-vel. (f)	$x18,0$	0	y-position	$x2,0$	$sy,0$
z-position	$x12,0$	0	z-pos. (r)	$x22,0$	$zr,0$	steering angle	$x3,0$	δ_0
z-velocity	$x13,0$	0	z-vel. (r)	$x23,0$	0			

Προσομοίωση αριστερής στροφής

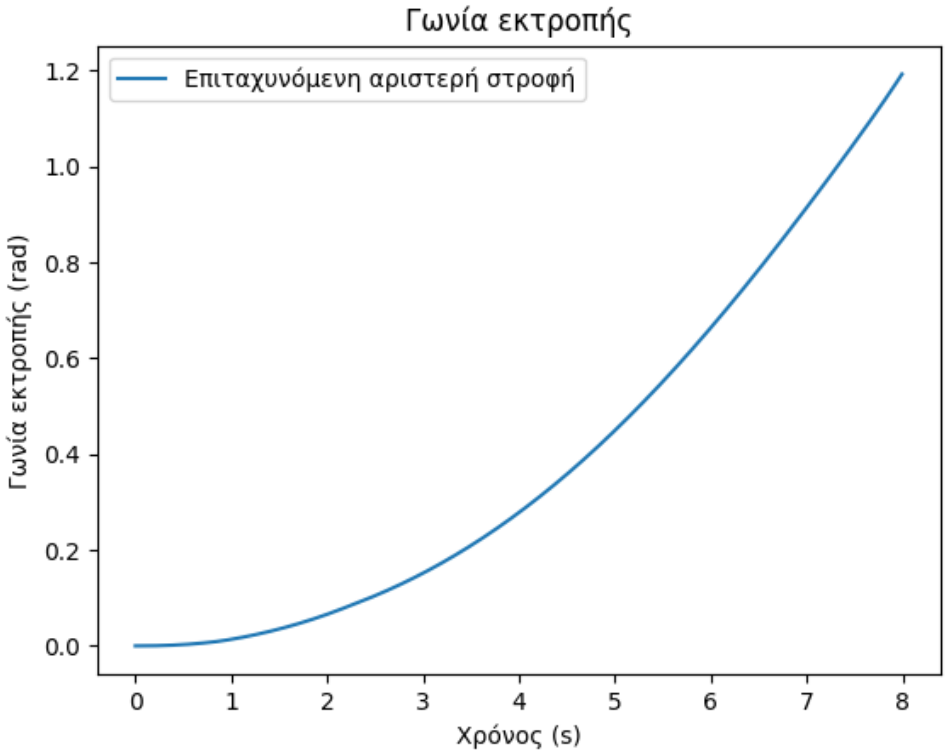
Λόγω του ότι το αυτοκίνητο που θα χρησιμοποιηθεί στις προσομοιώσεις (Bmw 320i) είναι εξ' ορισμού πισωκίνητο (RWD), εκτελείται μετατροπή στο αρχείο παραμέτρων (A.5), προκειμένου να μετατραπεί σε τετρακίνητο (4WD). Με τον τρόπο αυτό δίνεται δυνατότητα κατανομής της ροπής μεταξύ του εμπρόσθιου-οπίσθιου άξονα (Front-rear TV) επιπλέον της κατανομής μεταξύ της αριστερής-δεξιάς πλευράς (Left-right TV) του οχήματος.

Η μετατροπή στο αρχείο παραμέτρων γίνεται στο A.5, γραμμή 117, αφού η παράμετρος $T_{s,e}$ είναι υπεύθυνη για το διαμοιρασμό της ροπής του κινητήρα μεταξύ του εμπρόσθιου και του οπίσθιου άξονα (βλ. Ενότητα 2.4). Η παράμετρος τίθεται σε $T_{s,e}$: 0.5.

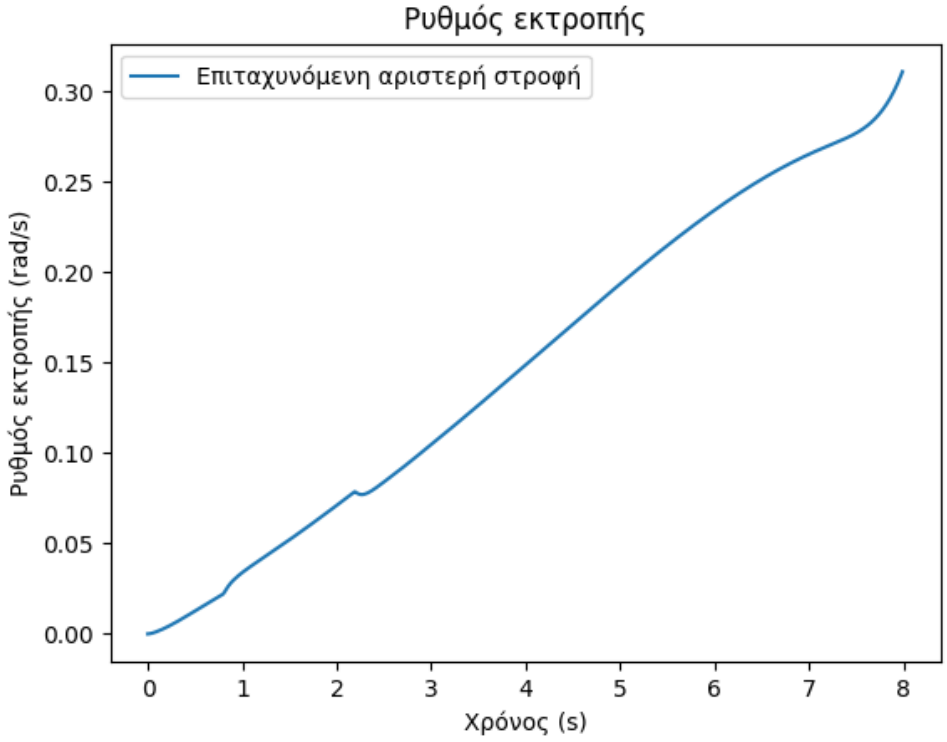
Η προσομοίωση αριστερής στροφής με αρχική ταχύτητα 15 m/s, ρυθμό αύξησης γωνίας διεύθυνσης 0.005 rad/s και επιτάχυνση 0.2g φαίνεται παρακάτω στα Σχ. 2.2, 2.3, 2.4. Οι τιμές $v_\delta = 0.005 rad/s$ και $a_{\text{long}} = 0.2g$ εισάγονται στο διάνυσμα εισόδου (βλ. Ενότητα 2.2) και η αρχική ταχύτητα ορίζεται στις αρχικές συνθήκες. Το απόσπασμα κώδικα για την προσομοίωση παρατίθεται στο Παράρτημα Α, A.7.



Σχήμα 2.2: Τροχιά αριστερής στροφής με $v_0 = 15\,m/s$, $v_\delta = 0.005\,rad/s$ και $a_{\text{long}} = 0.2g$



Σχήμα 2.3: Γωνία εκτροπής αριστερής στροφής με $v_0 = 15\,m/s$, $v_\delta = 0.005\,rad/s$ και $a_{\text{long}} = 0.2g$



Σχήμα 2.4: Ρυθμός εκτροπής αριστερής στροφής με $v_0 = 15\,m/s$, $v_\delta = 0.005\,rad/s$ και $a_{\text{long}} = 0.2g$

Από το Σχ. 2.2 παρατηρείται ομαλή αριστερή στροφή χωρίς απότομες αλλαγές κατεύθυνσης, αποτέλεσμα αναμενόμενο λόγω του μικρού ρυθμού άυξησης της γωνίας διεύθυνσης που επιβάλλεται. Μελετώντας το ρυθμό εκτροπής του οχήματος στο Σχ. 2.4, το όχημα δεν παρουσιάζει φαινόμενα υπερστροφής ή υποστροφής, αφού αυτά σχετίζονται με απότομες διακυμάνσεις του ρυθμού εκτροπής. Αντιθέτως, σε αυτή την προσομοίωση παρατηρείται σχεδόν γραμμική αύξηση, χωρίς διακυμάνσεις.

Κεφάλαιο 3

Torque Vectoring

Στο κεφάλαιο αυτό πραγματοποιείται εμβάθυνση στην τεχνολογία του Torque Vectoring (TV) και ύστερα υλοποίηση ενός τέτοιου συστήματος σε προγραμματιστική γλώσσα Python.

3.1 Θεωρητικό υπόβαθρο

Στα συμβατικά αυτοκίνητα με τετρακίνηση, είτε όλοι οι τροχοί του οχήματος κινούνται συνεχώς, το οποίο αναφέρεται ως all-wheel-drive αυτοκίνητο (AWD), είτε ένας από τους δύο άξονες είναι πάντοτε συνδεδεμένος με τον κινητήρα, ενώ ο δεύτερος μπορεί να συνδεθεί (χειροκίνητα ή αυτόματα) όταν απαιτείται. Παρόλο που αυτές οι διαμορφώσεις του συστήματος μετάδοσης κίνησης μπορούν να βελτιώσουν την πρόσφυση και τη δυναμική οδήγησης σε αρκετές οδικές συνθήκες, τα τελευταία χρόνια έχουν εισαχθεί συστήματα ενεργών διαφορικών, τα οποία είναι σε θέση να διανέμουν την ισχύ του κινητήρα στους εμπρόσθιους και τους οπίσθιους άξονες, καθώς και στους αριστερούς και δεξιούς τροχούς κάθε άξονα, ανάλογα με τους ελιγμούς κατά τη διάρκεια της οδήγησης αλλά και τις οδικές συνθήκες.

Ο όρος "κατανομή ροπής" ("Torque Vectoring") είναι όρος που εισήγαγε η εταιρεία Ricardo [6] για να περιγράψει έναν τρόπο μεταβολής της διανομής της ισχύος του κινητήρα μεταξύ των δύο πλευρών ενός διαφορικού. Αρχικά, η κατανομή ροπής πραγματοποιούνταν με βάση τις σχετικές ταχύτητες του αριστερού και του δεξιού άξονα. Στη συνέχεια, εισήχθησαν ενεργά συστήματα Torque Vectoring, παρουσιάζοντας δυνατότητες κατανομής ροπής κατά απαίτηση. Πιο συγκεκριμένα, ένα κεντρικό διαφορικό μοιράζει τη διαθέσιμη ισχύ του κινητήρα στα διαφορικά των εμπρόσθιων και οπίσθιων αξόνων. Κάθε διαφορικό του εμπρόσθιου/οπίσθιου άξονα μπορεί να εξοπλιστεί με ένα ενεργό σύστημα κατανομής ροπής, το οποίο επιτρέπει διαμοιρασμό της ροπής και μεταξύ των αριστερών και δεξιών τροχών. Συνεπώς, ένα ενεργό κεντρικό διαφορικό κατανέμει την ροπή ανάλογα με την δυναμική οδήγησης και τις δυνατότητες πρόσφυσης σε κάθε άξονα (εμπρόσθιο-οπίσθιο TV), ενώ ένα ενεργό διαφορικό σε κάθε άξονα κατανέμει τη ροπή σε κάθε τροχό (αριστερό-δεξί TV). Η τακτική αυτή μπορεί να χρησιμοποιηθεί ως σύστημα ελέγχου ευστάθειας. Στην παρούσα Διπλωματική Εργασία γίνεται η υπόθεση ότι το όχημα που χρησιμοποιείται διαθέτει την παραπάνω ακριβώς τεχνολογία. Στόχος είναι η δημιουργία αλγορίθμου κατανομής ροπής, με τη χρήση του οποίου το όχημα θα παρουσιάζει βελτιωμένες επιδόσεις αλλά και αυξημένη ασφάλεια κατά τη διάρκεια ελιγμών. Η αξιολόγηση θα πραγματοποιηθεί συγκρίνοντας τις επιδόσεις του οχήματος με το σύστημα ανενεργό (σταθερή τετρακίνηση) και ύστερα ενεργό.

Υποστροφή

Υποστροφή παρατηρείται όταν ένα όχημα στρίβει λιγότερο από το αναμενόμενο για δεδομένη γωνία στροφής τιμονιού. Πιο συγκεκριμένα, τα εμπρόσθια ελαστικά χάνουν πρόσφυση και γλιστρούν περισσότερο από τα οπίσθια κατά τη διάρκεια της στροφής. Ως αποτέλεσμα, το όχημα τείνει να βγεί εκτός στροφής και ο οδηγός πρέπει να στρίψει περισσότερο το τιμόνι για να το κρατήσει στην επιθυμητή πορεία. Η υποστροφή είναι ευκολότερα διαχειρίσιμη σε σχέση με την υπερστροφή, αλλά μπορεί να οδηγήσει σε μειωμένη απόδοση, ιδιαίτερα στις "κλειστές" στροφές.

Υπερστροφή

Υπερστροφή παρατηρείται όταν ένα όχημα στρίβει περισσότερο από το αναμενόμενο για δεδομένη γωνία στροφής τιμονιού. Σε αυτή την περίπτωση, τα οπίσθια ελαστικά χάνουν πρόσφυση και ολισθαίνουν περισσότερο από τα εμπρόσθια κατά τη διάρκεια της στροφής. Κατά συνέπεια, το οπίσθιο μέρος του οχήματος τείνει να ολισθαίνει εξωτερικά σε σχέση με το εμπρόσθιο μέρος. Το φαινόμενο της υπερστροφής είναι πιο συχνό σε οχήματα με κίνηση στους οπίσθιους τροχούς και συχνά συνδέεται με σπορ και επιθετικό στυλ οδήγησης. Μπορεί να προσφέρει μια πιο δυναμική και συναρπαστική εμπειρία οδήγησης, αλλά απαιτεί δεξιότητα για τον χειρισμό.

Αριστερό - δεξί TV

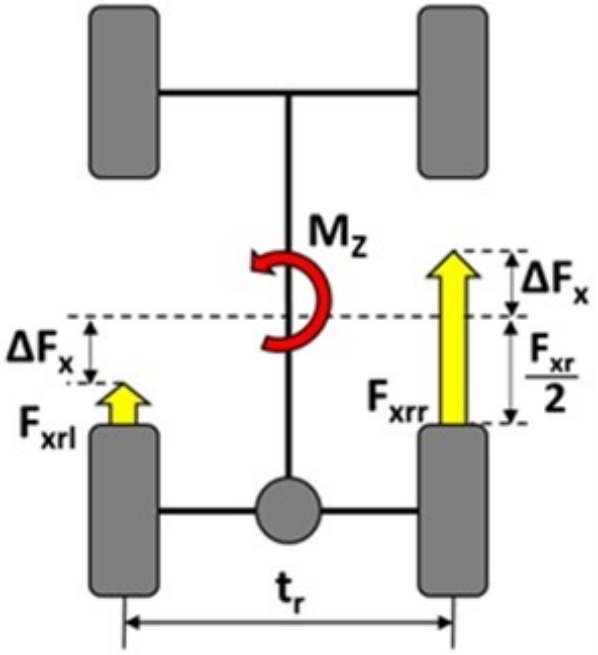
Όσον αφορά το αριστερό-δεξί TV, αύξηση της ροπής στους τροχούς της μίας μεριάς του οχήματος, άρα και αύξηση των διαμήκων δυνάμεων που ασκούνται στο επίπεδο από τα ελαστικά αυτής της μεριάς, μαζί με αντίστοιχη μείωση της ροπής στην άλλη μεριά του οχήματος, προκαλούν μια ροπή στρέψης στο αμάξωμα (βλ. Σχ. 3.1). Εφαρμόζοντας αριστερό-δεξί TV μπορεί να ελαττωθεί τυχόν υποστροφική συμπεριφορά μεταφέροντας μεγαλύτερο μέρος της διαθέσιμης ροπής στους εξωτερικούς τροχούς της στροφής. Η ροπή στρέψης που δημιουργείται βοηθά το αυτοκίνητο να αποκτήσει στενότερη τροχιά και να στραφεί πιο κοντά στην αναμενόμενη πορεία. Αντίστοιχα, μπορεί να ελαττωθεί τυχόν υπερστροφική συμπεριφορά μεταφέροντας μεγαλύτερο μέρος της διαθέσιμης ροπής στους εσωτερικούς τροχούς της στροφής. Το αυτοκίνητο θα αποκτήσει πλατύτερη τροχιά, καθιστώντας την πορεία του κοντύτερα στην επιθυμητή.

Εμπρόσθιο - οπίσθιο TV

Μια περιγραφή του εμπρόσθιου-οπίσθιου TV με βάση τους νόμους της φυσικής έχει ως εξής: Καθώς περισσότερη ροπή μεταφέρεται στον εμπρόσθιο άξονα, η επίδραση των διαμήκων δυνάμεων στους εμπρόσθιους τροχούς αυξάνεται. Λόγω αυτής της επίδρασης, η διαμήκης ολίσθηση του εμπρόσθιου άξονα αυξάνεται, ενώ αυτή του πίσω άξονα μειώνεται. Αυτό οδηγεί επίσης σε μείωση των πλευρικών δυνάμεων που παράγονται από τα εμπρόσθια ελαστικά σε σύγκριση με τα οπίσθια, όπως εξηγείται από τον κύκλο τριβής (friction ellipse). Η μείωση των πλευρικών δυνάμεων προκαλεί φαινόμενο υποστροφής, το οποίο μπορεί να βοηθήσει ένα όχημα που βρίσκεται σε κατάσταση υπερστροφής να αποκτήσει πιο ουδέτερη συμπεριφορά. Αυξάνοντας τη ροπή στα πίσω ελαστικά και λαμβάνοντας υπόψη το ίδιο φαινόμενο, προκαλείται τάση υπερστροφής, γεγονός που μπορεί να βοηθήσει ένα όχημα που

βρίσκεται σε κατάσταση υποστροφής να αποκτήσει πιο κλειστή τροχιά, αφού αυξάνει την πλευρική δύναμη στον εμπρόσθιο άξονα και, συνεπώς, βελτιώνει τη δυνατότητα του οδηγού να κατευθύνει το αυτοκίνητο [7].

Παρόλο που η φιλοσοφία του TV είναι παρόμοια με αυτή των συστημάτων ελέγχου ηλεκτρονικής ευστάθειας (ESC), το TV είναι πιο αποτελεσματικό, ιδίως στη δημιουργία διορθωτικής ροπής στρέψης του οχήματος σε υψηλές ταχύτητες και κατά τη διάρκεια έκτακτων ελιγμών κοντά στα όρια πρόσφυσης [8].



Σχήμα 3.1: Ασύμμετρη κατανομή ροπής μεταξύ των τροχών του οπίσθιου άξονα
ΔF_x: Αύξηση διαμήκους δύναμης ελαστικού ασκούμενη στο έδαφος
M_z: Διορθωτική ροπή στρέψης αμαξώματος

3.2 Επίδραση κατανομής ροπής

Στην Ενότητα αυτή θα εφαρμοστούν συνοπτικά στο μοντέλο οχήματος (βλ. Κεφάλαιο 2) τακτικές που περιγράφηκαν στην Ενότητα 3.1, προκειμένου να ενισχυθούν με οπτικά αποτελέσματα. Η υλοποίηση θα γίνει χωρίς την επίδραση κάποιου είδους ελέγχου.

Αριστερό - δεξί TV

Προκειμένου να επιτευχθεί ασύμμετρη κατανομή ροπής μεταξύ των τροχών της δεξιάς και της αριστερής πλευράς του μοντέλου οχήματος, επεκτείνεται το διάνυσμα εισόδου (βλ. εξ. 2.1) έτσι ώστε να δέχεται ακόμα μία μεταβλητή, τη διορθωτική ροπή (ΔT) που θα προστίθεται στους τροχούς της μίας πλευράς και ταυτόχρονα θα αφαιρείται από τους τροχούς της απέναντι πλευράς.

Το διάνυσμα εισόδου u αρχικά λάμβανε στην πρώτη θέση του το ρυθμό αύξησης της γωνίας διεύθυνσης και στη δεύτερη θέση του την επιτάχυνση του κέντρου μάζας του αυτοκινήτου. Το νέο διάνυσμα εισόδου που παρέχει τη δυνατότητα ασύμμετρης κατανομής ροπής είναι:

$$u = (u_1, u_2, u_3),\ \text{όπου}\ u_1 = v_\delta,\ u_2 = a_{\text{long}}\ \text{και}\ u_3 = \Delta T \tag{3.1}$$

Η μεταβλητή ΔT προστίθεται στις εξισώσεις αναπαράστασης της ταχύτητας των τροχών της δεξιάς πλευράς (εξ. 2.31, 2.33) και αφαιρείται από τις εξισώσεις αναπαράστασης της ταχύτητας των τροχών της αριστερής πλευράς (εξ. 2.30, 2.32), οπότε η ανανεωμένη δυναμική των τροχών έχει ως εξής:

$$\dot{x}_{24} = \frac{1}{I_{y,w}}\left(-R_w \cdot F_{x,LF} + \frac{1}{2}T_{s,b}T_B + \frac{1}{2}T_{s,e}T_E - T_{s,e}\Delta T\right) \tag{3.2}$$

$$\dot{x}_{25} = \frac{1}{I_{y,w}}\left(-R_w \cdot F_{x,RF} + \frac{1}{2}T_{s,b}T_B + \frac{1}{2}T_{s,e}T_E + T_{s,e}\Delta T\right) \tag{3.3}$$

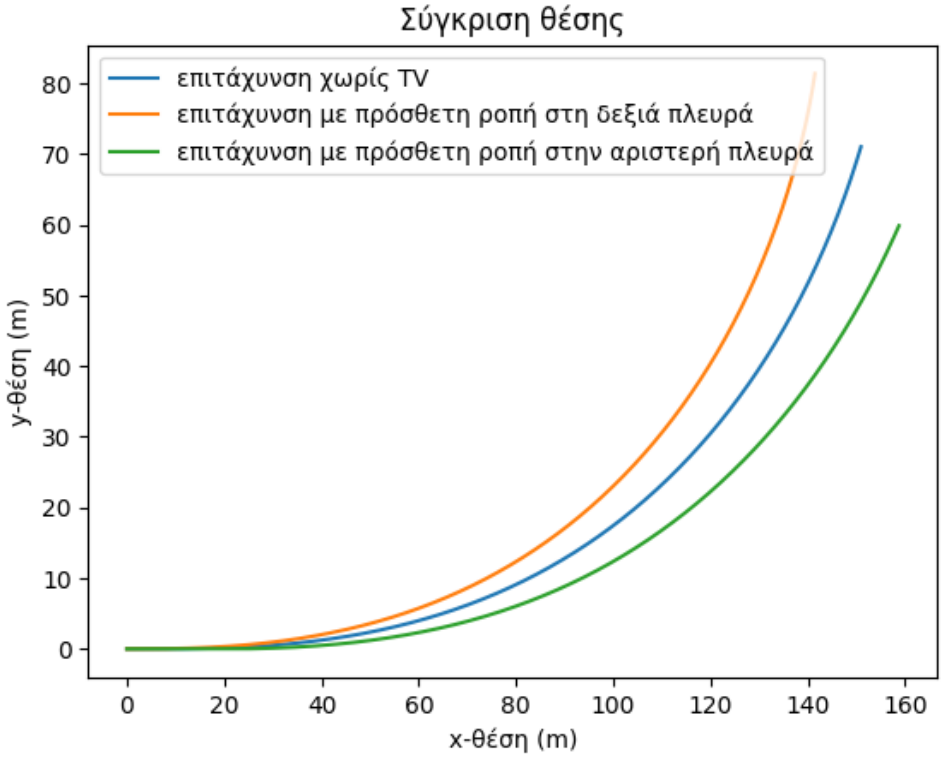
$$\dot{x}_{26} = \frac{1}{I_{y,w}}\left(-R_w \cdot F_{x,LR} + \frac{1}{2}(1-T_{s,b})T_B + \frac{1}{2}(1-T_{s,e})T_E - (1-T_{s,e})\Delta T\right) \tag{3.4}$$

$$\dot{x}_{27} = \frac{1}{I_{y,w}}\left(-R_w \cdot F_{x,RR} + \frac{1}{2}(1-T_{s,b})T_B + \frac{1}{2}(1-T_{s,e})T_E + (1-T_{s,e})\Delta T\right) \tag{3.5}$$

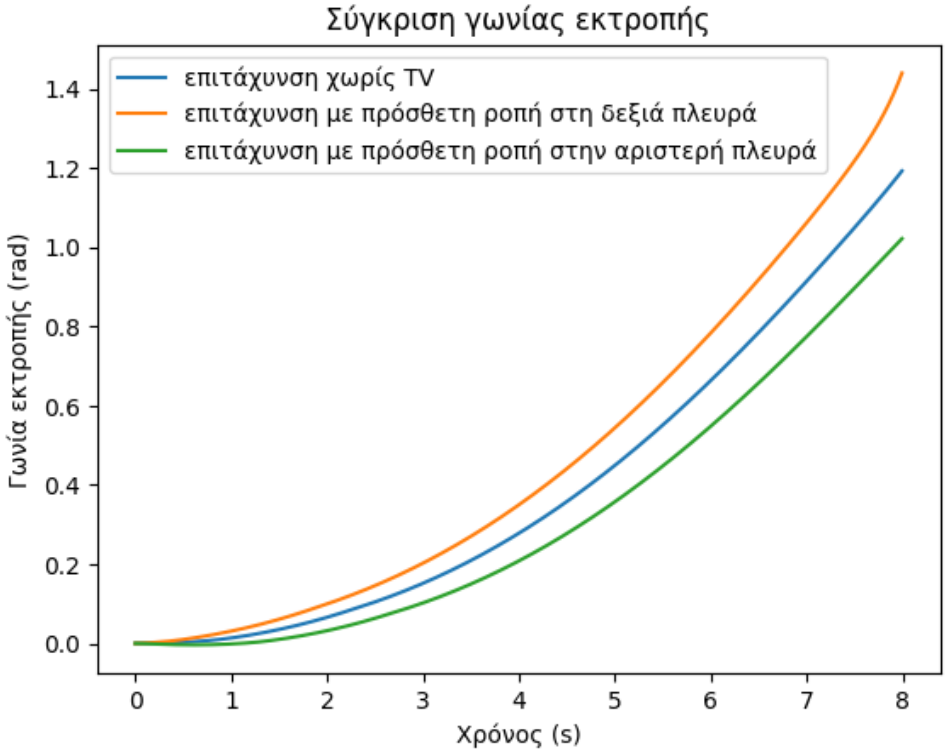
Η ενημέρωση του αρχείου δυναμικής του μοντέλου με τις παραπάνω προσθήκες παρατίθεται στο Παράρτημα Α, Α.8. Η επέκταση του διανύσματος εισόδου γίνεται στη γραμμή 73 του κώδικα και οι εξισώσεις κίνησης των τροχών τροποποιούνται στη γραμμή 330.

Προσομοίωση αριστερής στροφής με κατανομή ροπής

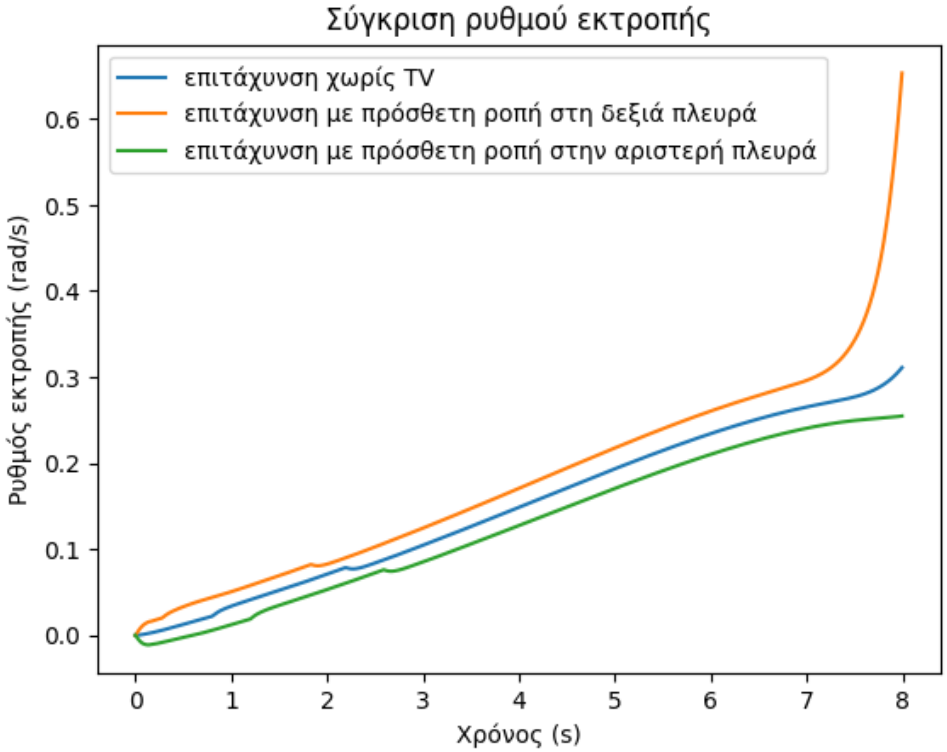
Θα εκτελεσθεί προσομοίωση αριστερής στροφής με αρχική ταχύτητα 15 m/s, ρυθμό αύξησης γωνίας διεύθυνσης 0.005 rad/s και επιτάχυνση 0.2g, όπως και στην Ενότητα 2.6. Ωστόσο, σε αυτή τη δοκιμή θα διανεμεί μεγαλύτερο μέρος της ροπής του κινητήρα στους τροχούς της δεξιάς πλευράς του οχήματος και ύστερα το αντίστροφο. Η επιπλέον διορθωτική ροπή που θα μεταφερθεί σε κάθε πλευρά επιλέγεται $\Delta T = 100\,Nm$. Τα αποτελέσματα της προσομοίωσης φαίνονται παρακάτω στα Σχ. 3.2, 3.3, 3.4. Το απόσπασμα κώδικα για την προσομοίωση παρατίθεται στο Παράρτημα Α, Α.9.



Σχήμα 3.2: Τροχιά αριστερής στροφής με $v_0 = 15\,m/s$, $v_\delta = 0.005\,rad/s$, $a_{\text{long}} = 0.2g$ και κατανομή ροπής $\Delta T = 100\,Nm$



Σχήμα 3.3: Γωνία εκτροπής αριστερής στροφής με $v_0 = 15\,m/s$, $v_\delta = 0.005\,rad/s$, $a_{\text{long}} = 0.2g$ και κατανομή ροπής $\Delta T = 100\,Nm$



Σχήμα 3.4: Ρυθμός εκτροπής αριστερής στροφής με $v_0 = 15\,m/s$, $v_\delta = 0.005\,rad/s$, $a_{\text{long}} = 0.2g$ και κατανομή ροπής $\Delta T = 100\,Nm$

Από την παραπάνω προσομοίωση επαληθεύτηκε η επίδραση του TV στη συμπεριφορά του αυτοκινήτου. Όπως αναφέρθηκε και στην Ενότητα 3.1, η μεταφορά μεγαλύτερου μέρους της διαθέσιμης ροπής στους εξωτερικούς τροχούς της στροφής δημιούργησε ροπή στρέψης που ώθησε το αυτοκίνητο σε στενότερη τροχιά (Σχ. 3.2) και σε υψηλότερο ρυθμό εκτροπής (Σχ. 3.4). Συγκριτικά, η γωνία εκτροπής για $t = 7.5\,s$ παρουσιάζει αύξηση σχεδόν 0.2 rad, όπως φαίνεται στο σχήμα 3.4. Αντιθέτως, η μεταφορά μεγαλύτερου μέρους της διαθέσιμης ροπής στους εσωτερικούς τροχούς της στροφής ώθησε το αυτοκίνητο σε πλατύτερη τροχιά (Σχ. 3.2) και σε χαμηλότερο ρυθμό εκτροπής (Σχ. 3.2). Η γωνία εκτροπής τη χρονική στιγμή $t = 7.5\,s$ μειώθηκε στα 0.9 rad, σε σχέση με το όχημα χωρίς TV, που την ίδια στιγμή παρουσίασε γωνία εκτροπής 1 rad.

Εμπρόσθιο - οπίσθιο TV

Προκειμένου να επιτευχθεί ασύμμετρη κατανομή ροπής και μεταξύ του εμπρόσθιου και του οπίσθιου άξονα του μοντέλου οχήματος, επεκτείνεται ξανά το διάνυσμα εισόδου (βλ. εξ. 3.1) έτσι ώστε να δέχεται ακόμα μία μεταβλητή, την παράμετρο $T_{s,e}$, η οποία είναι υπεύθυνη για αυτή ακριβώς τη διαδικασία (βλ. Ενότητα 2.4). Επομένως, το νέο διάνυσμα εισόδου που παρέχει τη δυνατότητα ασύμμετρης κατανομής της ροπής μεταξύ των τροχών κάθε πλευράς, αλλά και μεταξύ του εμπρόσθιου και οπίσθιου άξονα είναι:

$$u = (u_1, u_2, u_3, u_4),\ \text{όπου}\ u_1 = v_\delta,\ u_2 = a_{\text{long}},\ u_3 = \Delta T\ \text{και}\ u_4 = T_{s,e} \tag{3.6}$$

Η παράμετρος $T_{s,e}$ εκφράζεται πλέον ως είσοδος u_4 στις εξισώσεις αναπαράστασης της ταχύτητας των τροχών, οι οποίες παίρνουν τη μορφή:

$$\dot{x}_{24} = \frac{1}{I_{y,w}}\left(-R_w \cdot F_{x,LF} + \frac{1}{2}T_{s,b}T_B + \frac{1}{2}u_4 T_E - u_4\Delta T\right) \tag{3.7}$$

$$\dot{x}_{25} = \frac{1}{I_{y,w}}\left(-R_w \cdot F_{x,RF} + \frac{1}{2}T_{s,b}T_B + \frac{1}{2}u_4 T_E + u_4\Delta T\right) \tag{3.8}$$

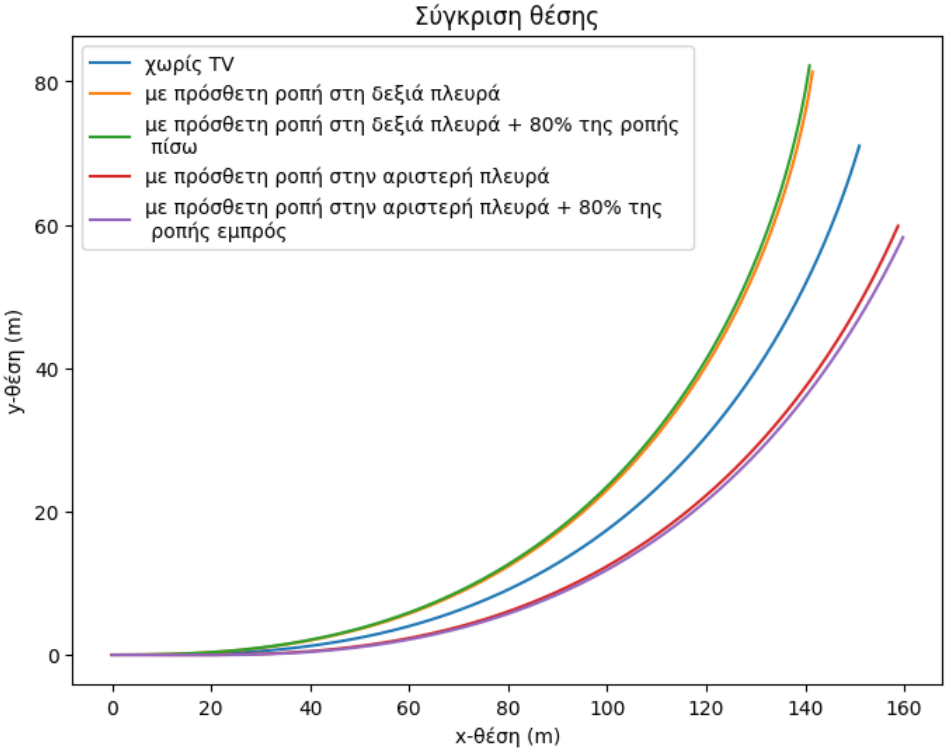
$$\dot{x}_{26} = \frac{1}{I_{y,w}}\left(-R_w \cdot F_{x,LR} + \frac{1}{2}(1-T_{s,b})T_B + \frac{1}{2}(1-u_4)T_E - (1-u_4)\Delta T\right) \tag{3.9}$$

$$\dot{x}_{27} = \frac{1}{I_{y,w}}\left(-R_w \cdot F_{x,RR} + \frac{1}{2}(1-T_{s,b})T_B + \frac{1}{2}(1-u_4)T_E + (1-u_4)\Delta T\right) \tag{3.10}$$

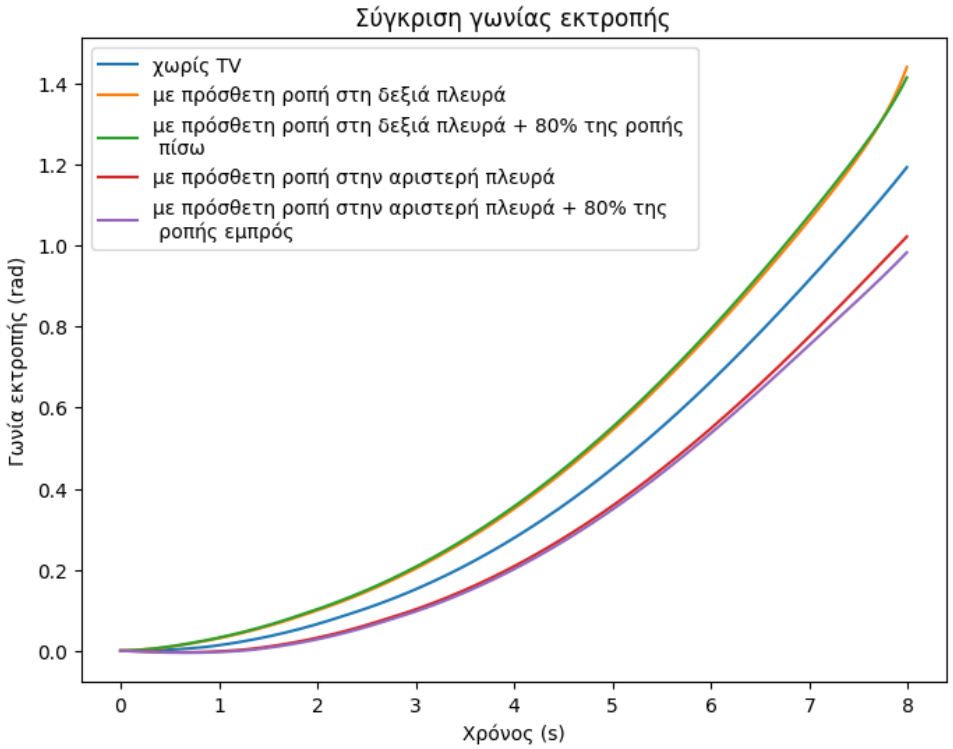
Η ενημέρωση του αρχείου δυναμικής του μοντέλου με τις παραπάνω προσθήκες παρατίθεται στο Παράρτημα Α, Α.10. Η επέκταση του διανύσματος εισόδου γίνεται στη γραμμή 72 του κώδικα και οι εξισώσεις κίνησης των τροχών τροποποιούνται στη γραμμή 330.

Προσομοίωση αριστερής στροφής με κατανομή ροπής δεξιά - αριστερά και εμπρός - πίσω

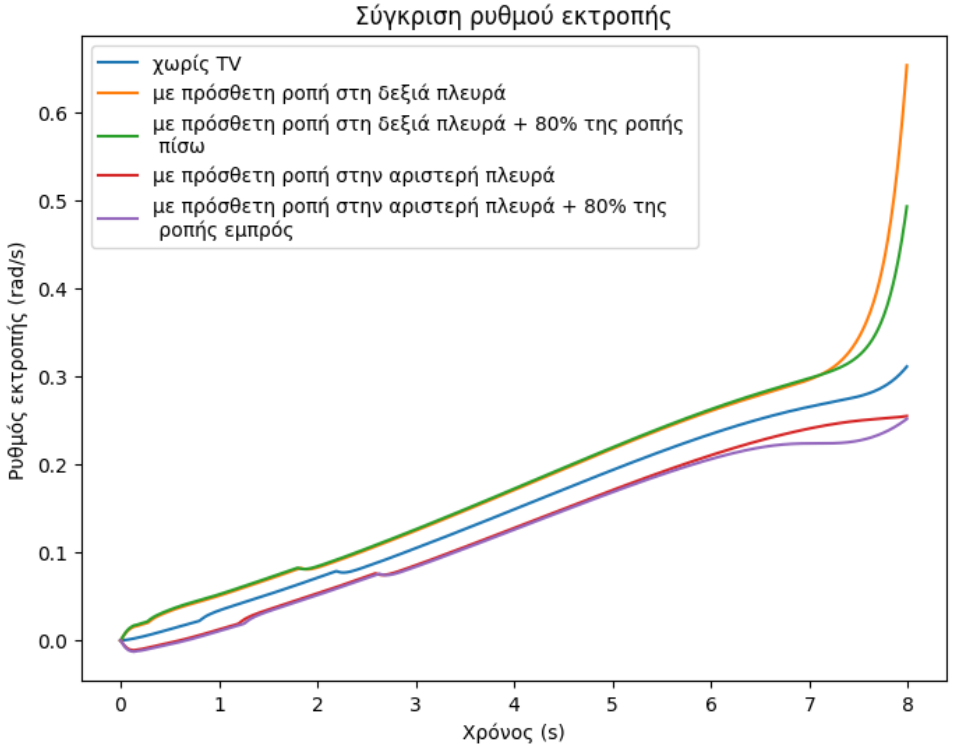
Θα εκτελεσθεί προσομοίωση αριστερής στροφής με αρχική ταχύτητα 15 m/s, ρυθμό αύξησης γωνίας διεύθυνσης 0.005 rad/s και επιτάχυνση 0.2g. Ωστόσο, σε αυτή τη δοκιμή, εκτός από κατανομή ροπής $\Delta T = 100\,Nm$ στους τροχούς κάθε πλευράς του οχήματος, θα επιβληθεί και ασύμμετρη κατανομή ροπής μεταξύ του εμπρόσθιου και του οπίσθιου άξονα. Πιο συγκεκριμένα, όταν εκτελείται μεταφορά ροπής στους εξωτερικούς τροχούς της στροφής, θα αυξάνεται παράλληλα η διαθέσιμη ροπή στον οπίσθιο άξονα (επιλέγεται $u_4 = 0.2$, βλ. Ενότητα 2.4). Αντιθέτως, όταν εκτελείται μεταφορά ροπής στους εσωτερικούς τροχούς, θα αυξάνεται η διαθέσιμη ροπή στον εμπρόσθιο άξονα (επιλέγεται $u_4 = 0.8$). Τα αποτελέσματα της προσομοίωσης παρατίθενται παρακάτω, στα Σχ. 3.5, 3.6, 3.7. Το απόσπασμα κώδικα για την προσομοίωση παρατίθεται στο Παράρτημα Α, Α.11.



Σχήμα 3.5: Τροχιά αριστερής στροφής με $v_0 = 15\,m/s$, $v_\delta = 0.005\,rad/s$ και $a_{\text{long}} = 0.2g$, κατανομή ροπής $\Delta T = 100\,Nm$ και $u_4 = 0.2/0.8$



Σχήμα 3.6: Γωνία εκτροπής αριστερής στροφής με $v_0 = 15\,m/s$, $v_\delta = 0.005\,rad/s$ και $a_{\text{long}} = 0.2g$, κατανομή ροπής $\Delta T = 100\,Nm$ και $u_4 = 0.2/0.8$



Σχήμα 3.7: Ρυθμός εκτροπής αριστερής στροφής με $v_0 = 15\,m/s$, $v_\delta = 0.005\,rad/s$ και $a_{\text{long}} = 0.2g$, κατανομή ροπής $\Delta T = 100\,Nm$ και $u_4 = 0.2/0.8$

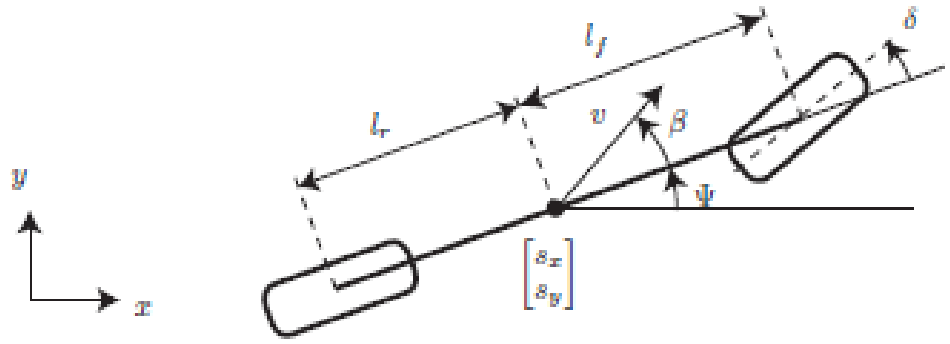
Πράγματι, από το διάγραμμα της τροχιάς και του ρυθμού εκτροπής του οχήματος φαίνεται ότι η συνδυαστική κατανομή ροπής μεταξύ των αριστερών-δεξιών τροχών και του εμπρόσθιου-οπίσθιου άξονα ενίσχυσε τη δημιουργία διορθωτικής ροπής στρέψης στο αμάξωμα. Στο Σχ. 3.5 συγκρίνεται η τροχιά του αυτοκινήτου εφαρμόζοντας αριστερό-δεξί TV με την τροχιά όταν εφαρμοστεί επιπλέον εμπρόσθιο-οπίσθιο TV. Παρατηρείται στενότερη τροχιά όταν τα 2 συστήματα δρουν συνδυαστικά, ωστόσο η διαφορά φαίνεται να είναι ελάχιστη. Στο Σχ. 3.7 συγκρίνεται ο ρυθμός εκτροπής του αυτοκινήτου εφαρμόζοντας αριστερό-δεξί TV με το ρυθμό εκτροπής όταν εφαρμοστεί επιπλέον εμπρόσθιο-οπίσθιο TV. Σε αυτή την περίπτωση, οι διαφορές που παρατηρούνται είναι ακόμα μικρότερες όταν τα 2 συστήματα δρουν συνδυαστικά. Η επίδραση του αριστερού-δεξιού και του εμπρόσθιου-οπίσθιου TV συνεργατικά, αλλά και μεμονωμένα, θα εξεταστούν εις βάθος στη συνέχεια,

3.3 Υλοποίηση συστήματος κατανομής ροπής

Σε αυτή την Ενότητα θα υλοποιηθεί ένα σύστημα κατανομής ροπής, το οποίο θα προσαρμοστεί στο μοντέλο του οχήματος που περιγράφηκε στο Κεφάλαιο 2.

Ιδανικό μοντέλο οχήματος

Προκειμένου να εφαρμοστεί τακτική ελέγχου που θα αναλάβει την κατανομή της ροπής του κινητήρα στους τροχούς, πρέπει να οριστεί ένα μοντέλο το οποίο θα παρουσιάζει την επιθυμητή συμπεριφορά του οχήματος κατά τη διάρκεια ελιγμών. Ένα μοντέλο που χρησιμοποιείται από πολλούς ερευνητές για τον σκοπό αυτό είναι το μοντέλο ποδηλάτου (Single Track Model ή Bicycle Model). Πρόκειται για ένα γραμμικοποιημένο μοντέλο που αναπαριστά την τροχιά ενός οχήματος όταν αυτό βρίσκεται σε σταθερή κατάσταση στροφής (steady state cornering) [[9] σελ.59], δηλαδή δεν παρουσιάζει υποστροφή ή υπερστροφή (βλ. Ενότητα 3.1). Παρακάτω φαίνεται ένα μοντέλο ποδηλάτου.



Σχήμα 3.8: Μοντέλο μισού οχήματος - ποδηλάτου

Ιδανικός ρυθμός εκτροπής

Τα μεγέθη που θα συγκρίνονται κάθε χρονική στιγμή προκειμένου να προκύπτει το σφάλμα που θα εισάγεται στον αλγόριθμο ελέγχου είναι ο ρυθμός εκτροπής του ιδανικού μοντέλου (επιθυμητός) και ο πραγματικός ρυθμός εκτροπής του οχήματος της προσομοίωσης. Το μέγεθος αυτό εξετάζεται διότι είναι άμεσα συνδεδεμένο με τα φαινόμενα υποστροφής και υπερστροφής. Πιο συγκεκριμένα, όταν ο πραγματικός ρυθμός εκτροπής είναι μεγαλύτερος του ιδανικού, αυτό δηλώνει ότι το αυτοκίνητο τείνει να στραφεί περισσότερο από το επιθυμητό για δεδομένη γωνία στροφής τιμονιού, συνεπώς παρατηρείται υπερστροφή. Αντιθέτως, πραγματικός ρυθμός εκτροπής μικρότερος του ιδανικού παραπέμπει σε κατάσταση υποστροφής. Σύμφωνα με το βιβλίο του Rajamani [9], σελ. 231, για γωνία διεύθυνσης δ και ταχύτητα οχήματος V, ο ιδανικός ρυθμός εκτροπής υπολογίζεται ως εξής:

$$\dot{\psi}_{\text{des}} = \frac{V\delta}{l_f + l_r + \frac{mV^2(l_r C_{ar} - l_f C_{af})}{2C_{af}C_{ar}L}}\,, \tag{3.11}$$

όπου l_f, l_r οι αποστάσεις του εμπρόσθιου και του οπίσθιου άξονα από το κέντρο βάρους του αυτοκινήτου και C_{af}, C_{ar} οι Cornering Stiffness των εμπρόσθιων και των οπίσθιων ελαστικών αντίστοιχα.

Ωστόσο, ο επιθυμητός ρυθμός εκτροπής δεν είναι πάντα δυνατό να επιτευχθεί. Δεν είναι ασφαλές, για παράδειγμα, να θεωρηθεί η παραπάνω τιμή ιδανικού ρυθμού εκτροπής (βλ. 3.11), εάν ο συντελεστής τριβής του εδάφους δεν είναι σε θέση να παράσχει τις απαιτούμενες δυνάμεις ελαστικών. Ως εκ τούτου, ο επιθυμητός ρυθμός εκτροπής πρέπει να περιορίζεται από μια συνάρτηση του συντελεστή τριβής του ελαστικού και του επιπέδου. Αυτό υλοποιείται φράζοντας την πλευρική επιτάχυνση από αυτόν τον συντελεστή. Η παραπάνω τακτική αναλύεται εις βάθος στο βιβλίο του Rajamani [9], σελ. 233-234. Συνεπώς, ο ιδανικός ρυθμός εκτροπής του μοντέλου θα υπολογίζεται από την εξίσωση 3.11, εκτός αν είναι μεγαλύτερος από το άνω όριο που θα χρησιμοποιηθεί, το οποίο τίθεται ως:

$$\dot{\psi}_{\text{upperbound}} = \frac{0.85\mu g}{V} \tag{3.12}$$

Στο απόσπασμα κώδικα A.12 που βρίσκεται στο Παράρτημα Α παρατίθεται η συνάρτηση ***desired_values***, η οποία δέχεται ως ορίσματα τη γωνία διεύθυνσης δ, την ταχύτητα V και την επιτάχυνση $u_2 = a_{\text{long}}$ (βλ. εξ. 3.6) και επιστρέφει τον επιθυμητό ρυθμό εκτροπής, λαμβάνοντας υπόψη και τον παραπάνω περιορισμό.

Αλγόριθμος για αριστερό - δεξί TV

Γίνεται χρήση ενός PD ελεγκτή (Proportional-Derivative Controller), ο οποίος δέχεται ως είσοδο το σφάλμα του ρυθμού εκτροπής και επιστρέφει ως έξοδο το μέγεθος και το πρόσημο της διορθωτικής ροπής που θα προστεθεί στους τροχούς κάθε μεριάς του αυτοκινήτου. Οι ελεγκτές PD χρησιμοποιούνται ευρέως σε συστήματα ελέγχου, συμπεριλαμβανομένων των εφαρμογών TV, καθώς προσφέρουν ισορροπία μεταξύ αναλογικών (proportional) και παράγωγων (derivative) δράσεων, καθιστώντας τους κατάλληλη επιλογή για τον έλεγχο δυναμικών συστημάτων όπως τα οχήματα. Η αναλογική συνιστώσα επιτρέπει στον ελεγκτή να ανταποκρίνεται άμεσα στο παρόν σφάλμα (ρυθμού εκτροπής στη συγκεκριμένη περίπτωση), βοηθώντας στην καθοδήγηση του οχήματος προς την επιθυμητή τροχιά. Η παράγωγη συνιστώσα λαμβάνει υπόψη το ρυθμό μεταβολής του σφάλματος, επιτρέποντας στον ελεγκτή να προβλέπει μελλοντικές τάσεις και να προβαίνει σε προληπτικές προσαρμογές για την αποφυγή υπερπηδήσεων ή ταλαντώσεων. Αυτό το χαρακτηριστικό είναι χρήσιμο για τη διατήρηση της σταθερότητας κατά τη διάρκεια των ελιγμών. Επιπλέον, οι ελεγκτές PD είναι σχετικά απλοί στην υλοποίηση και υπολογιστικά αποδοτικοί, καθιστώντας τους μια ρεαλιστική επιλογή για εφαρμογές πραγματικού χρόνου στην αυτοκινητοβιομηχανία.

Ακολουθεί αναλυτική περιγραφή του αλγορίθμου που υλοποιήθηκε, ο οποίος εφαρμόζεται σε κάθε χρονικό βήμα της προσομοίωσης:

- Υπολογισμός διαθέσιμης ροπής από τον κινητήρα (T_E), λαμβάνοντας υπόψη τους περιορισμούς της επιτάχυνσης (βλ. Ενότητα 2.4)
- Υπολογισμός του σφάλματος, αφαιρώντας την πραγματική τιμή του ρυθμού εκτροπής από την επιθυμητή
- Χρήση ενός PD Ελεγκτή για τον υπολογισμό της διορθωτικής ροπής ΔT (βλ. Ενότητα 3.2). Για θετικές τιμές σφάλματος ($\psi_{\text{des}} > \psi_{\text{actual}}$) το όχημα υποστρέφει, οπότε μεταφέρεται μεγαλύτερο μέρος της ροπής στους εξωτερικούς τροχούς της στροφής. Σε αντίθετη περίπτωση, αυξάνεται η ροπή στους τροχούς εσωτερικά της στροφής (βλ. Ενότητα 3.1). Η έξοδος του ελεγκτή βελτιστοποιείται έπειτα από ρύθμιση των παραμέτρων του (k_p και k_d)
- Φραγή της τιμής εξόδου του ελεγκτή μεταξύ των τιμών $[-\frac{T_E}{4}, \frac{T_E}{4}]$, αφού αυτή είναι η μέγιστη μεταφορά ροπής που δύναται να γίνει μεταξύ των δεξιών και των αριστερών τροχών κάθε άξονα
- Ενημέρωση του διανύσματος εισόδου (βλ. εξ. 3.1) με την τιμή της διορθωτική ροπής που υπολογίστηκε για το συγκεκριμένο χρονικό βήμα της προσομοίωσης
- Ενημέρωση του μοντέλου οχήματος με το ανανεωμένο διάνυσμα εισόδου, το οποίο περιέχει την τιμή ΔT, που θα εκτελέσει την κατανομή ροπής για το συγκεκριμένο χρονικό βήμα της προσομοίωσης

Αλγόριθμος για εμπρόσθιο - οπίσθιο TV

Εκτελείται γραμμική παρεμβολή προκειμένου να ρυθμιστεί η τιμή της παραμέτρου $T_{s,e}$ του διανύσματος εισόδου (βλ. εξ. 3.6), με βάση την τιμή του σφάλματος του ρυθμού εκτροπής. Ο αλγόριθμος λειτουργεί ως εξής: Όταν το σφάλμα είναι θετικό (επιθυμητός ρυθμός εκτροπής $>$ πραγματικού), οπότε το όχημα βρίσκεται σε κατάσταση υποστροφής, η παράμετρος $T_{s,e}$ μειώνεται για να κατανεμηθεί μεγαλύτερο μέρος της ροπής στον οπίσθιο άξονα (βλ. εξ.2.30 έως 2.33). Αντιθέτως, όταν ο επιθυμητός ρυθμός εκτροπής $<$ πραγματικό, η παράμετρος $T_{s,e}$ αυξάνεται για να κατανεμηθεί μεγαλύτερο μέρος στον εμπρόσθιο άξονα. Οι τακτικές αυτές αναλύθηκαν προηγουμένως στην Ενότητα 3.1. Αναλυτικότερα:

- Χρησιμοποιείται η εντολή ***interp1d*** της κλάσης *scipy.interpolate*, η οποία παρεμβάλλει γραμμικά τις τιμές σφάλματος στο διάστημα [0.1, 0.9], το οποίο ορίζει τις οριακές τιμές της παραμέτρου ελέγχου $T_{s,e}$. Η επιλογή του διαστήματος [0.1, 0.9] σημαίνει ότι ο αλγόριθμος θα είναι σε θέση να απαιτεί μεταφορά ροπής έως και του 90% της διαθέσιμης σε έναν άξονα.
- Οι τιμές σφάλματος αρχικά φράζονται στα $[-1.5, 1.5]\, rad/s$, που σημαίνει ότι εξωτερικά αυτών η παράμετρος $T_{s,e}$ παίρνει τις οριακές της τιμές. Η ευαισθησία του ελεγκτή αυξάνεται όσο το διάστημα αυτό μικραίνει, αφού η $T_{s,e}$ λαμβάνει τις οριακές τιμές της γρηγορότερα.

Ο παραπάνω αλγόριθμος υλοποιείται με τη συνάρτηση ***tune__T__se***, η οποία παρατίθεται στο Παράρτημα Α, απόσπασμα κώδικα A.13.

Κεφάλαιο 4

Εφαρμογή Torque Vectoring στο όχημα

Σε αυτό το Κεφάλαιο θα γίνει προσαρμογή των αλγορίθμων που παρουσιάστηκαν στην Ενότητα 3.3 στο μοντέλο του οχήματος (βλ. Κεφάλαιο 2). Για την αξιολόγηση του συστήματος που δημιουργήθηκε θα εκτελεσθούν τρεις ελιγμοί με τρεις διαφορετικές αρχικές ταχύτητες ($4\,m/s$ ή $14.4\,km/h$, $8\,m/s$ ή $28.8\,km/h$, $15\,m/s$ ή $54\,km/h$) για κάθε ελιγμό:

1. Επιταχυνόμενη αριστερή στροφή με ρυθμό αύξησης γωνίας διεύθυνσης $v_\delta = 0.05\,rad/s$ και επιτάχυνση $a_{\text{long}} = 0.2g$

2. Επιβολή ακαριαίας γωνίας διεύθυνσης $\delta = 0.314\,rad$ και επιτάχυνσης $a_{\text{long}} = 0.3g$ ενώ το αυτοκίνητο κινείται σε ευθεία με σταθερή ταχύτητα (step-steer maneuver)

3. Ελιγμός αποφυγής εμποδίου στο οδόστρωμα (obstacle avoidance maneuver). Καθώς το αυτοκίνητο ταξιδεύει με αρχική ταχύτητα, ο οδηγός αναγκάζεται να αποφύγει εμπόδιο και να επιστρέψει στη λωρίδα του όσο πιο γρήγορα και ασφαλώς γίνεται, προκειμένου να μην προκληθεί ατύχημα με τα αυτοκίνητα που πλησιάζουν από την απέναντι λωρίδα. Για τον σκοπό αυτό επιλέγεται επιτάχυνση $a_{\text{long}} = 0.2g$. Προκειμένου να προσομοιωθεί η ιδανική τροχιά του οχήματος για κάθε περίπτωση αρχικής ταχύτητας, χρησιμοποιείται το Single Track Drift Model του Commonroad ([1], σελ.11). Το σχετικό απόσπασμα κώδικα θα παρουσιαστεί στη συνέχεια. Εκτελώντας δοκιμές πάνω στο μοντέλο αυτό, εξάγεται για κάθε περίπτωση αρχικής ταχύτητας η συνάρτηση του ρυθμού αύξησης γωνίας διεύθυνσης που απαιτείται για να εκτελεσθεί ο ελιγμός αποφυγής εμποδίου:

   - Για $v_0 = 4\,m/s$: $v_\delta = -0.5\cos\left(\frac{2\pi}{4}\cdot \text{time}[i] + \frac{\pi}{35}\right)$
   - Για $v_0 = 8\,m/s$: $v_\delta = -0.5\cos\left(\frac{2\pi}{4}\cdot \text{time}[i] + \frac{\pi}{10.1}\right)$
   - Για $v_0 = 15\,m/s$: $v_\delta = -0.5\cos\left(\frac{2\pi}{4}\cdot \text{time}[i] + \frac{\pi}{6.13}\right)$,

   όπου $\text{time}[i]$ ο τρέχον χρόνος στο i-οστό βήμα της προσομοίωσης.

Οι ελιγμοί θα πραγματοποιηθούν ακολουθώντας την παρακάτω φιλοσοφία:

- Σύστημα TV ανενεργό
- Αριστερό-δεξί TV ενεργό
- Εμπρόσθιο-οπίσθιο TV ενεργό
- Αριστερό-δεξί & εμπρόσθιο-οπίσθιο TV ενεργό

4.1 Επιταχυνόμενη αριστερή στροφή

4.1.1 Σύστημα TV ανενεργό

Ο ελιγμός εκτελείται δύο φορές. Αρχικά χωρίς επιβολή ροπής κινητήρα ή φρένου και ύστερα με την τιμή της επιτάχυνσης που προαναφέρθηκε. Έτσι αποτυπώνεται καλύτερα η επίδραση της επιτάχυνσης κατά τη διάρκεια στροφής. Το διάνυσμα εισόδου (βλ. 3.6) για την συγκεκριμένη προσομοίωση ορίζεται ως εξής:

$$u_{\text{coast}} = (0.05,\, 0,\, 0,\, 0.5)$$

και

$$u_{\text{acc}} = (0.05,\, 0.2g,\, 0,\, 0.5)$$

Τα σχήματα αποτελεσμάτων παρατίθενται στα Σχ.5.1 - Σχ.5.9, με τη λογική σειρά που αναφέρθηκε προηγουμένως. Όλα τα σχήματα προσομοίωσης των ελιγμών με όλους τους συνδυασμούς εφαρμογής των συστημάτων TV βρίσκονται στο Κεφάλαιο 5, συνοδευόμενα από αναλυτικό σχολιασμό.

4.1.2 Αριστερό - δεξί TV ενεργό

Εκτέλεση του ίδιου ελιγμού με αριστερό-δεξί TV. Η υλοποίηση σε Python του αλγόριθμου που αναπτύχθηκε στην Ενότητα 3.3 και η προσαρμογή του στο μοντέλο αναπαράστασης του οχήματος φαίνεται στο Παράρτημα Α, απόσπασμα κώδικα Α.14. Τα σχήματα αποτελεσμάτων παρατίθενται στα Σχ.5.10 - Σχ.5.18.

4.1.3 Εμπρόσθιο - οπίσθιο TV ενεργό

Η υλοποίηση σε Python του αλγόριθμου που αναπτύχθηκε στην Ενότητα 3.3 και η προσαρμογή του στο μοντέλο αναπαράστασης του οχήματος φαίνεται στο Παράρτημα Α, απόσπασμα κώδικα Α.15. Στη γραμμή 43 εξάγεται για κάθε χρονικό βήμα της προσομοίωσης η τιμή της παραμέτρου ελέγχου $T_{s,e}$, ενώ στη γραμμή 53 ενημερώνεται το μοντέλου αναπαράστασης του οχήματος με το ανανεωμένο διάνυσμα εισόδου. Τα σχήματα αποτελεσμάτων παρατίθενται στα Σχ.5.43 - Σχ.5.51.

4.1.4 Αριστερό - δεξί & εμπρόσθιο-οπίσθιο TV ενεργό

Ο κώδικας υλοποίησης των δυο τακτικών συνδυαστικά παρατίθεται στο Παράρτημα Α, Α.16. Τα σχήματα αποτελεσμάτων παρατίθενται στα Σχ.5.70 - Σχ.5.78.

4.2 Step - steer maneuver

4.2.1 Σύστημα TV ανενεργό

Το διάνυσμα εισόδου (βλ. 3.6) για την συγκεκριμένη προσομοίωση ορίζεται ως εξής:

$$u = (0.,\ 0.3g,\ 0,\ 0.5)$$

Τα σχήματα αποτελεσμάτων παρατίθενται στα Σχ.5.19 - Σχ.5.21 και ο κώδικας για την προσομοίωση στο Παράρτημα Α, Α.17.

4.2.2 Αριστερό - δεξί TV ενεργό

Εκτέλεση του ίδιου ελιγμού με αριστερό-δεξί TV. Η υλοποίηση σε Python και η προσαρμογή του στο μοντέλο αναπαράστασης του οχήματος φαίνεται στο Παράρτημα Α, απόσπασμα κώδικα Α.18. Τα αποτελέσματα παρατίθενται στα Σχ.5.22 - Σχ.5.30.

4.2.3 Εμπρόσθιο - οπίσθιο TV ενεργό

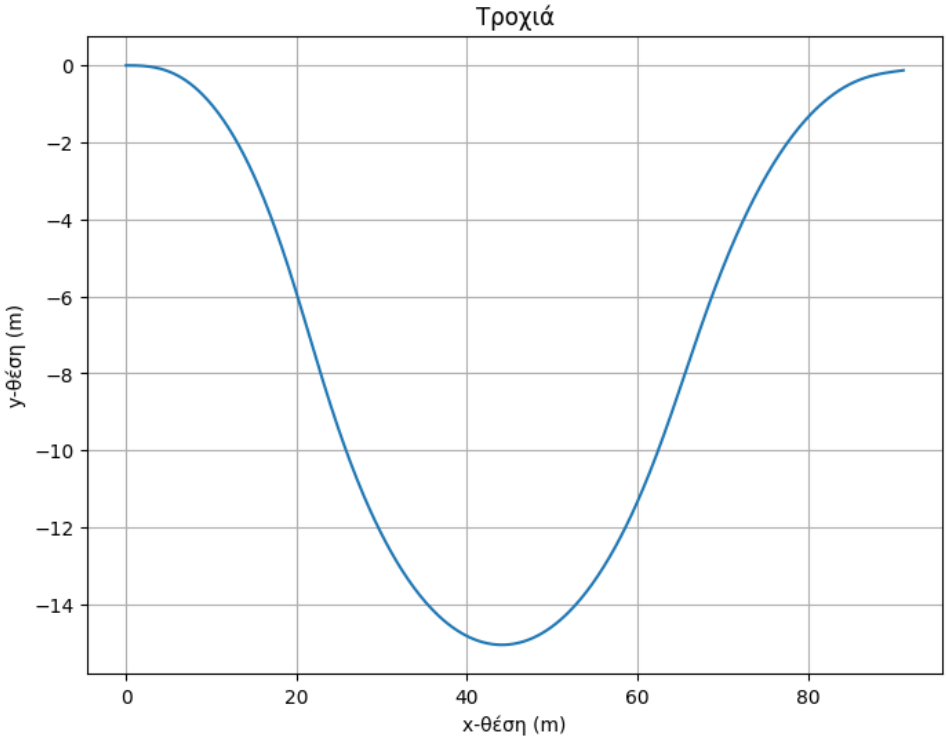
Η υλοποίηση σε Python φαίνεται στο Παράρτημα Α, απόσπασμα κώδικα Α.19. Τα αποτελέσματα παρατίθενται στα Σχ.5.52 - Σχ.5.60.

4.2.4 Αριστερό - δεξί & εμπρόσθιο-οπίσθιο TV ενεργό

Ο κώδικας υλοποίησης των δυο τακτικών συνδυαστικά παρατίθεται στο Παράρτημα Α, Α.20. Τα αποτελέσματα παρατίθενται στα Σχ.5.79 - Σχ.5.87.

4.3 Ελιγμός αποφυγής εμποδίου

Όπως αναφέρθηκε προηγουμένως, χρησιμοποιείται το Single Track Drift Model του Commonroad ([1], σελ.11) για την εξαγωγή της συνάρτησης του ρυθμού αύξησης γωνίας διεύθυνσης που απαιτείται για να εκτελεσθεί ο ελιγμός. Ο κώδικας για την περίπτωση του ελιγμού με $v_0 = 4\,m/s$ παρατίθεται στο Παράρτημα Α, Α.21. Η επιθυμητή τροχιά του ελιγμού φαίνεται παρακάτω στο Σχ.4.1. Ομοίως γίνεται και για τους ελιγμούς με $v_0 = 8\,m/s$ και $v_0 = 15\,m/s$. Η επιθυμητή τους τροχιά είναι παραπλήσια με το Σχ.4.1.



Σχήμα 4.1: Επιθυμητή τροχιά για ελιγμό αποφυγής εμποδίου με $v_0 = 4\,m/s$

4.3.1 Σύστημα TV ανενεργό

Το διάνυσμα εισόδου (βλ. 3.6) για την συγκεκριμένη προσομοίωση ορίζεται ως εξής:

$$u_{v_{0,i}} = (v_{\delta_{v_{0,i}}},\, 0.2g,\, 0,\, 0.5),\; i \in [1,\, 3]$$

όπου $v_{\delta_{v_{0,i}}}$ είναι η συνάρτηση του ρυθμού αύξησης γωνίας διεύθυνσης για κάθε αρχική ταχύτητα. Τα σχήματα αποτελεσμάτων παρατίθενται στα Σχ.5.31 - Σχ.5.33 και ο κώδικας για την προσομοίωση στο Παράρτημα Α, Α.22.

4.3.2 Αριστερό - δεξί TV ενεργό

Εκτέλεση του ίδιου ελιγμού με αριστερό-δεξί TV. Η υλοποίηση σε Python και η προσαρμογή του στο μοντέλο αναπαράστασης του οχήματος φαίνεται στο Παράρτημα Α, απόσπασμα κώδικα Α.23. Τα σχήματα αποτελεσμάτων παρατίθενται στα Σχ.5.34 - Σχ.5.42.

4.3.3 Εμπρόσθιο - οπίσθιο TV ενεργό

Η υλοποίηση σε Python φαίνεται στο Παράρτημα Α, απόσπασμα κώδικα Α.24. Τα σχήματα αποτελεσμάτων παρατίθενται στα Σχ.5.61 - Σχ.5.69.

4.3.4 Αριστερό - δεξί & εμπρόσθιο-οπίσθιο TV ενεργό

Ο κώδικας υλοποίησης των δυο τακτικών συνδυαστικά παρατίθεται στο Παράρτημα Α, Α.25. Ιδιαίτερο ενδιαφέρον παρουσιάζουν τα αποτελέσματα, τα οποία φαίνονται στα Σχ.5.88 - Σχ.5.96.

Κεφάλαιο 5

Αξιολόγηση αποτελεσμάτων και μελλοντική εργασία

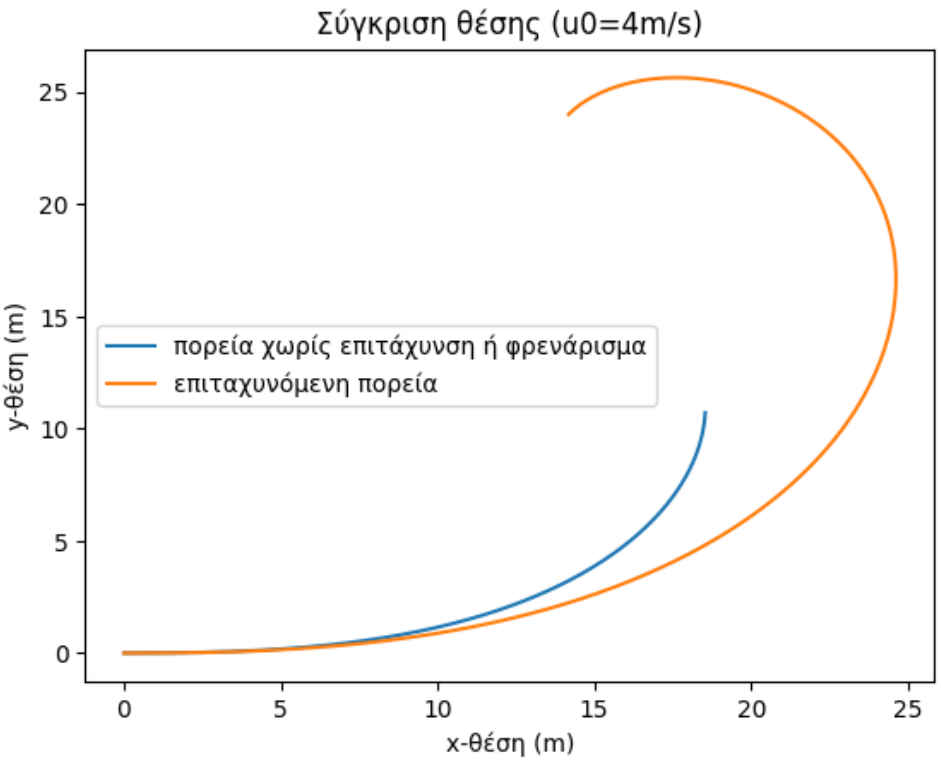
Σε αυτό το Κεφάλαιο θα συζητηθούν τα αποτελέσματα της εφαρμογής των αλγορίθμων κατανομής ροπής που αναπτύχθηκαν στο Κεφάλαιο 3 και θα αναφερθούν τρόποι που ενδέχεται να βελτιώσουν την αποδοτικότητα του συστήματος σε μελλοντική εργασία.

5.1 Αριστερό - δεξί TV

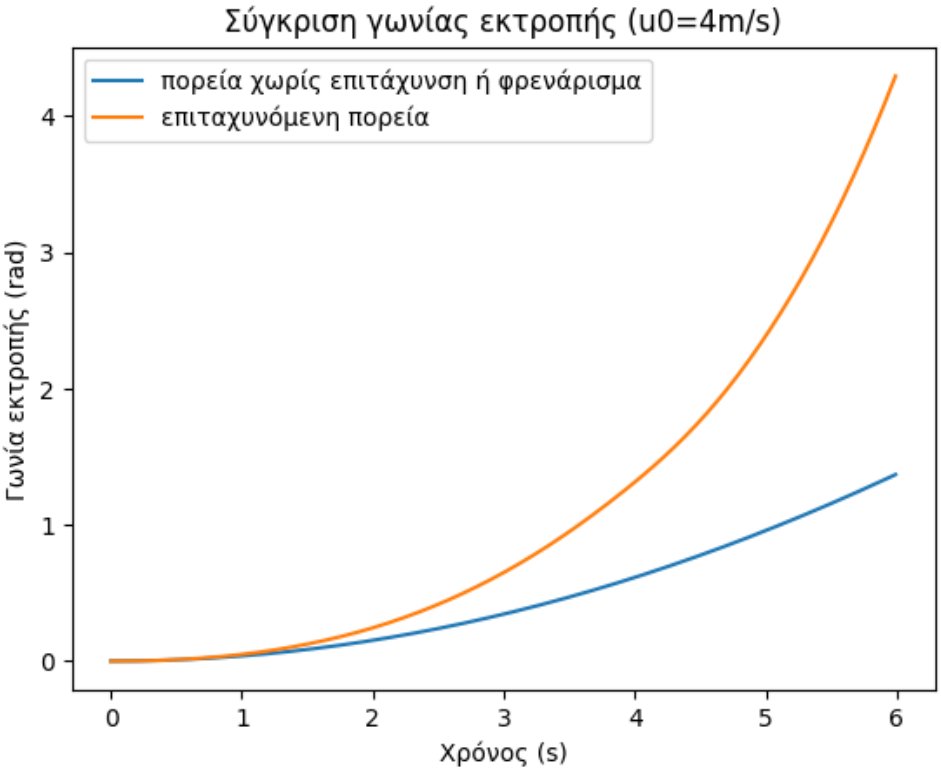
Η τακτική αυτή παρουσίασε τη μεγαλύτερη απόδοση, τόσο σε θέμα επιδόσεων κατά τη στροφή, όσο και σε αποφυγή επικίνδυνων καταστάσεων για τον οδηγό και τυχόν επιβάτες.

Επιταχυνόμενη αριστερή στροφή

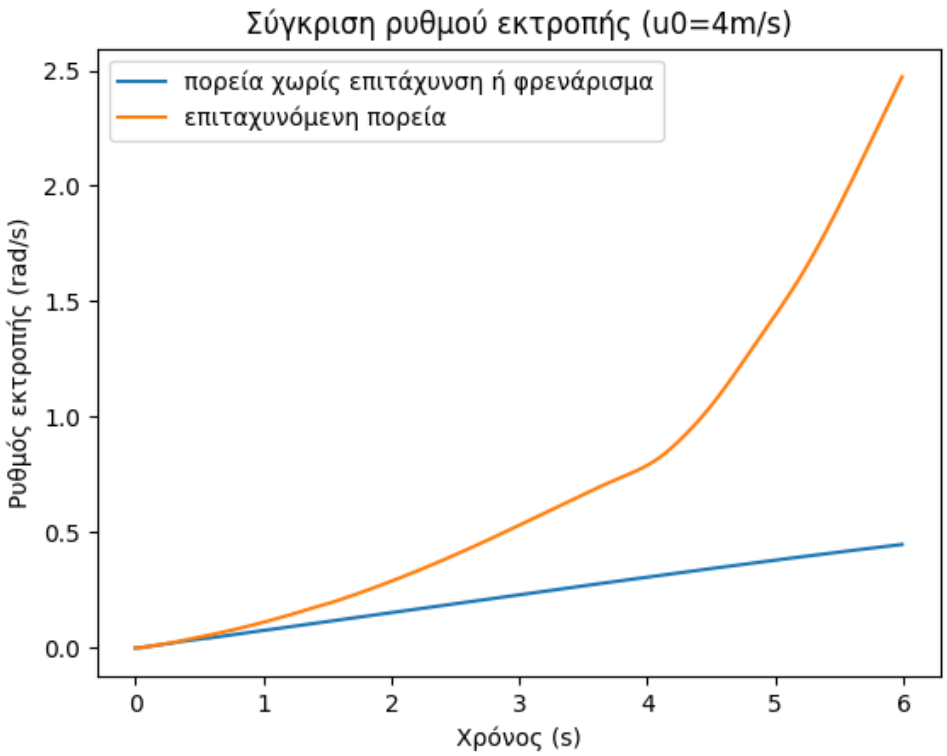
Στα Σχ.5.1 - Σχ.5.9 παρακάτω φαίνεται ο ελιγμός χωρίς την επίδραση του TV.



Σχήμα 5.1: Τροχιά επιταχυνόμενης αριστερής στροφής με ρυθμό αύξησης γωνίας διεύθυνσης $v_\delta = 0.05\,rad/s$ και επιτάχυνση $a_{\text{long}} = 0.2g$



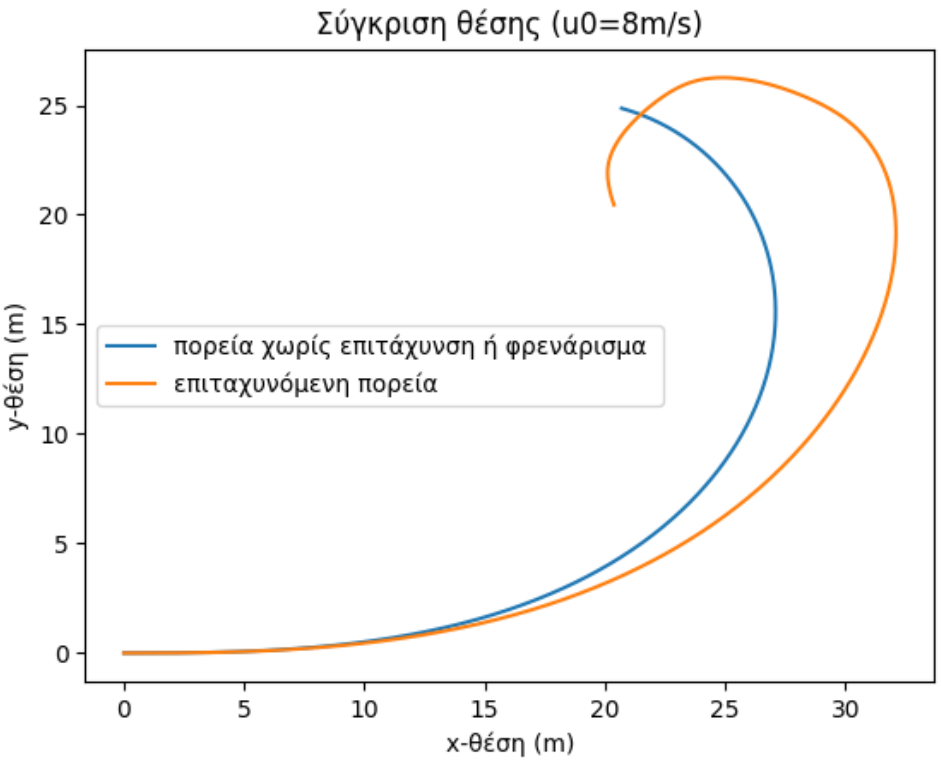
Σχήμα 5.2: Γωνία εκτροπής επιταχυνόμενης αριστερής στροφής με ρυθμό αύξησης γωνίας διεύθυνσης $v_\delta = 0.05\,rad/s$ και επιτάχυνση $a_{\text{long}} = 0.2g$



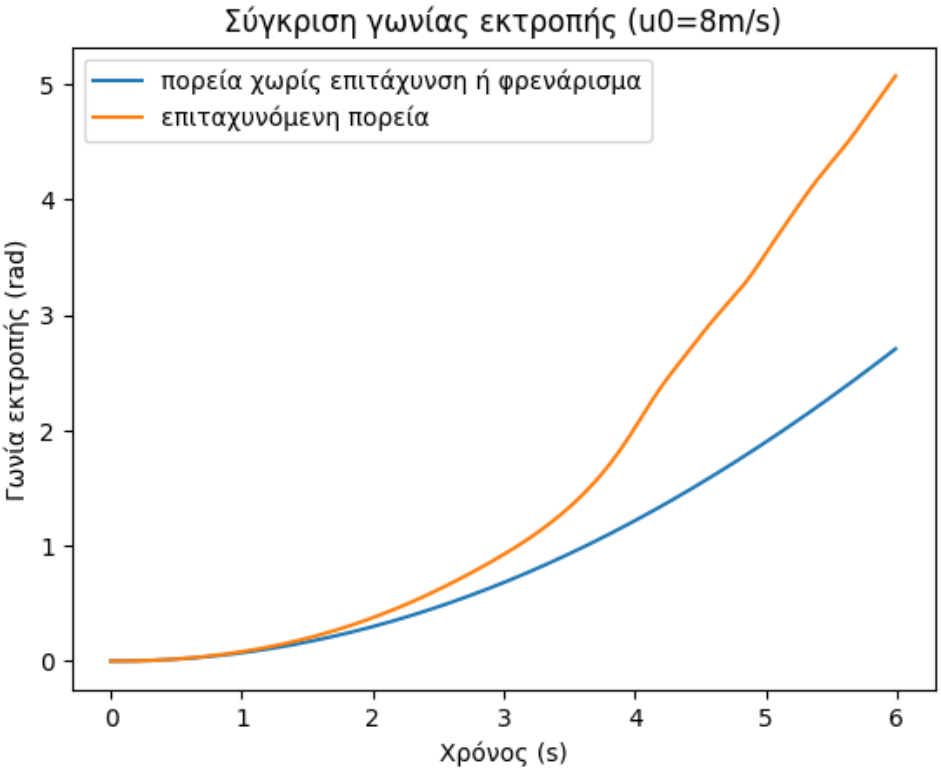
Σχήμα 5.3: Ρυθμός εκτροπής επιταχυνόμενης αριστερής στροφής με ρυθμό αύξησης γωνίας διεύθυνσης $v_\delta = 0.05\,rad/s$ και επιτάχυνση $a_{\text{long}} = 0.2g$

Στα σχήματα 5.1 - 5.3 παρουσιάζονται τα αποτελέσματα των προσομοιώσεων για δύο περιπτώσεις: Αρχικά χωρίς επιβολή ροπής κινητήρα ή φρένου και ύστερα με επιτάχυνση $a_{\text{long}} = 0.2g$. Έτσι αποτυπώνεται καλύτερα η επίδραση της επιβολής ροπής κινητήρα κατά τη διάρκεια στροφής στη συμπεριφορά του οχήματος. Από Σχ. 5.3 παρατηρείται συνεχόμενη αύξηση του ρυθμού εκτροπής του επιταχυνόμενου οχήματος σε σχέση με το μη επιταχυνόμενο. Συνεπώς, επιβολή ροπής κινητήρα επάνω στη στροφή δημιουργεί φαινόμενο υπερστροφής σε αυτή την προσομοίωση.

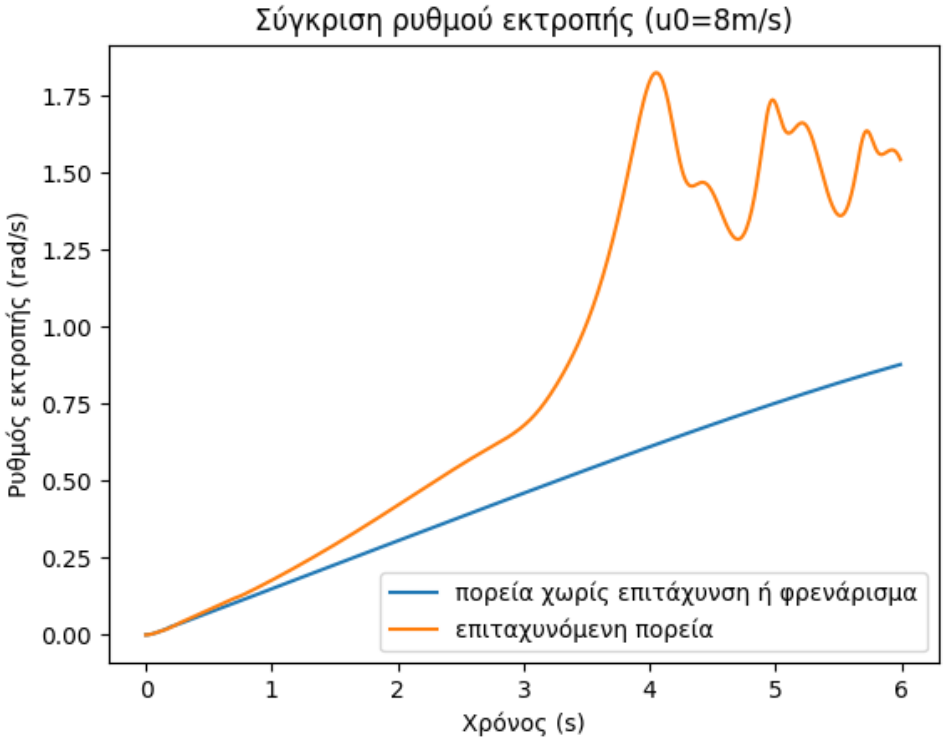
Ωστόσο, για $v_0 = 4\,m/s$ δεν παρατηρείται κάποια ασταθής κατάσταση κατά τη διάρκεια του ελιγμού, αφού η ταχύτητα είναι ακόμα χαμηλή. Αυτό φαίνεται από τα Σχ. 5.2 και 5.3, όπου δεν υπάρχουν ραγδαίες μεταβολές στη γωνία και στο ρυθμό εκτροπής του οχήματος.



Σχήμα 5.4: Τροχιά επιταχυνόμενης αριστερής στροφής με ρυθμό αύξησης γωνίας διεύθυνσης $v_\delta = 0.05\,rad/s$ και επιτάχυνση $a_{\text{long}} = 0.2g$

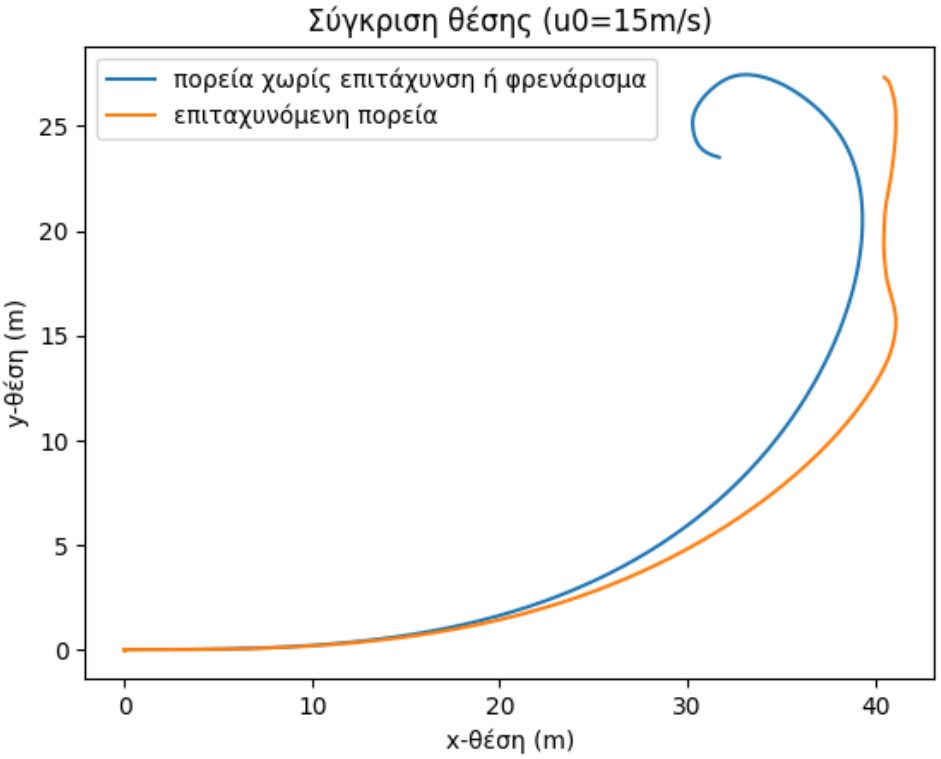


Σχήμα 5.5: Γωνία εκτροπής επιταχυνόμενης αριστερής στροφής με ρυθμό αύξησης γωνίας διεύθυνσης $v_\delta = 0.05\,rad/s$ και επιτάχυνση $a_{\text{long}} = 0.2g$

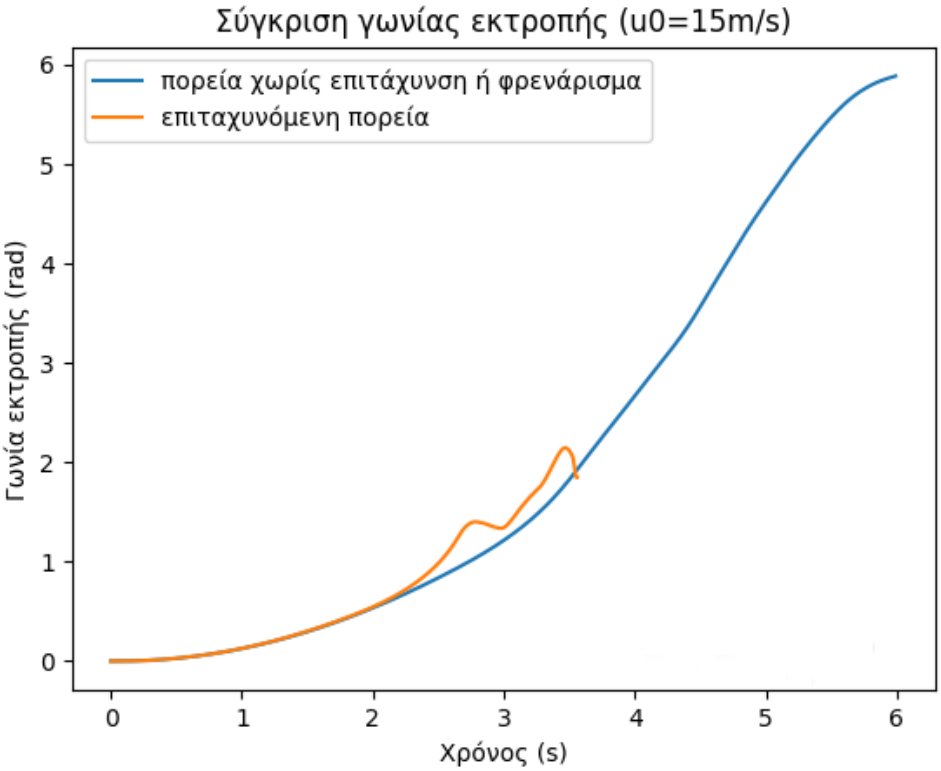


Σχήμα 5.6: Ρυθμός εκτροπής επιταχυνόμενης αριστερής στροφή με ρυθμό αύξησης γωνίας διεύθυνσης $v_\delta = 0.05\,rad/s$ και επιτάχυνση $a_{\text{long}} = 0.2g$

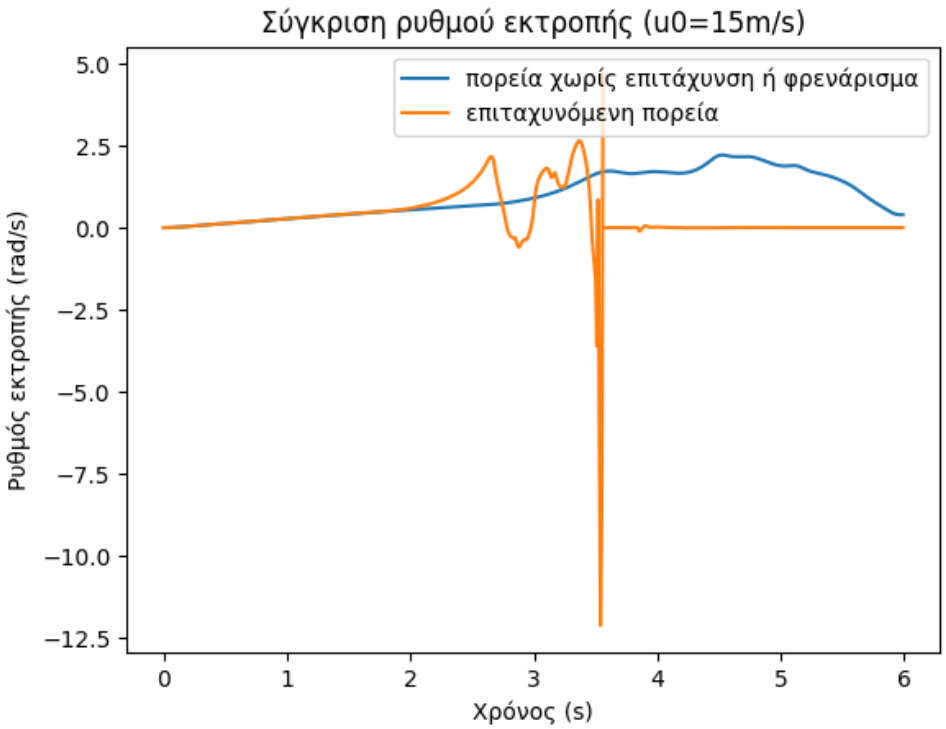
Για $v_0 = 8\,m/s$, παρατηρείται μεγάλη διακύμανση της τάξεως των 0.5 rad/s στο διάγραμμα του ρυθμού εκτροπής (Σχ.5.6), που σημαίνει πως το όχημα πλησιάζει σε οριακές καταστάσεις πρόσφυσης. Οι αυξομειώσεις αυτές παραπέμπουν σε αλλεπάλληλα φαινόμενα υπερστροφής και υποστροφής. Το ίδιο συμπέρασμα εξάγεται μελετώντας το Σχ. 5.5, παρατηρώντας τις διακυμάνσεις που εκδηλώνει ο προσανατολισμός του οχήματος. Η τροχιά (Σχ. 5.4) εκδηλώνει επίσης πιο ασταθή συμπεριφορά αφού παρατηρούνται πιο επιθετικές αλλαγές κατευθύνσεων σε σχέση με την τροχιά του ελιγμού με αρχική ταχύτητα $v_0 = 4\,m/s$ (Σχ. 5.1).



Σχήμα 5.7: Τροχιά επιταχυνόμενης αριστερής στροφής με ρυθμό αύξησης γωνίας διεύθυνσης $v_\delta = 0.05\,rad/s$ και επιτάχυνση $a_{\text{long}} = 0.2g$



Σχήμα 5.8: Γωνία εκτροπής επιταχυνόμενης αριστερής στροφής με ρυθμό αύξησης γωνίας διεύθυνσης $v_\delta = 0.05\,rad/s$ και επιτάχυνση $a_{\text{long}} = 0.2g$

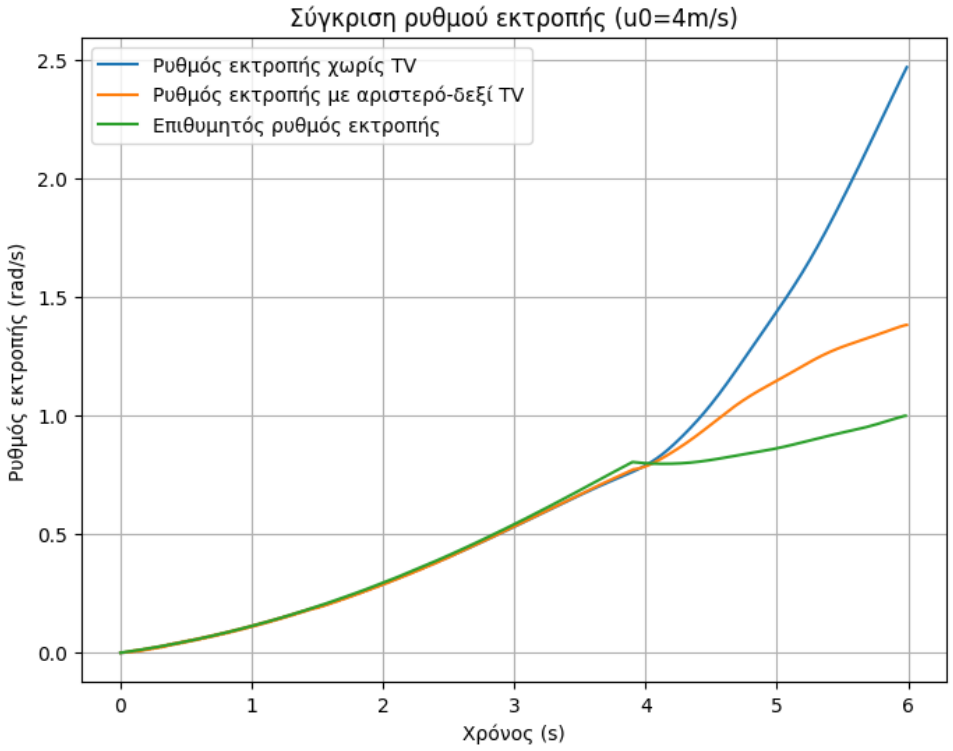


Σχήμα 5.9: Ρυθμός εκτροπής επιταχυνόμενης αριστερής στροφής με ρυθμό αύξησης γωνίας διεύθυνσης $v_\delta = 0.05\,rad/s$ και επιτάχυνση $a_{\text{long}} = 0.2g$

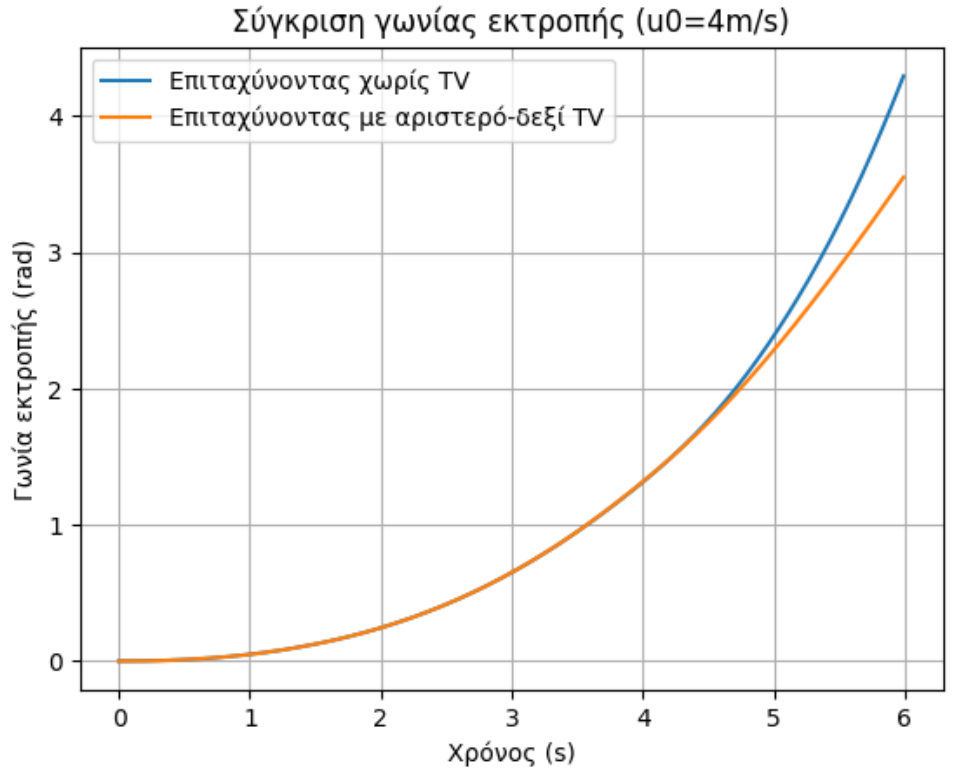
Για $v_0 = 15\,m/s$, το όχημα χάνει τον έλεγχο και εκτρέπεται της πορείας του λίγο πριν τη χρονική στιγμή $t = 4\,s$. Αυτό γίνεται φανερό από το Σχ.5.8, όπου η γωνία εκτροπής ξαφνικά αρχίζει να μειώνεται ραγδαία και ύστερα το μοντέλο σταματά να παράγει τιμές. Το ίδιο προκύπτει και από το Σχ.5.9 του ρυθμού εκτροπής, στο οποίο παρατηρείται πολύ μεγάλη αύξηση και αμέσως μετά πολύ μεγάλη μείωση της τιμής, αγγίζοντας τα 12.5 rad/s, μέγεθος που υποδηλώνει πως το όχημα παρεκτράπηκε βίαια. Η συμπεριφορά αυτή είναι εμφανώς επικίνδυνη. Στο διάγραμμα της θέσης, Σχ.5.7, αποτυπώνεται η ασταθής πορεία του οχήματος που οδήγησε σε απώλεια ελέγχου.

Όπως ήταν αναμενόμενο, υψηλότερες ταχύτητες έναρξης των ελιγμών οδήγησαν σε επιδείνωση της συμπεριφοράς του οχήματος.

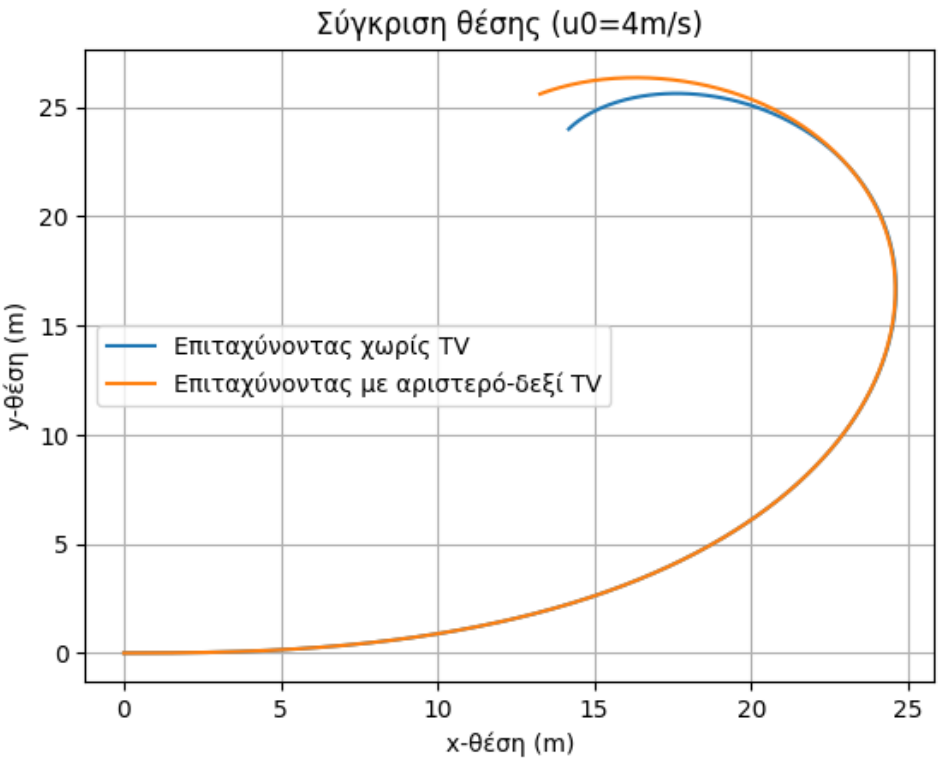
Στα Σχ.5.10 - Σχ.5.18 που ακολουθούν φαίνεται ο ελιγμός με το σύστημα κατανομής ροπής ενεργό.



Σχήμα 5.10: Ρυθμός εκτροπής επιταχυνόμενης αριστερής στροφής με ρυθμό αύξησης γωνίας διεύθυνσης $v_\delta = 0.05\,rad/s$ και επιτάχυνση $a_{\text{long}} = 0.2g$

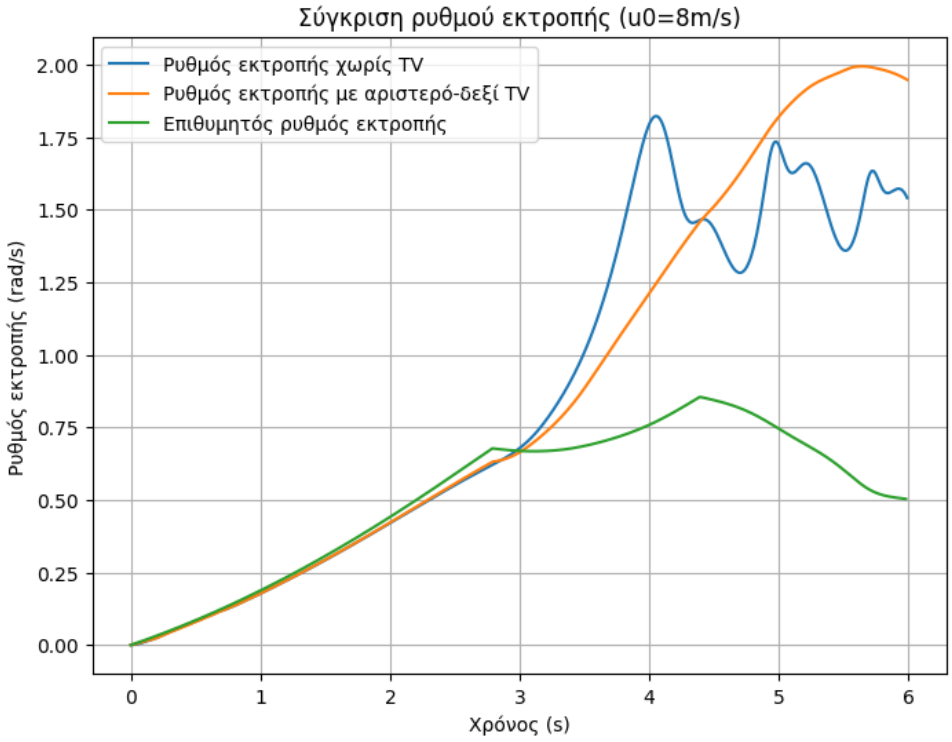


Σχήμα 5.11: Γωνία εκτροπής επιταχυνόμενης αριστερής στροφής με ρυθμό αύξησης γωνίας διεύθυνσης $v_\delta = 0.05\,rad/s$ και επιτάχυνση $a_{\text{long}} = 0.2g$

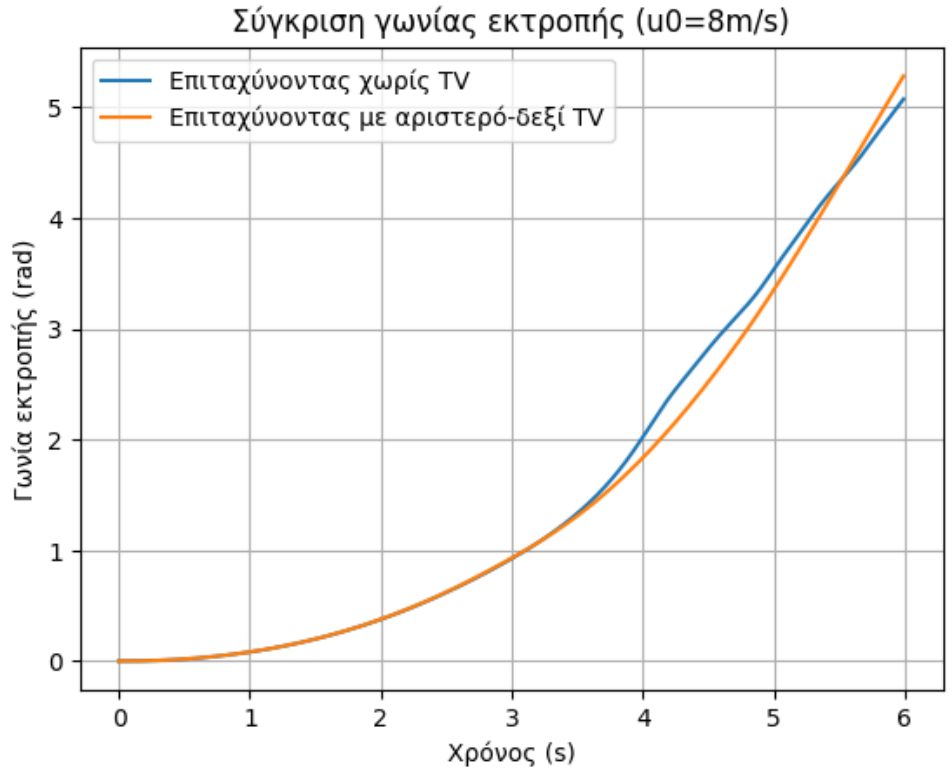


Σχήμα 5.12: Τροχιά επιταχυνόμενης αριστερής στροφής με ρυθμό αύξησης γωνίας διεύθυνσης $v_\delta = 0.05\,rad/s$ και επιτάχυνση $a_{\text{long}} = 0.2g$

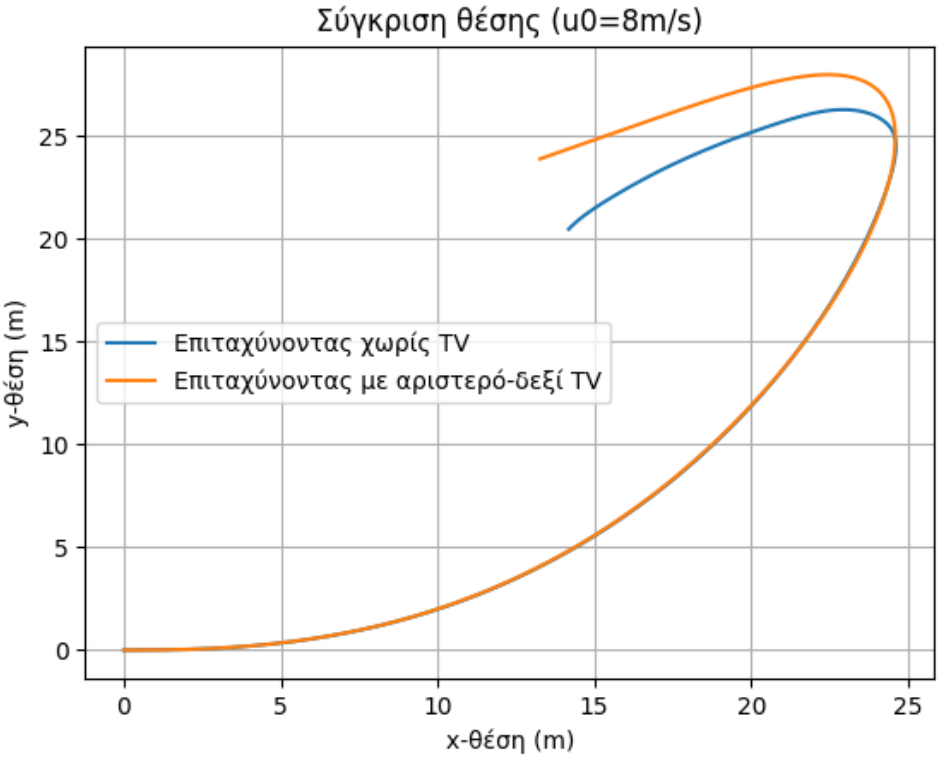
Για $v_0 = 4\,m/s$, στο Σχ.5.10 φαίνεται πως το σύστημα TV που αναπτύχθηκε κατέστησε το ρυθμό εκτροπής του οχήματος πολύ κοντά στον επιθυμητό, όταν άρχισε να παρατηρείται απόκλιση από τη χρονική στιγμή $t = 4\,s$ και έπειτα. Πιο συγκεκριμένα, το σύστημα διόρθωσε μία υπερστροφική συμπεριφορά, αφού ο ρυθμός εκτροπής είχε υπερβεί έως και κατά 1.5 rad/s τον επιθυμητό τη χρονική στιγμή $t = 6\,s$. Με το σύστημα ενεργό, η μέγιστη απόκλιση ήταν 0.4 rad/s, που σημαίνει ότι υπήρξε βελτίωση της τάξης του 73% τουλάχιστον. Ο αλγόριθμος μετέφερε περισσότερη ροπή στους αριστερούς τροχούς (εσωτερικούς), με αποτέλεσμα να ωθήσει το όχημα εξωτερικά της στροφής. Αυτό αποτυπώνεται στα Σχ. 5.12 και 5.11, όπου φαίνεται η πλατύτερη τροχιά και μείωση της γωνίας εκτροπής προς το τέλος του ελιγμού.



Σχήμα 5.13: Ρυθμός εκτροπής επιταχυνόμενης αριστερής στροφής με ρυθμό αύξησης γωνίας διεύθυνσης $v_\delta = 0.05\,rad/s$ και επιτάχυνση $a_{\text{long}} = 0.2g$

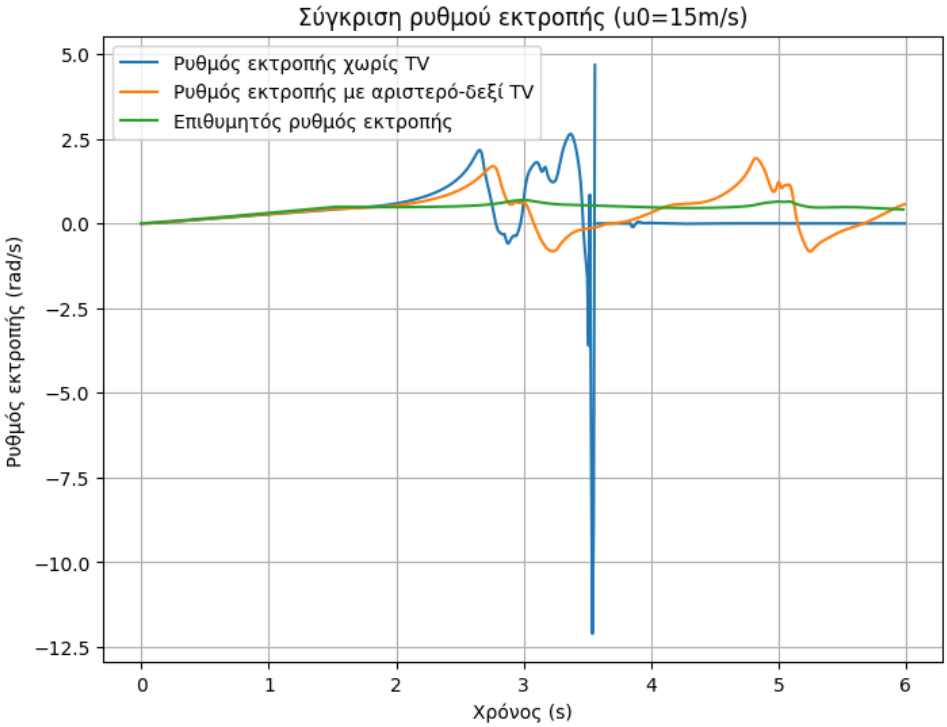


Σχήμα 5.14: Γωνία εκτροπής επιταχυνόμενη αριστερής στροφής με ρυθμό αύξησης γωνίας διεύθυνσης $v_\delta = 0.05\,rad/s$ και επιτάχυνση $a_{\text{long}} = 0.2g$

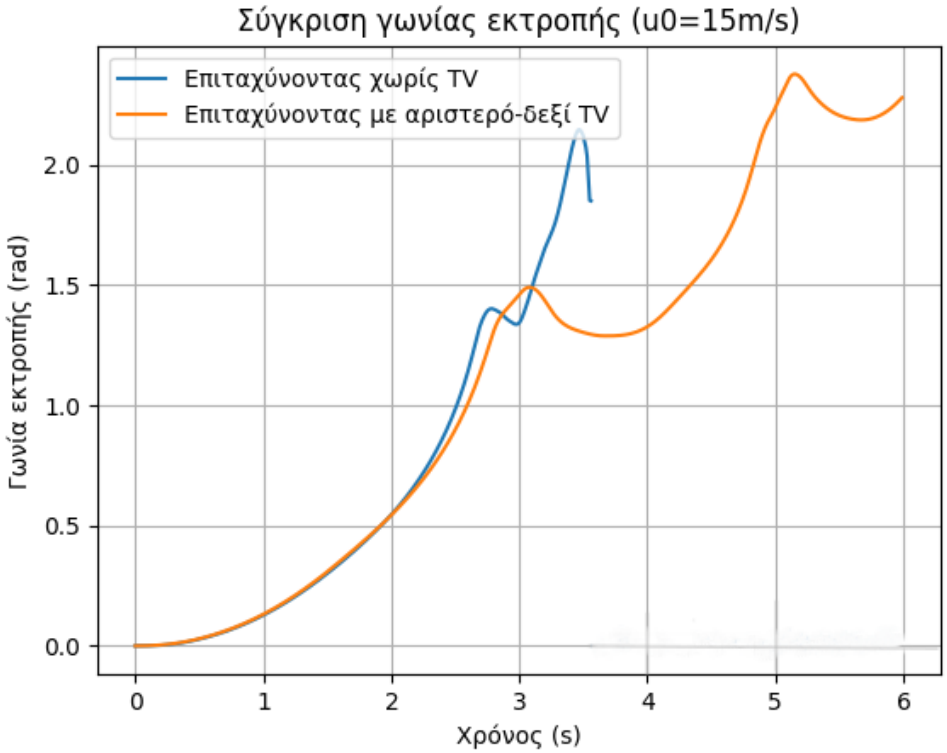


Σχήμα 5.15: Τροχιά επιταχυνόμενης αριστερής στροφής με ρυθμό αύξησης γωνίας διεύθυνσης $v_\delta = 0.05\,rad/s$ και επιτάχυνση $a_{\text{long}} = 0.2g$

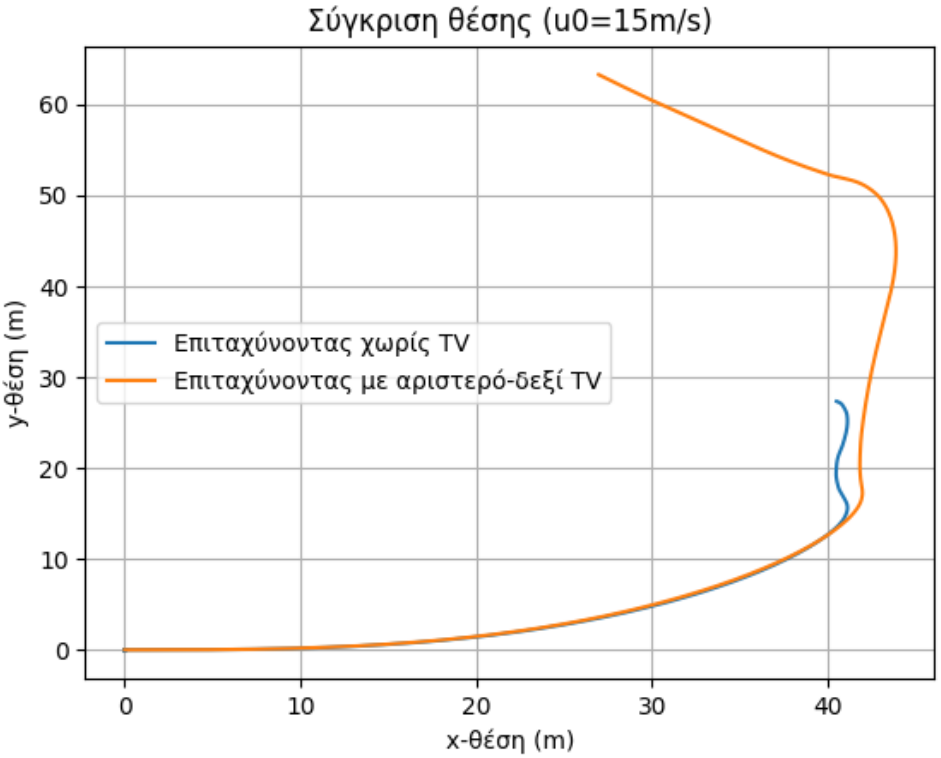
Για $v_0 = 8\,m/s$, στο Σχ.5.13 του ρυθμού εκτροπής παρατηρείται ότι το σύστημα κατ-απολέμησε τις διακυμάνσεις που μπορούσαν εύκολα να οδηγήσουν σε απώλεια ελέγχου. Αξιόλογη βελτίωση παρουσιάζεται και στο Σχ. 5.14 της γωνίας εκτροπής, όπου η καμπύλη τιμών είναι πιο ομαλή σε σχέση με τις αυξομειώσεις της καμπύλης χωρίς TV. Αυτό υποδηλώνει πως το όχημα δεν εκτελεί απότομες αλλαγές κατευθύνσεων, αποφεύγοντας επικίνδυνες καταστάσεις.



Σχήμα 5.16: Ρυθμός εκτροπής επιταχυνόμενης αριστερής στροφής με ρυθμό αύξησης γωνίας διεύθυνσης $v_\delta = 0.05\,rad/s$ και επιτάχυνση $a_{\text{long}} = 0.2g$



Σχήμα 5.17: Γωνία εκτροπής επιταχυνόμενης αριστερής στροφής με ρυθμό αύξησης γωνίας διεύθυνσης $v_\delta = 0.05\,rad/s$ και επιτάχυνση $a_{\text{long}} = 0.2g$

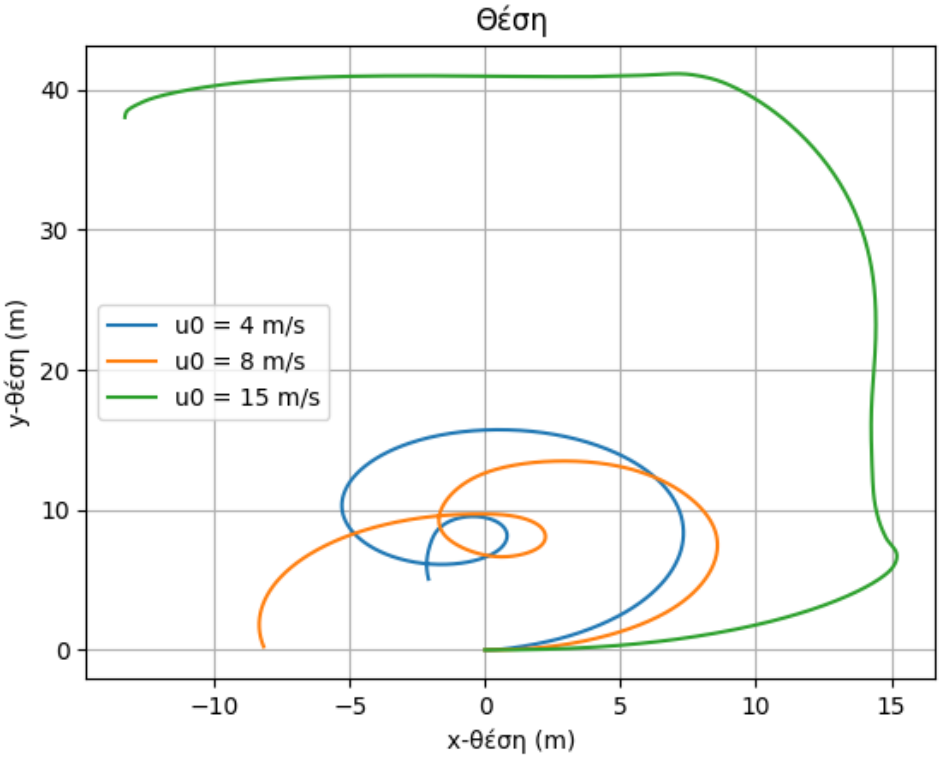


Σχήμα 5.18: Τροχιά επιταχυνόμενης αριστερής στροφής με ρυθμό αύξησης γωνίας διεύθυνσης $v_\delta = 0.05\,rad/s$ και επιτάχυνση $a_{\text{long}} = 0.2g$

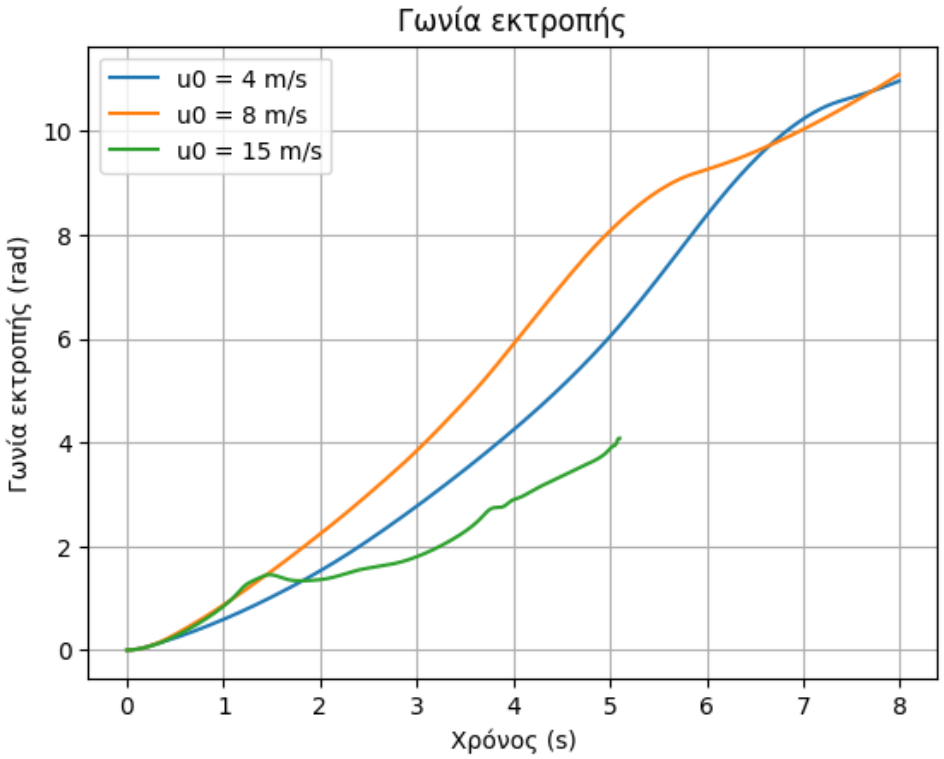
Από το Σχ. 5.16 του ρυθμού εκτροπής γίνεται φανερό πως το όχημα φέρει εις πέρας ολόκληρο τον ελιγμό διάρκειας $t = 6\,s$, παρουσιάζοντας πολύ πιο ελεγχόμενα φαινόμενα υπερστροφής και υποστροφής. Ο ρυθμός εκτροπής δεν ξεπερνά ποτέ τα $2\,\text{rad/s}$, παραμένοντας σε ασφαλή πλαίσια. Η επιτυχής ολοκλήρωση του ελιγμού επαληθεύεται και από το Σχ. 5.18, όπου φαίνεται η συνολική απόσταση που διένυσε το όχημα. Η επίδραση του συστήματος αποδείχθηκε σωτήρια στη συγκεκριμένη προσομοίωση, αφού απέτρεψε την απώλεια ελέγχου και πιθανό ατύχημα.

Step - steer maneuver

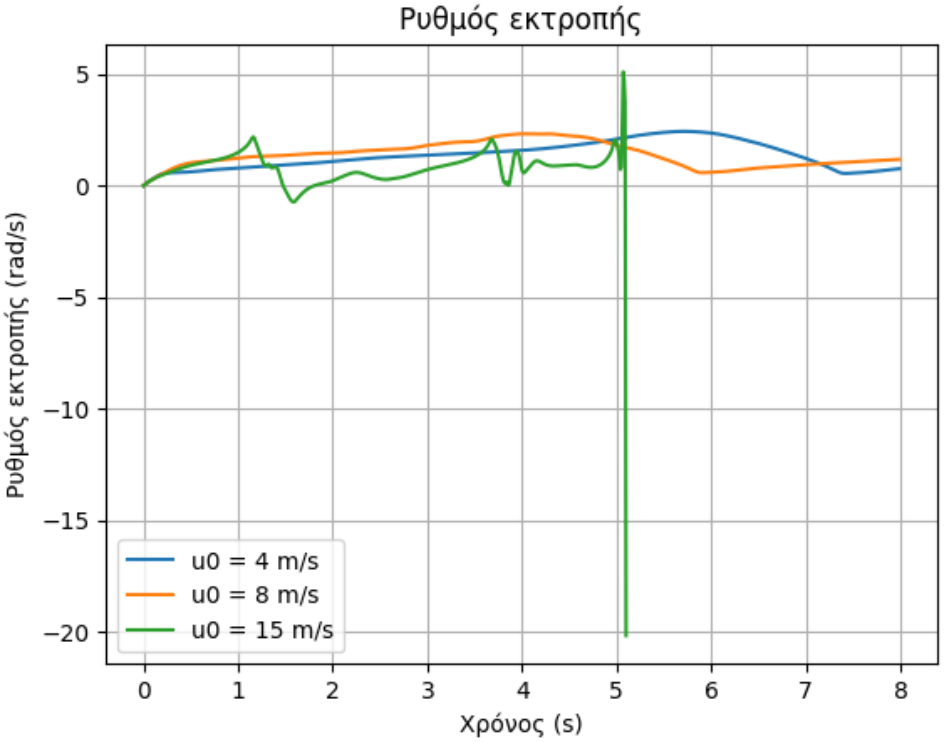
Στα Σχ. 5.19 - Σχ. 5.21 φαίνεται ο ελιγμός χωρίς την επίδραση του TV.



Σχήμα 5.19: Τροχιά ελιγμού step-steer με επιβολή ακαριαίας γωνίας διεύθυνσης $\delta = 0.314\,rad$ και επιτάχυνση $a_{\text{long}} = 0.3g$



Σχήμα 5.20: Γωνία εκτροπής ελιγμού step-steer με επιβολή ακαριαίας γωνίας διεύθυνσης $\delta = 0.314\,rad$ και επιτάχυνση $a_{\text{long}} = 0.3g$



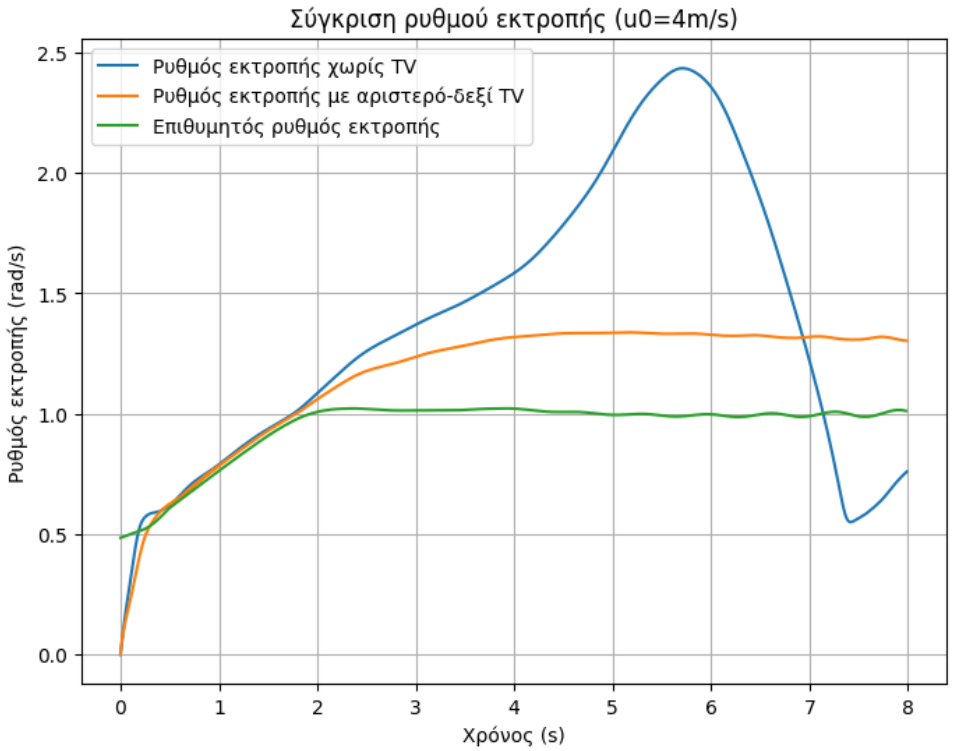
Σχήμα 5.21: Ρυθμός εκτροπής ελιγμού step-steer με επιβολή ακαριαίας γωνίας διεύθυνσης $\delta = 0.314\,rad$ και επιτάχυνση $a_{\text{long}} = 0.3g$

Για $v_0 = 4\,m/s$, στο Σχ. 5.21 παρατηρείται απότομη αύξηση του ρυθμού εκτροπής της τάξης των 0.5 rad/s από $t = 5\,s$ έως $t = 6\,s$ και ύστερα απότομη μείωση μεγέθους 2 rad/s, καθιστώντας την τιμή του πολύ χαμηλή. Αυτή είναι ένδειξη φαινομένου υπερστροφής, ακολουθούμενο από υποστροφή.

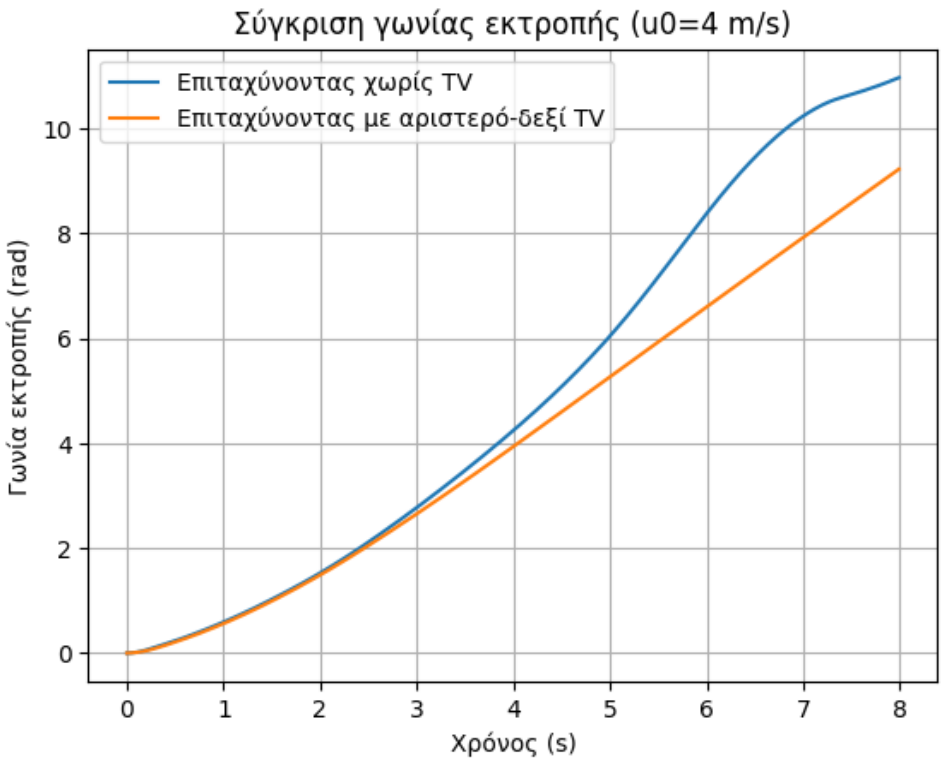
Ακριβώς το ίδιο φαινόμενο παρατηρείται για $v_0 = 8\,m/s$, αλλά αυτή τη φορά εμφανίζεται νωρίτερα, τη χρονική στιγμή $t = 3\,s$. Η πρόωρη εμφάνιση του φαινομένου οφείλεται στην υψηλότερη αρχική ταχύτητα του οχήματος. Η ταχύτητα που έχει το όχημα όταν παρουσιάζει αυτή τη συμπεριφορά επιτεύχθηκε νωρίτερα στη δεύτερη περίπτωση.

Για $v_0 = 15\,m/s$, οι υψηλές τιμές που λαμβάνει ο ρυθμός εκτροπής στο Σχ. 5.21 (έως και 20 rad/s) και η απώλειά τους αμέσως μετά τη χρονική στιγμή $t = 5\,s$ υποδηλώνει απώλεια ελέγχου και εκτροπή της πορείας του οχήματος ακριβώς εκείνη τη χρονική στιγμή. Το ίδιο επιβεβαιώνεται από το Σχ. 5.20, όπου η γωνία εκτροπής σταματάει να λαμβάνει τιμές για $t = 5\,s$ κι έπειτα.

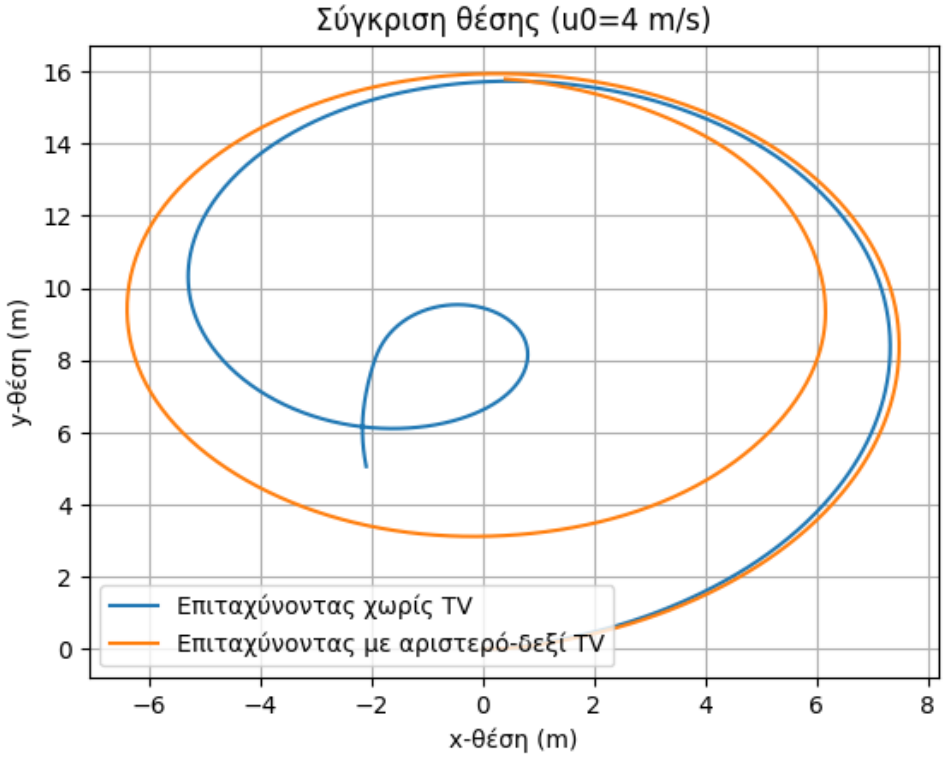
Στα Σχ.5.22 - Σχ.5.30 φαίνεται ο ελιγμός με το σύστημα κατανομής ροπής ενεργό.



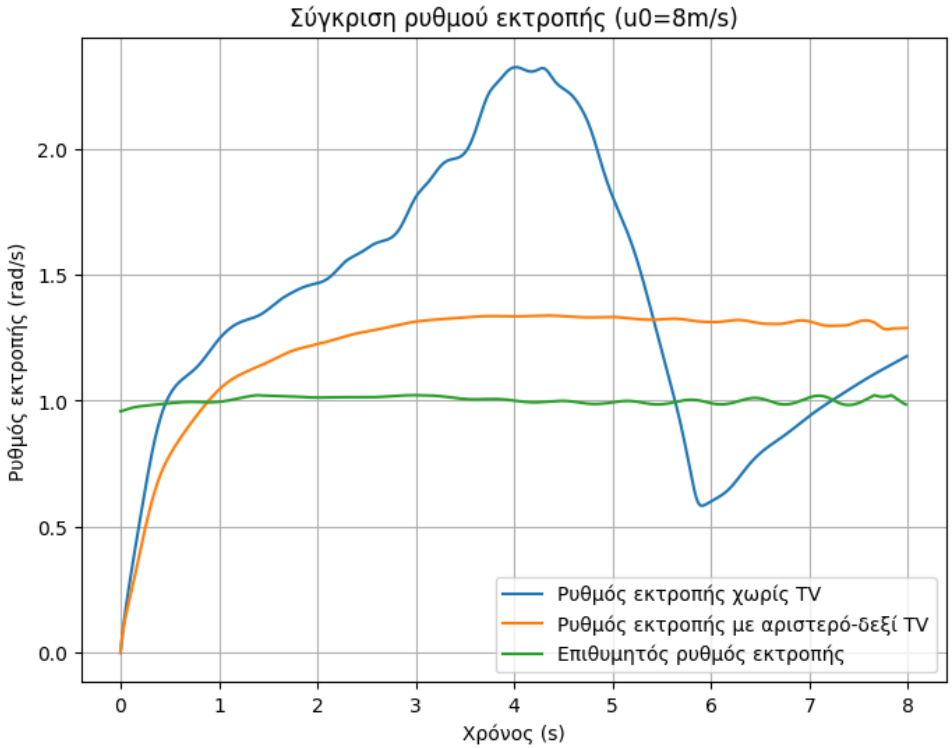
Σχήμα 5.22: Ρυθμός εκτροπής ελιγμού step-steer με επιβολή ακαριαίας γωνίας διεύθυνσης $\delta = 0.314\,rad$ και επιτάχυνση $a_{\text{long}} = 0.3g$



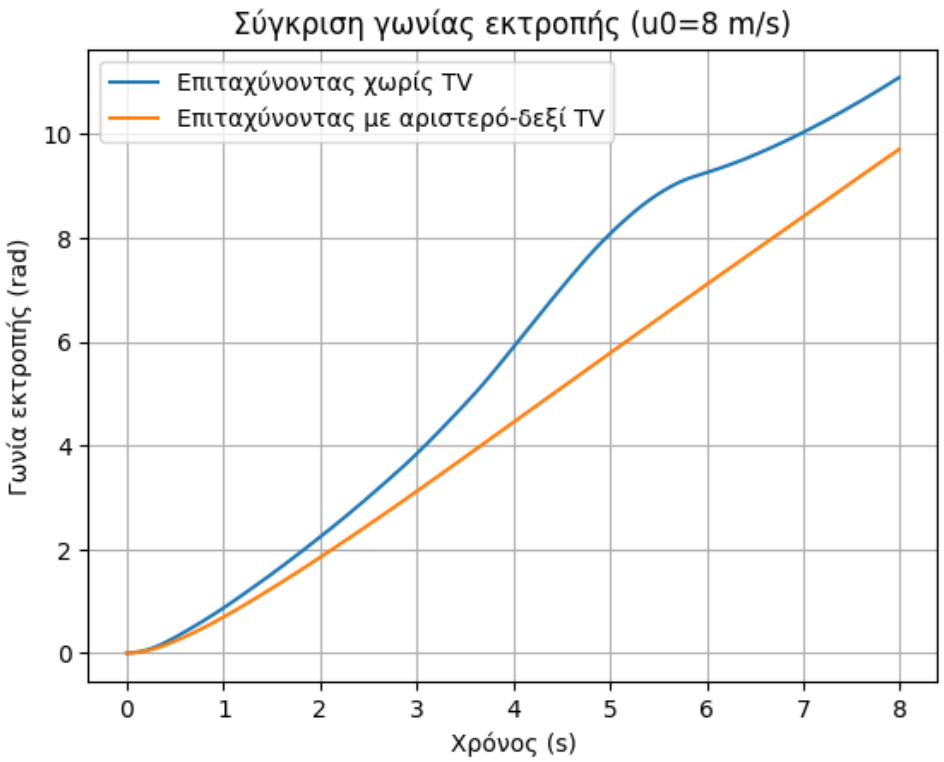
Σχήμα 5.23: Γωνία εκτροπής ελιγμού step-steer με επιβολή ακαριαίας γωνίας διεύθυνσης $\delta = 0.314\,rad$ και επιτάχυνση $a_{\text{long}} = 0.3g$



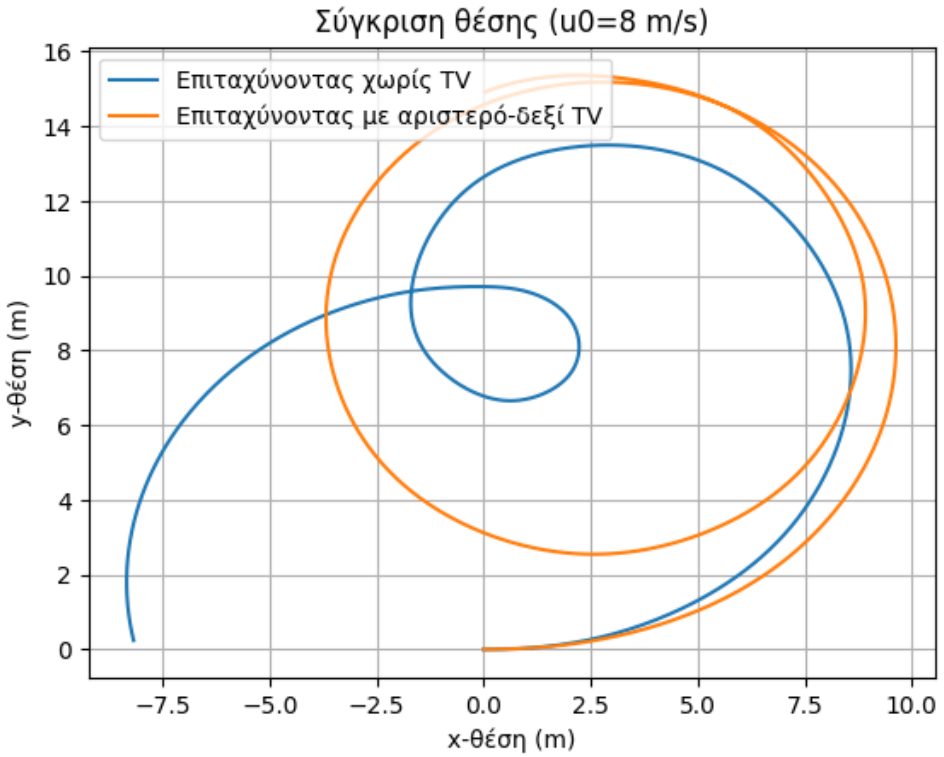
Σχήμα 5.24: Τροχιά ελιγμού step-steer με επιβολή ακαριαίας γωνίας διεύθυνσης $\delta = 0.314\,rad$ και επιτάχυνση $a_{\text{long}} = 0.3g$



Σχήμα 5.25: Ρυθμός εκτροπής ελιγμού step-steer με επιβολή ακαριαίας γωνίας διεύθυνσης $\delta = 0.314\,rad$ και επιτάχυνση $a_{\text{long}} = 0.3g$

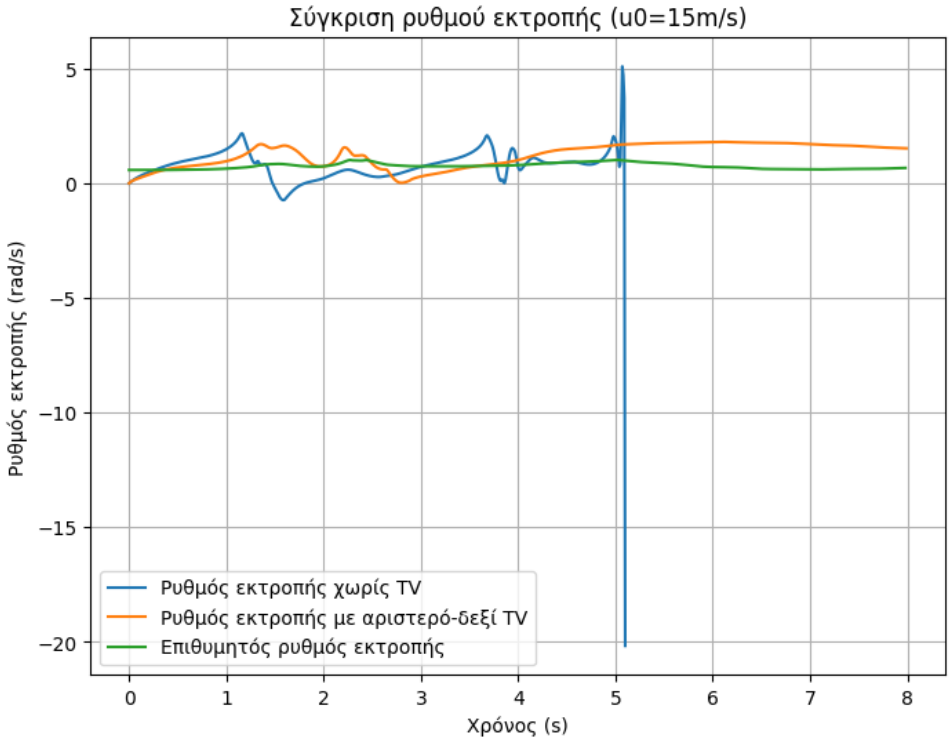


Σχήμα 5.26: Γωνία εκτροπής ελιγμού step-steer με επιβολή ακαριαίας γωνίας διεύθυνσης $\delta = 0.314\,rad$ και επιτάχυνση $a_{\text{long}} = 0.3g$

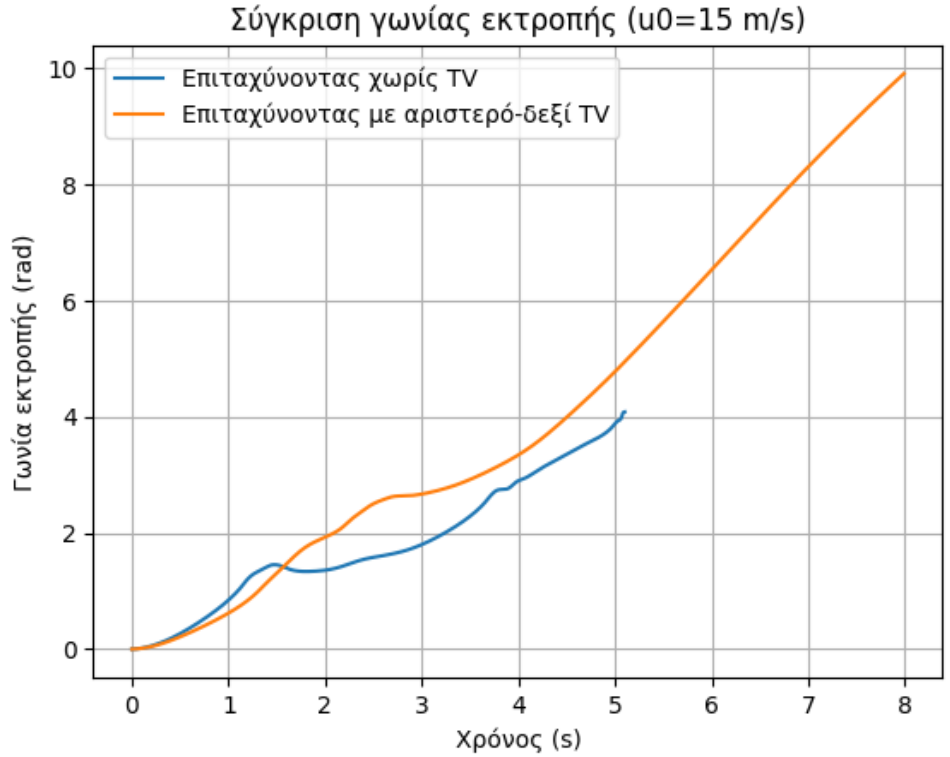


Σχήμα 5.27: Τροχιά ελιγμού step-steer με επιβολή ακαριαίας γωνίας διεύθυνσης $\delta = 0.314\,rad$ και επιτάχυνση $a_{\text{long}} = 0.3g$

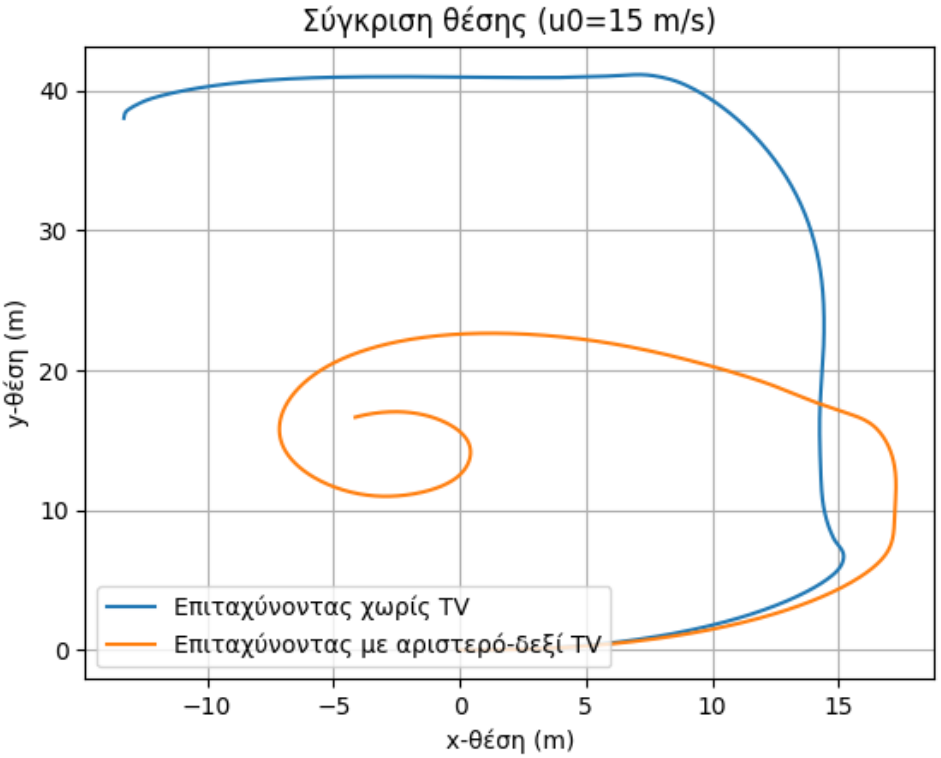
Για $v_0 = 4\,m/s$ & $8\,m/s$, το σύστημα που αναπτύχθηκε κατέστησε το ρυθμό εκτροπής του αυτοκινήτου πολύ κοντά στον επιθυμητό, όπως προκύπτει από τα Σχ.5.22 και 5.25. Τα φαινόμενα υπερστροφής ακολουθούμενα από υποστροφή που αναφέρθηκαν για τους ελιγμούς χωρίς την επίδραση TV προηγουμένως, διακρίνονται πολύ πιο καθαρά σε αυτά τα σχήματα. Με την επιβολή TV, τα φαινόμενα αυτά καταπολεμήθηκαν και ο ρυθμός εκτροπής σταθεροποιήθηκε σε σφάλμα λιγότερο του 0.5 rad/s και για τις 2 περιπτώσεις αρχικών ταχυτήτων. Συγκριτικά, κατά τη διάρκεια του ελιγμού με το σύστημα ανενεργό, το σφάλμα άγγιζε τα 1.5 rad/s, που σημαίνει πως υπήρξε έως και 66% βελτίωση. Τα σχήματα 5.23 και 5.26 της γωνίας εκτροπής αναδεικνύουν επίσης τη βελτιωμένη συμπεριφορά του οχήματος, αφού οι τιμές παρουσιάζουν σχεδόν γραμμική αύξηση (χωρίς διακυμάνσεις). Μελετώντας τα διαγράμματα της θέσης (Σχ. 5.24, 5.27) παρατηρείται τροχιά πολύ κοντά της αναμενόμενης, με την ακτίνα της κυκλικής διαδρομής που εκτελείται να μειώνεται με σταθερό ρυθμό, σε αντίθεση με την τροχιά που εκτέλεσε το όχημα με το TV ανενεργό. Σε κάθε περίπτωση, η απόκριση του αυτοκινήτου με το σύστημα ενεργό είναι αρκετά προβλέψιμη, εμπνέοντας ασφάλεια.



Σχήμα 5.28: Ρυθμός εκτροπής ελιγμού step-steer με επιβολή ακαριαίας γωνίας διεύθυνσης $\delta = 0.314\,rad$ και επιτάχυνση $a_{\text{long}} = 0.3g$



Σχήμα 5.29: Γωνία εκτροπής ελιγμού step-steer με επιβολή ακαριαίας γωνίας διεύθυνσης $\delta = 0.314\,rad$ και επιτάχυνση $a_{\text{long}} = 0.3g$

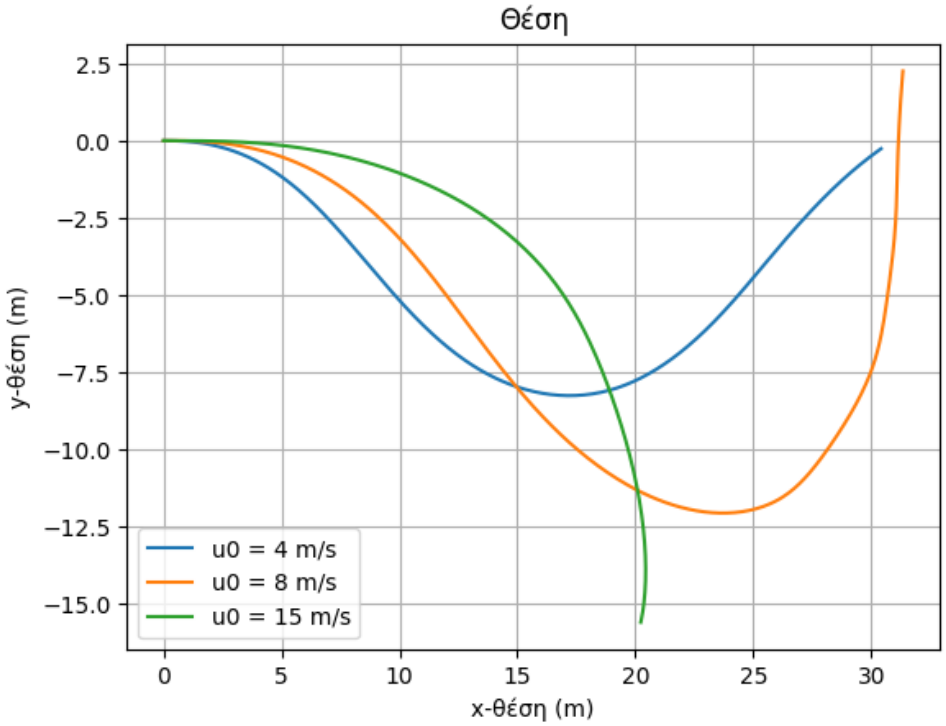


Σχήμα 5.30: Τροχιά ελιγμού step-steer με επιβολή ακαριαίας γωνίας διεύθυνσης $\delta = 0.314\,rad$ και επιτάχυνση $a_{\text{long}} = 0.3g$

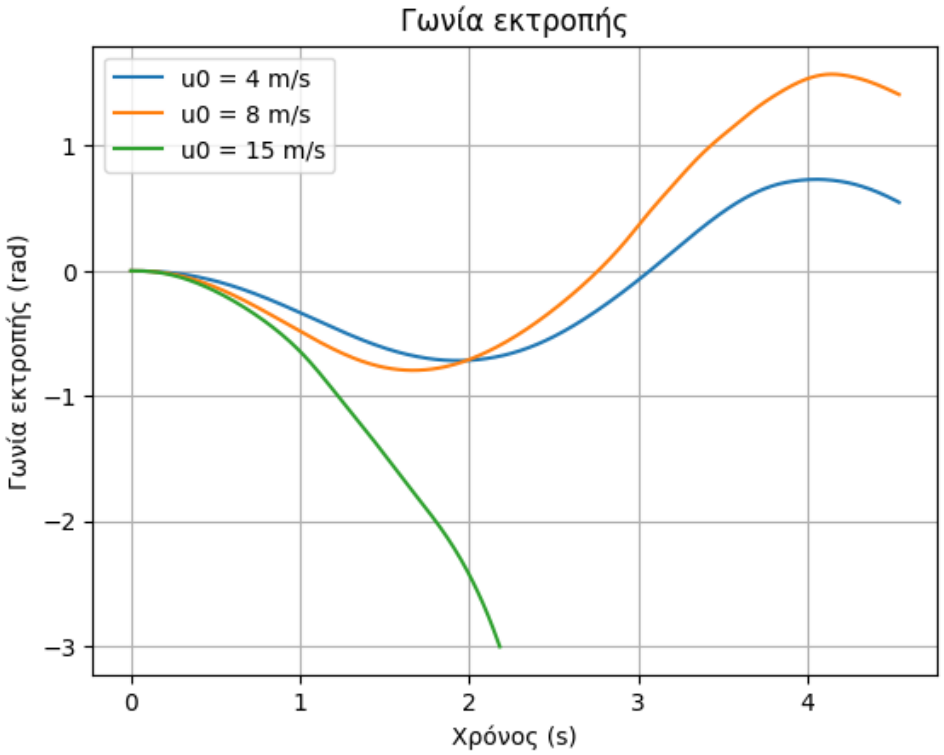
Για $v_0 = 15\,m/s$, το όχημα κατάφερε να ολοκληρώσει τον ελιγμό χωρίς απώλεια ελέγχου, όπως προκύπτει από από το Σχ. 5.28. Ο ρυθμός εκτροπής δεν υπερέβη τα 2 rad/s. Η τροχιά που ακολούθησε το όχημα (Σχ.5.30) ανταποκρίνεται με μεγαλύτερη ακρίβεια στις κατευθυντικές εντολές που έδωσε ο οδηγός, αν και παρουσιάζει αρκετή απόκλιση από την τροχιά με μικρότερες αρχικές ταχύτητες. Ωστόσο, η ικανότητα του συστήματος να αποτρέψει για άλλη μια φορά την απώλεια ελέγχου αναδεικνύει την αποτελεσματικότητα της τακτικής.

Ελιγμός αποφυγής εμποδίου

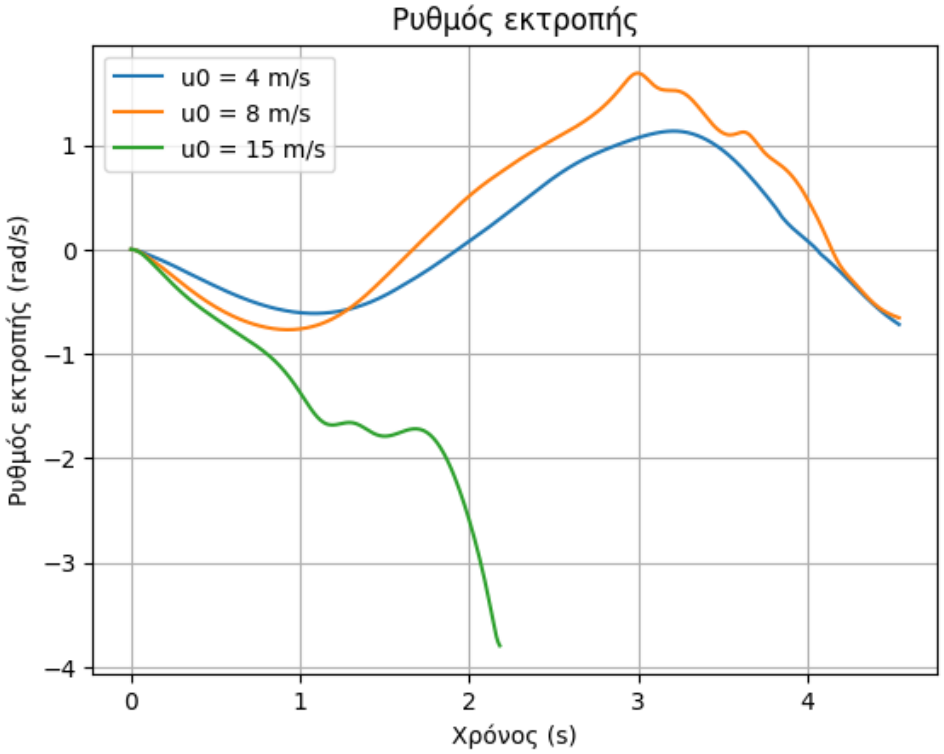
Στα Σχ. 5.31 - Σχ. 5.33 παρακάτω φαίνεται ο ελιγμός χωρίς την επίδραση του TV.



Σχήμα 5.31: Τροχιά ελιγμού αποφυγής εμποδίου με επιτάχυνση $a_{\text{long}} = 0.2g$



Σχήμα 5.32: Γωνία εκτροπής ελιγμού αποφυγής εμποδίου με επιτάχυνση $a_{\text{long}} = 0.2g$



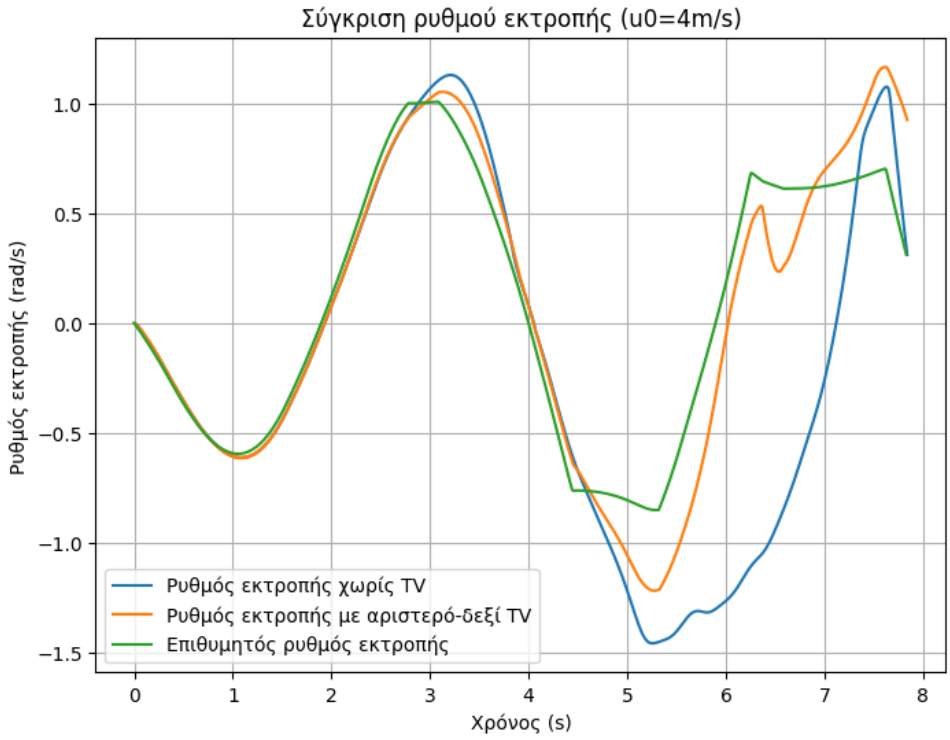
Σχήμα 5.33: Ρυθμός εκτροπής ελιγμού αποφυγής εμποδίου με επιτάχυνση $a_{\text{long}} = 0.2g$

Για αρχική ταχύτητα $v_0 = 4\,m/s$, ο ελιγμός ολοκληρώνεται με επιτυχία, όπως φαίνεται στο Σχ. 5.31. Το όχημα αποφεύγει το εμπόδιο και επιστρέφει ασφαλώς στη λωρίδα του. Ο ρυθμός και η γωνία εκτροπής (Σχ. 5.33 και 5.32) παρουσιάζουν ομαλές μεταβολές, ανάλογες της τροχιάς του αυτοκινήτου.

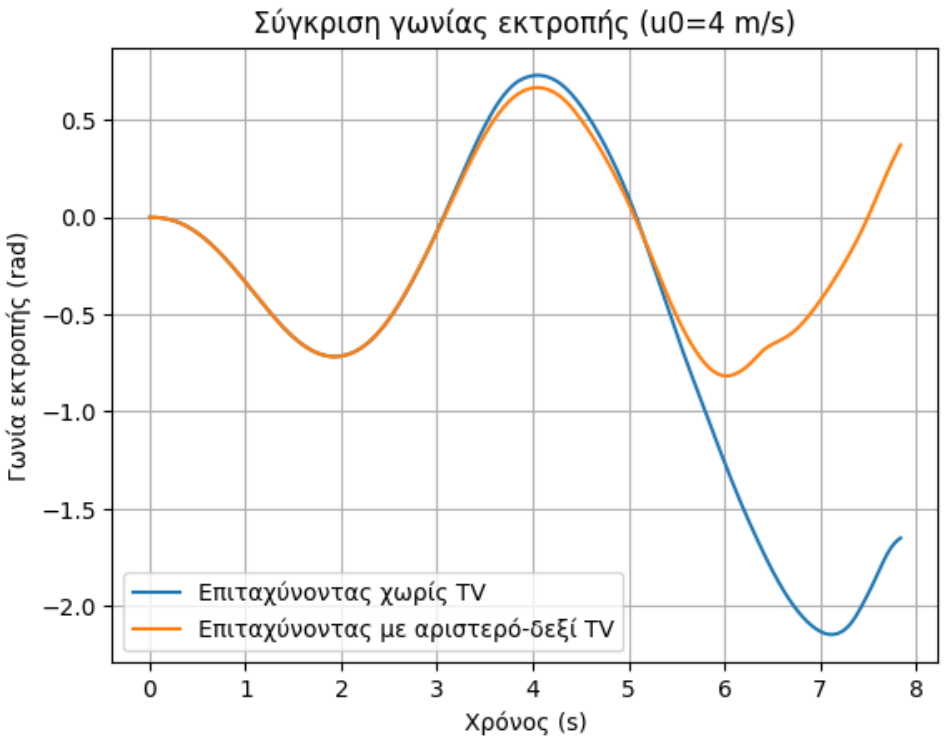
Για $v_0 = 8\,m/s$, κατά τη διάρκεια της αριστερής στροφής προς επιστροφή στη λωρίδα, το όχημα εμφανίζει υπερστροφή. Αυτό γίνεται φανερό από το Σχ. 5.33 τη χρονική στιγμή $t = 3\,s$, όπου παρατηρείται αιχμή στις τιμές του ρυθμού εκτροπής και μετά ανεπιθύμητες διακυμάνσεις. Η απόκριση αυτή είναι επικίνδυνη καθώς το όχημα εκτελεί μεγαλύτερη στροφή προς τα αριστερά από το επιθυμητό, με κίνδυνο να βγει εκτός δρόμου εάν ο οδηγός δεν εισάγει τις απαιτούμενες διορθώσεις.

Όπως και στους δύο προηγούμενους ελιγμούς, για αρχική ταχύτητα $v_0 = 15\,m/s$ το όχημα παρουσιάζει απώλεια ελέγχου και εκτρέπεται της πορείας του λίγο μετά τη χρονική στιγμή $t = 2\,s$, αφού ο ρυθμός εκτροπής αυξήθηκε κατά 2 rad/s σχεδόν ακαριαία.

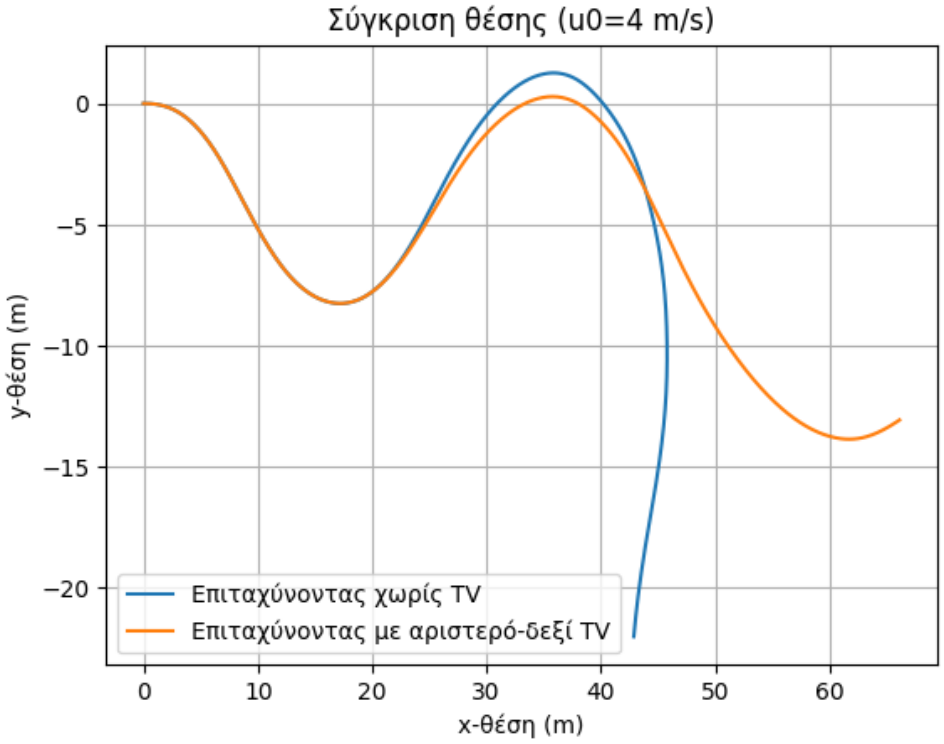
Στα Σχ.5.34 - Σχ.5.42 φαίνεται ο ελιγμός με το σύστημα κατανομής ροπής ενεργό.



Σχήμα 5.34: Ρυθμός εκτροπής ελιγμού αποφυγής εμποδίου με επιτάχυνση $a_{\text{long}} = 0.2g$



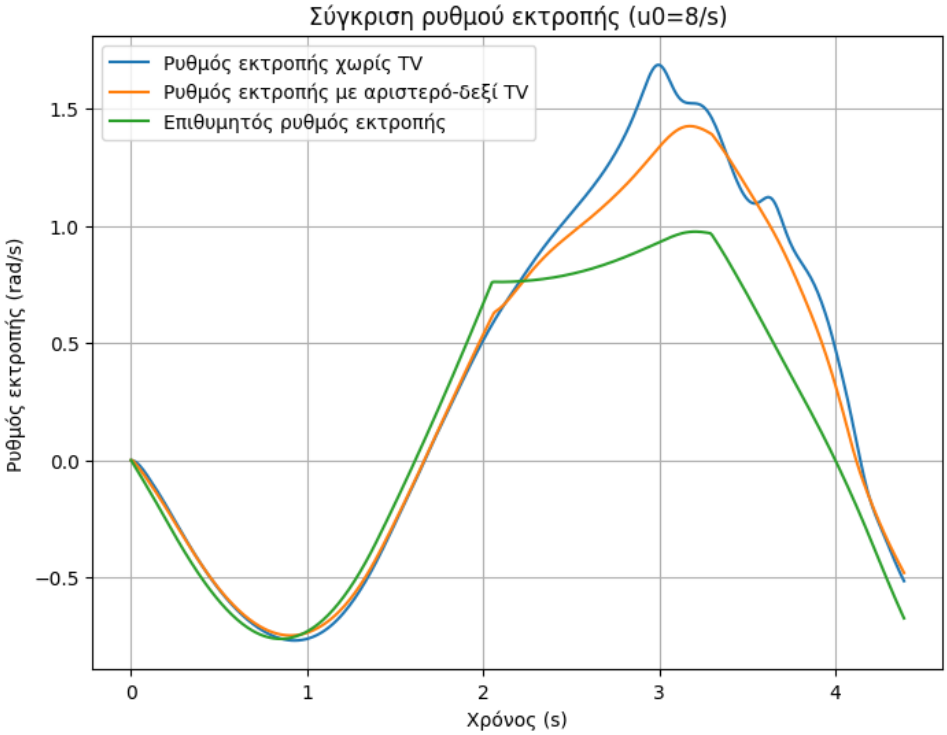
Σχήμα 5.35: Γωνία εκτροπής ελιγμού αποφυγής εμποδίου με επιτάχυνση $a_{\text{long}} = 0.2g$



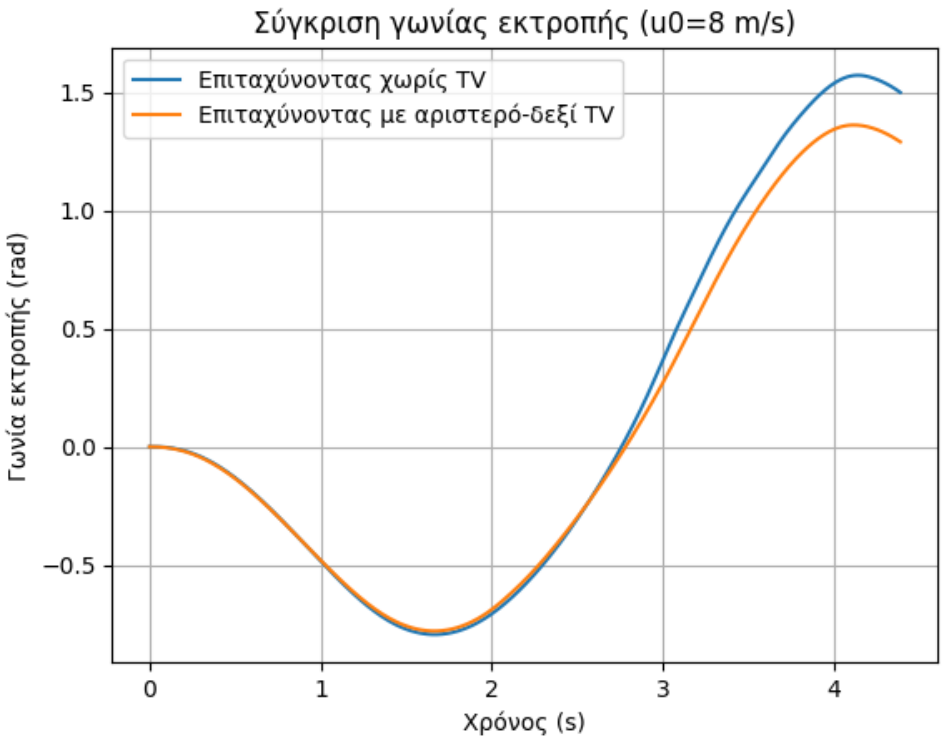
Σχήμα 5.36: Τροχιά ελιγμού αποφυγής εμποδίου με επιτάχυνση $a_{\text{long}} = 0.2g$

Για $v_0 = 4\,m/s$, ο ελιγμός συνεχίστηκε και αφότου το όχημα επέστρεψε στη λωρίδα του, όπως φαίνεται στο διάγραμμα τροχιάς (Σχ.5.36). Καθώς το όχημα έστριψε για να επαναλάβει

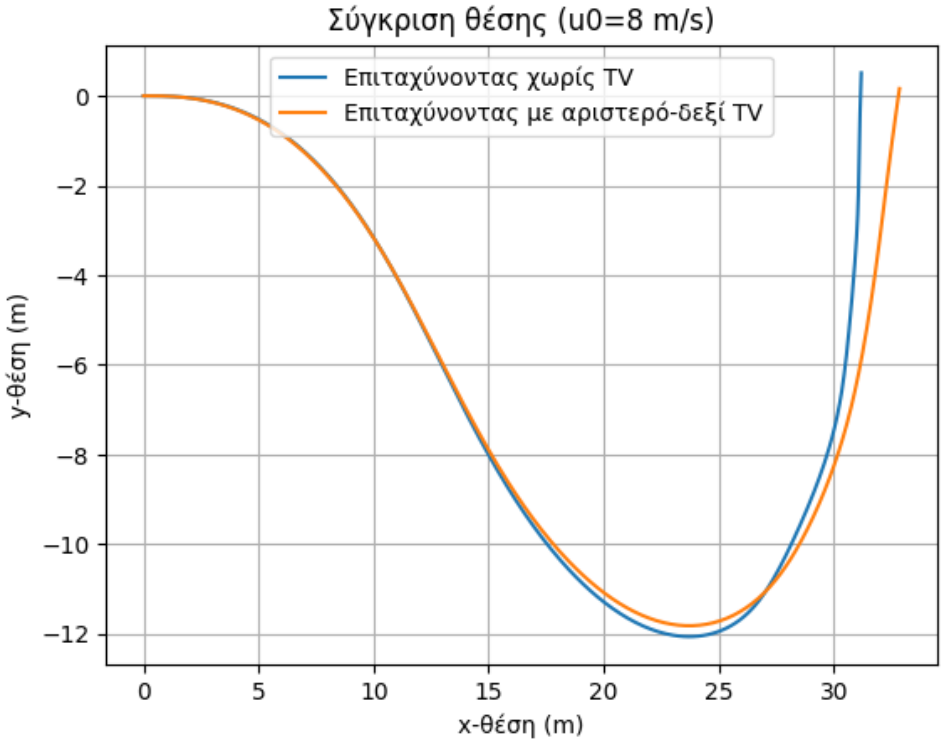
τον ίδιο ελιγμό, αυτή τη φορά προφανώς με υψηλότερη ταχύτητα, παρουσιάστηκε έντονο φαινόμενο υπερστροφής. Αυτό επαληθεύεται και από το Σχ. 5.34 του ρυθμού εκτροπής, λίγο μετά τη χρονική στιγμή $t = 5\,s$, όπου οι τιμές σχηματίζουν κορυφή. Με το σύστημα TV ενεργό, το όχημα δεν παρουσιάζει το έντονο φαινόμενο υπερστροφής και ακολουθεί την επιθυμητή πορεία κατά την εκτέλεση του ελιγμού για δεύτερη φορά. Η μεγαλύτερη βελτίωση κατά τον συγκεκριμένο ελιγμό παρατηρείται τη χρονική στιγμή $t = 6.2\,s$, όπου ο ρυθμός εκτροπής υπό την επίδραση του TV απέχει λιγότερο από 0.2 rad/s από τον επιθυμητό, ενώ χωρίς TV το σφάλμα ήταν 1.7 rad/s. Αυτό μεταφράζεται σε βελτίωση της τάξης του 88%.



Σχήμα 5.37: Ρυθμός εκτροπής ελιγμού αποφυγής εμποδίου με επιτάχυνση $a_{\text{long}} = 0.2g$



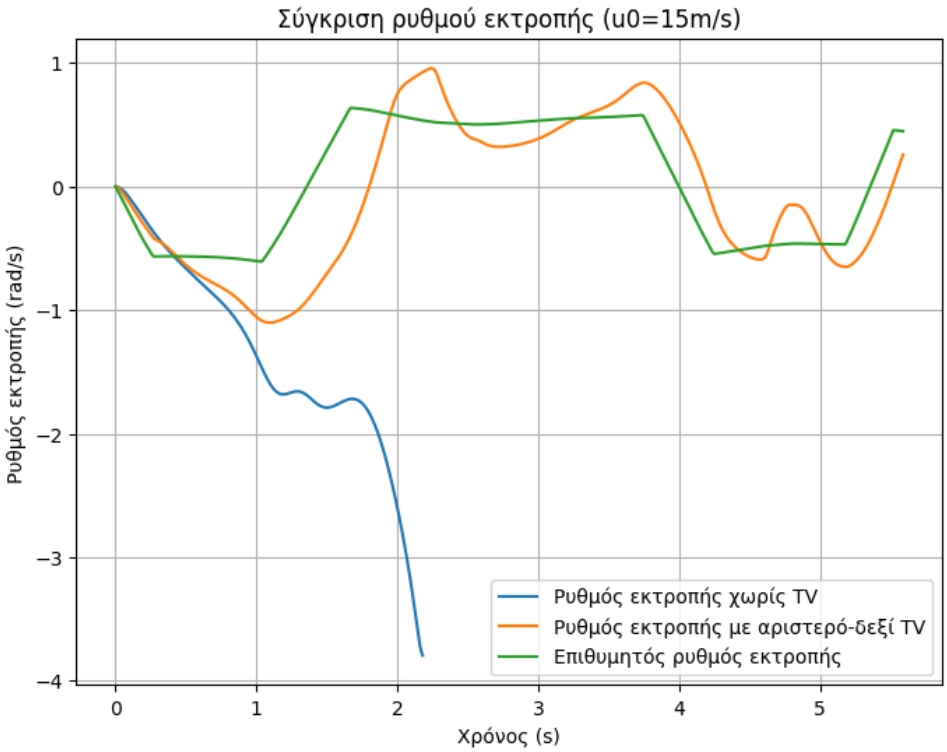
Σχήμα 5.38: Γωνία εκτροπής ελιγμού αποφυγής εμποδίου με επιτάχυνση $a_{\text{long}} = 0.2g$



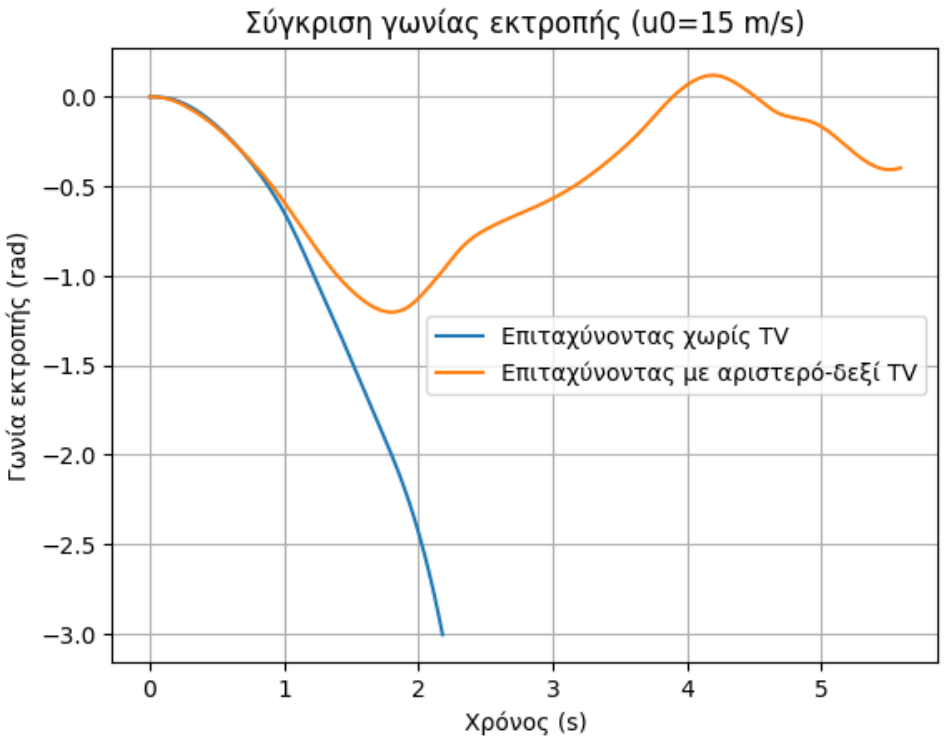
Σχήμα 5.39: Τροχιά ελιγμού αποφυγής εμποδίου με επιτάχυνση $a_{\text{long}} = 0.2g$

Για $v_0 = 8\,m/s$, διορθώνεται μια έντονη και επικίνδυνη υπερστροφική συμπεριφορά τη χρονική στιγμή $t = 3\,s$ του ελιγμού. Πιο συγκεκριμένα, στο Σχ. 5.37 του ρυθμού εκ-

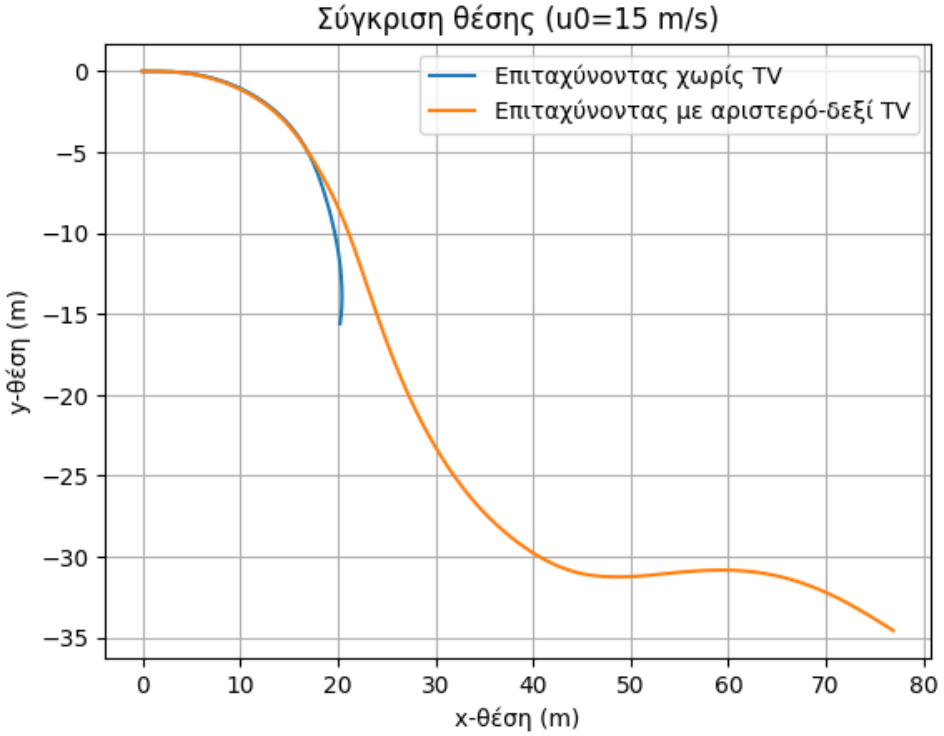
τροπής είναι ορατή η εξομάλυνση της κορυφής στο σημείο $t = 3\,s$. Αυτή η εξομάλυνση μεταφράζεται στο Σχ. 5.39 της θέσης σε άμβλυνση της τροχιάς, γεγονός που επιτρέπει στον οδηγό να εισάγει μικρότερη διόρθωση, μειώνοντας τις πιθανότητες ατυχήματος.



Σχήμα 5.40: Ρυθμός εκτροπής ελιγμού αποφυγής εμποδίου με επιτάχυνση $a_{\text{long}} = 0.2g$



Σχήμα 5.41: Γωνία εκτροπής ελιγμού αποφυγής εμποδίου με επιτάχυνση $a_{\text{long}} = 0.2g$



Σχήμα 5.42: Τροχιά ελιγμού αποφυγής εμποδίου με επιτάχυνση $a_{\text{long}} = 0.2g$

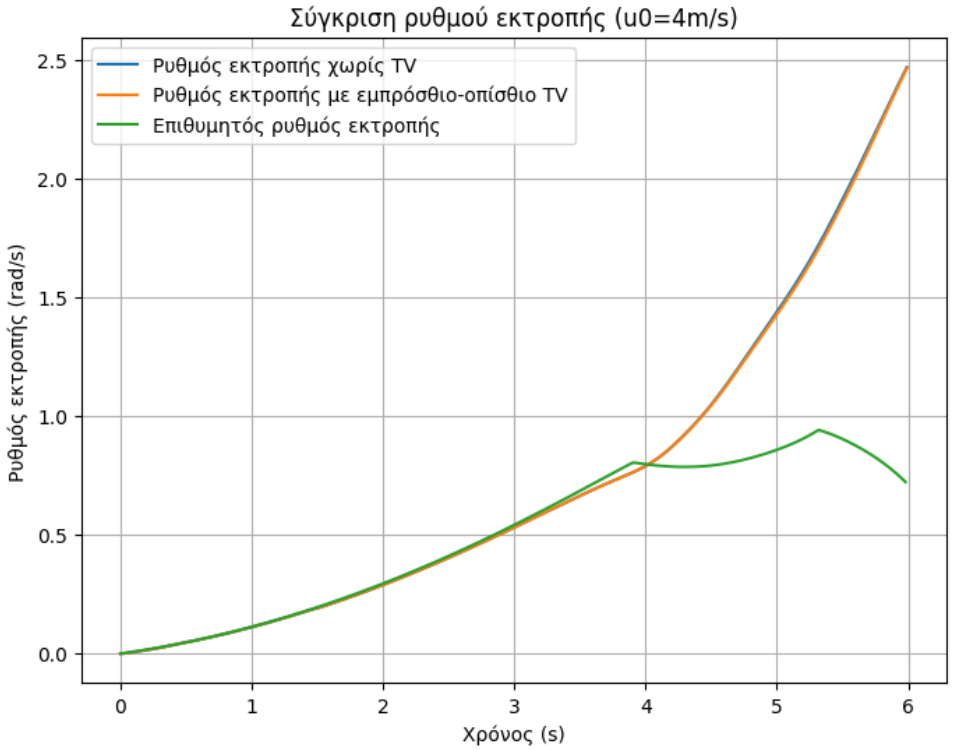
Για την ίδια δοκιμή με την υψηλότερη αρχική ταχύτητα ($v_0 = 15\,m/s$), αποφεύγεται η εκτροπή του αυτοκινήτου τη χρονική στιγμή $t = 2.2\,s$ και ύστερα ο ρυθμός εκτροπής

ακολουθεί με αρκετή ακρίβεια τον επιθυμητό, παρουσιάζοντας μέγιστη απόκλιση 1 rad/s για $t = 1.3\,s$, όπως φαίνεται στο Σχ. 5.40. Την ίδια χρονική στιγμή, το όχημα χωρίς TV παρουσίαζε σφάλμα μεγέθους 1.7 rad/s, το οποίο αυξανόταν διαρκώς. Ωστόσο, λόγω της υψηλής ταχύτητας κατά τη διάρκεια του ελιγμού, το σύστημα δεν κατάφερε να εκτελέσει τη διόρθωση σε βαθμό που να επιτρέψει στο όχημα να ολοκληρώσει τη δοκιμή με επιτυχία. Όπως φαίνεται και στο Σχ.5.42 που αποτυπώνεται η τροχιά, το όχημα δεν απέκτησε τον επιθυμητό ρυθμό εκτροπής κατά την αριστερή στροφή προκειμένου να επιστρέψει στη λωρίδα του. Στην περίπτωση αυτή, ο οδηγός οφείλει να εισάγει διορθωτική στροφή τιμονιού προς τα αριστερά.

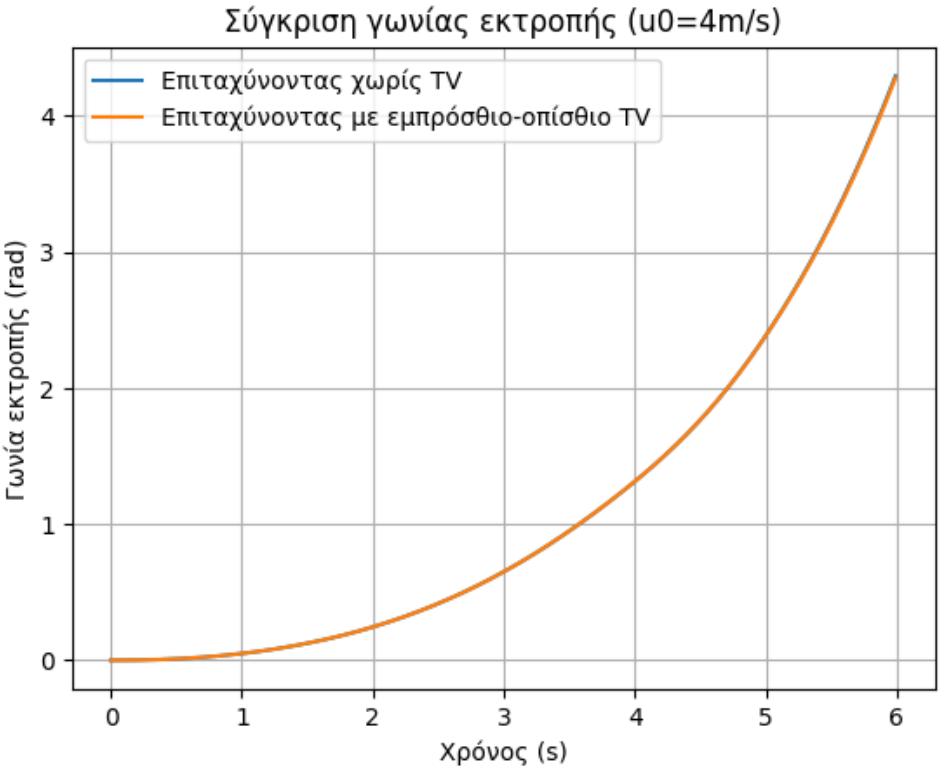
Συνοψίζοντας, η εφαρμογή συστήματος αριστερού-δεξιού TV στο όχημα αποδείχτηκε πολύ αποδοτική, συμβάλλοντας στη βελτίωση του κρατήματός του κατά τη διάρκεια των ελιγμών. Σημαντικότερα όμως, απέτρεψε κάθε περίπτωση απώλειας ελέγχου και εκτροπής του οχήματος από την πορεία του, καθιστώντας το ασφαλέστερο.

5.2 Εμπρόσθιο - οπίσθιο TV

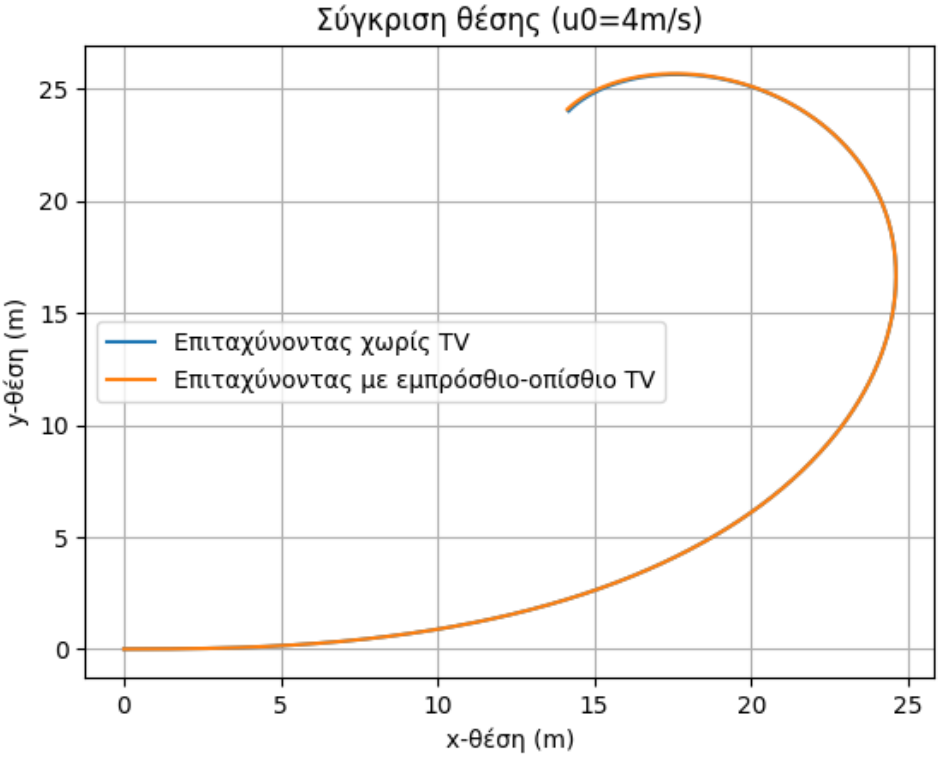
Τα Σχ.5.43 - 5.51, 5.52 - 5.60 και 5.61 - 5.69 που φαίνονται αμέσως μετά παραθέτουν τα αποτελέσματα επιβολής μόνο εμπρόσθιου-οπίσθιου TV στους ελιγμούς επιταχυνόμενης αριστερής στροφής, step-steer και αποφυγής εμποδίου αντιστοίχως. Λόγω του ότι τα αποτελέσματα παρουσιάζουν ξεκάθαρο μοτίβο και προς αποφυγή φλυαρίας, προτείνεται μελέτη των παρακάτω σχημάτων χωρίς επεμβάσεις και αναλυτικός σχολιασμός στο τέλος.



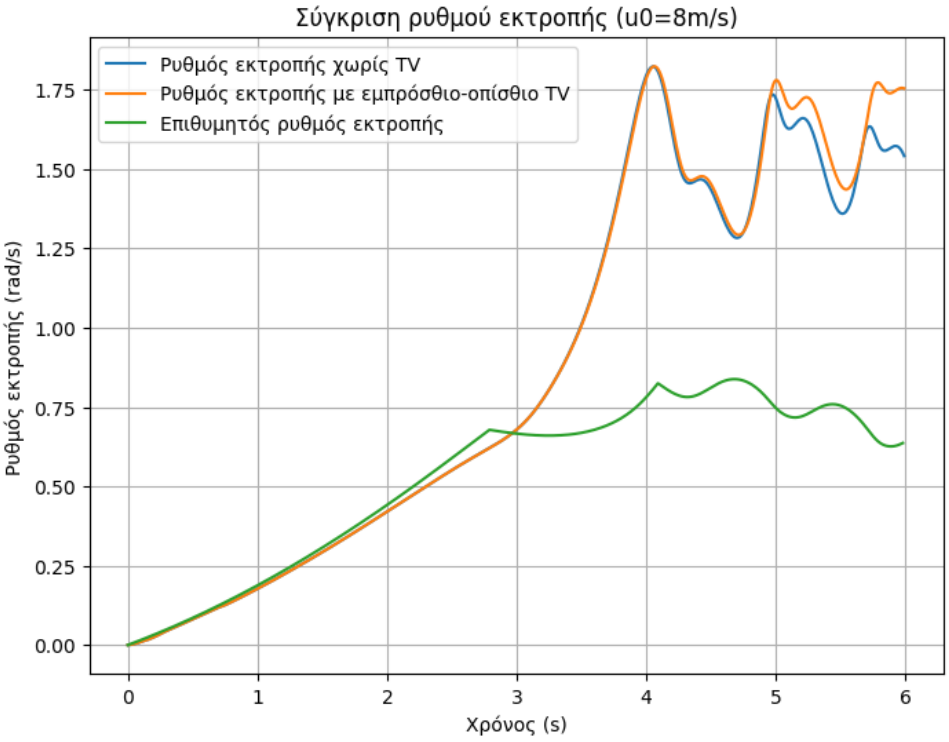
Σχήμα 5.43: Ρυθμός εκτροπής επιταχυνόμενης αριστερής στροφής με ρυθμό αύξησης γωνίας διεύθυνσης $v_\delta = 0.05\,rad/s$ και επιτάχυνση $a_{\text{long}} = 0.2g$



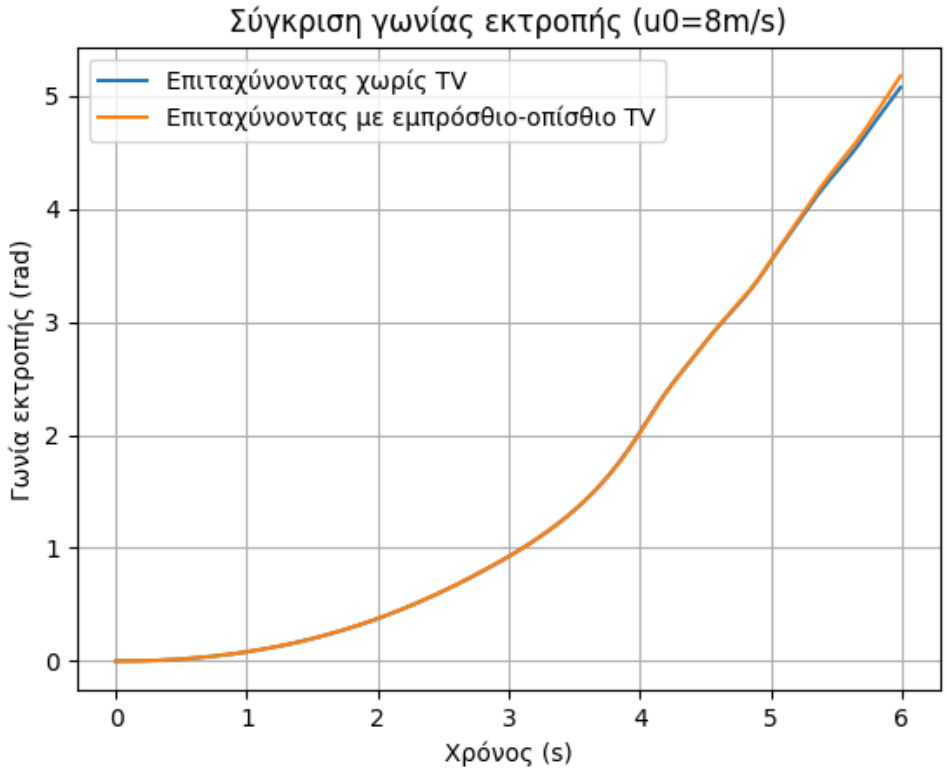
Σχήμα 5.44: Γωνία εκτροπής επιταχυνόμενης αριστερής στροφής με ρυθμό αύξησης γωνίας διεύθυνσης $v_\delta = 0.05\,rad/s$ και επιτάχυνση $a_{\text{long}} = 0.2g$



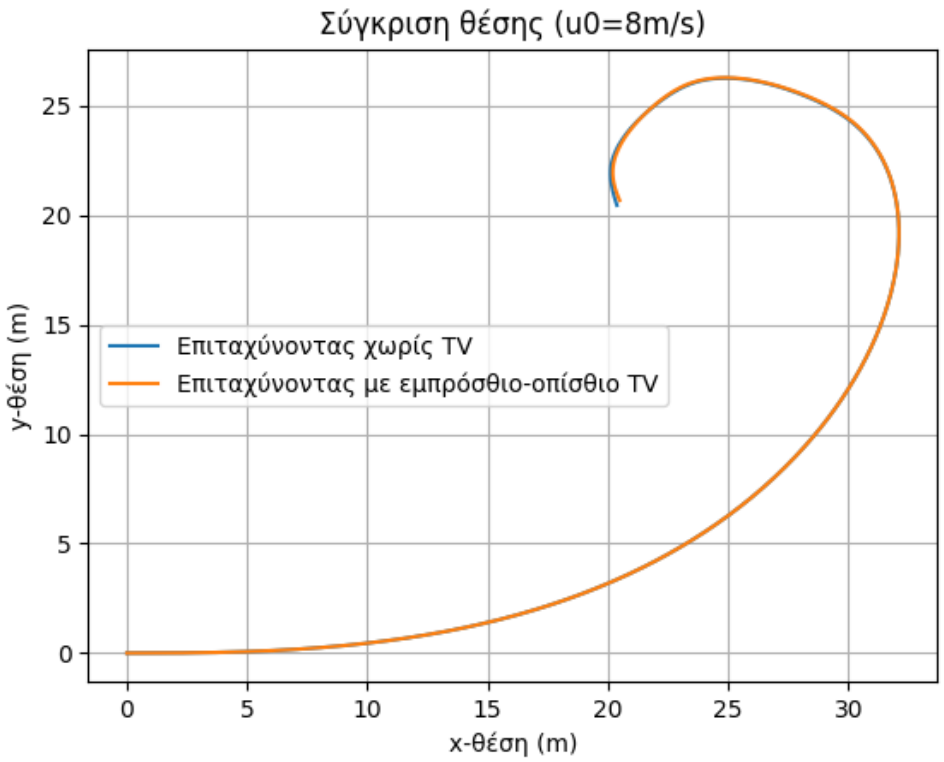
Σχήμα 5.45: Τροχιά επιταχυνόμενης αριστερής στροφής με ρυθμό αύξησης γωνίας διεύθυνσης $v_\delta = 0.05\,rad/s$ και επιτάχυνση $a_{\text{long}} = 0.2g$



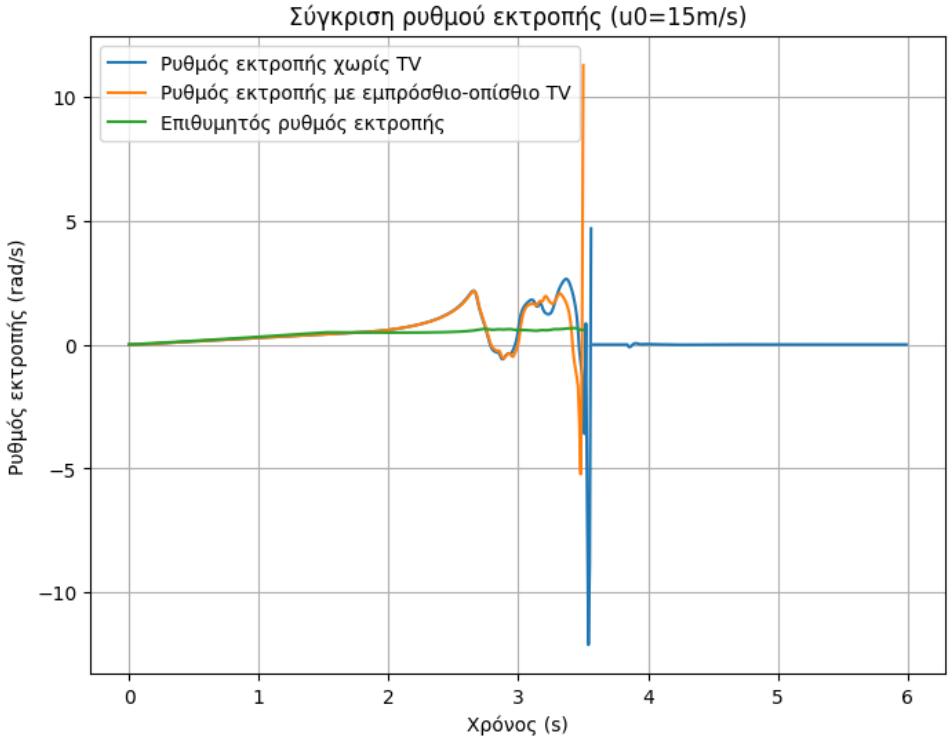
Σχήμα 5.46: Ρυθμός εκτροπής επιταχυνόμενης αριστερής στροφής με ρυθμό αύξησης γωνίας διεύθυνσης $v_\delta = 0.05\,rad/s$ και επιτάχυνση $a_{\text{long}} = 0.2g$



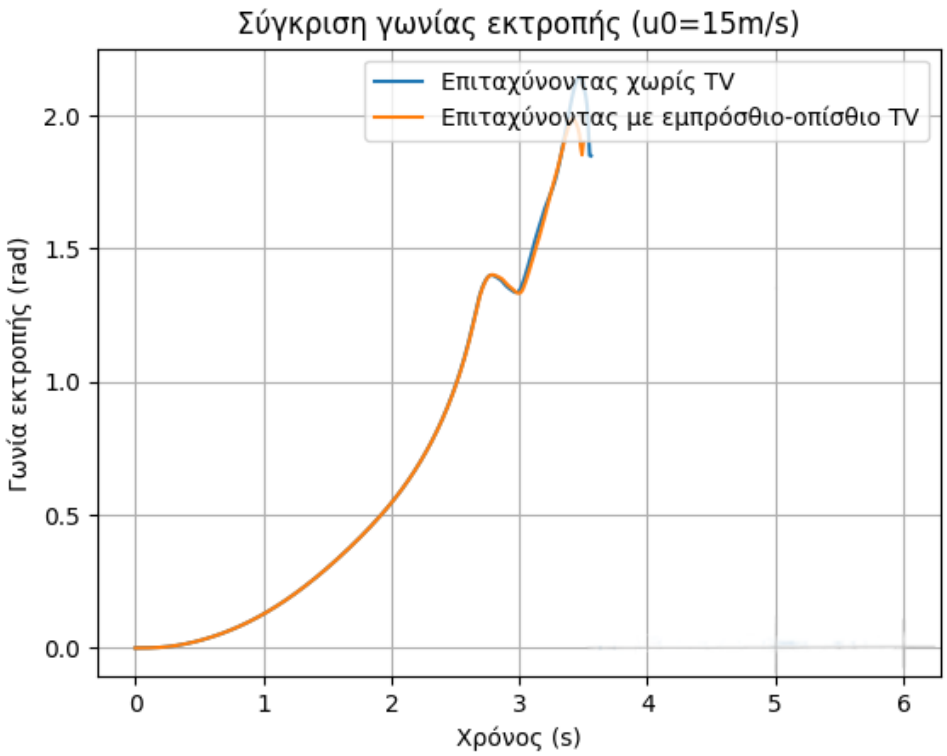
Σχήμα 5.47: Γωνία εκτροπής επιταχυνόμενης αριστερής στροφής με ρυθμό αύξησης γωνίας διεύθυνσης $v_\delta = 0.05\,rad/s$ και επιτάχυνση $a_{\text{long}} = 0.2g$



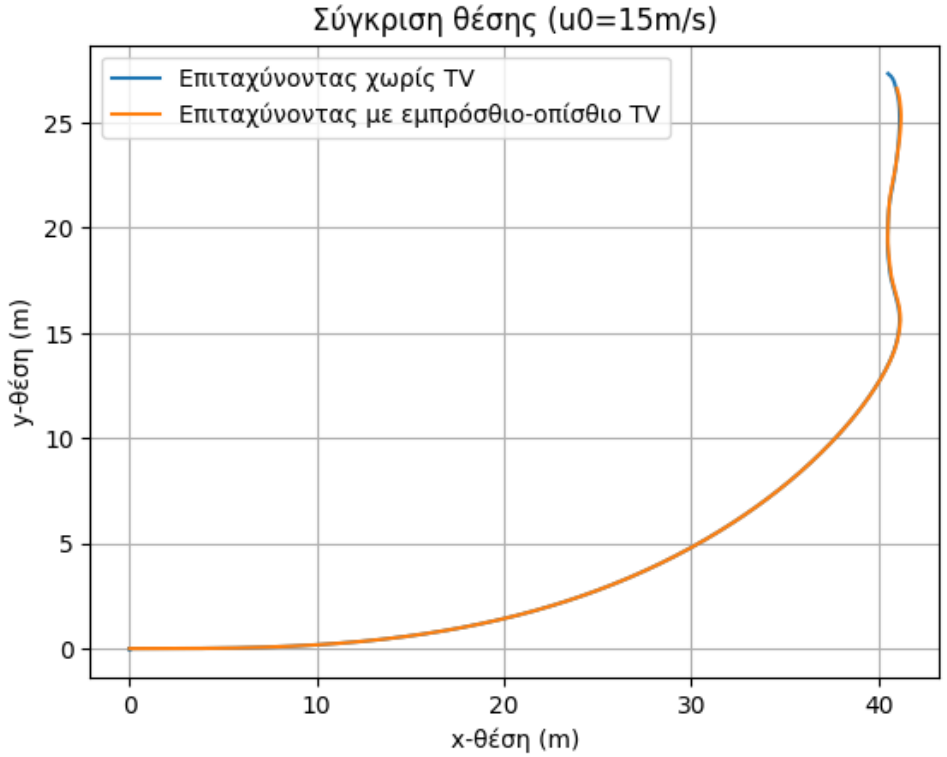
Σχήμα 5.48: Τροχιά επιταχυνόμενης αριστερής στροφής με ρυθμό αύξησης γωνίας διεύθυνσης $v_\delta = 0.05\,rad/s$ και επιτάχυνση $a_{\text{long}} = 0.2g$



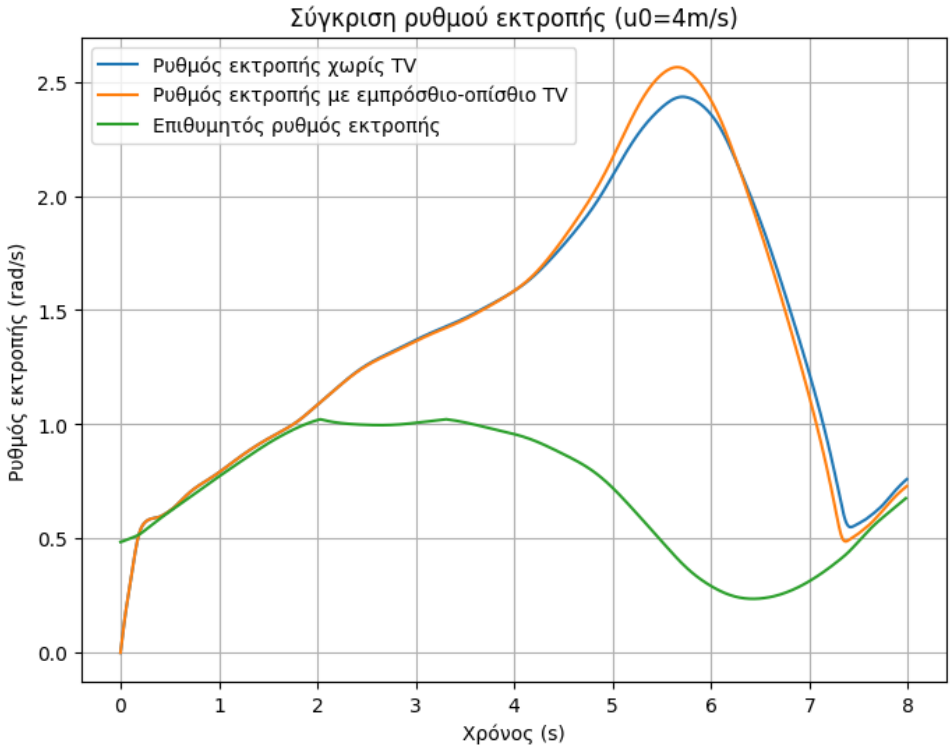
Σχήμα 5.49: Ρυθμός εκτροπής επιταχυνόμενης αριστερής στροφής με ρυθμό αύξησης γωνίας διεύθυνσης $v_\delta = 0.05\,rad/s$ και επιτάχυνση $a_{\text{long}} = 0.2g$



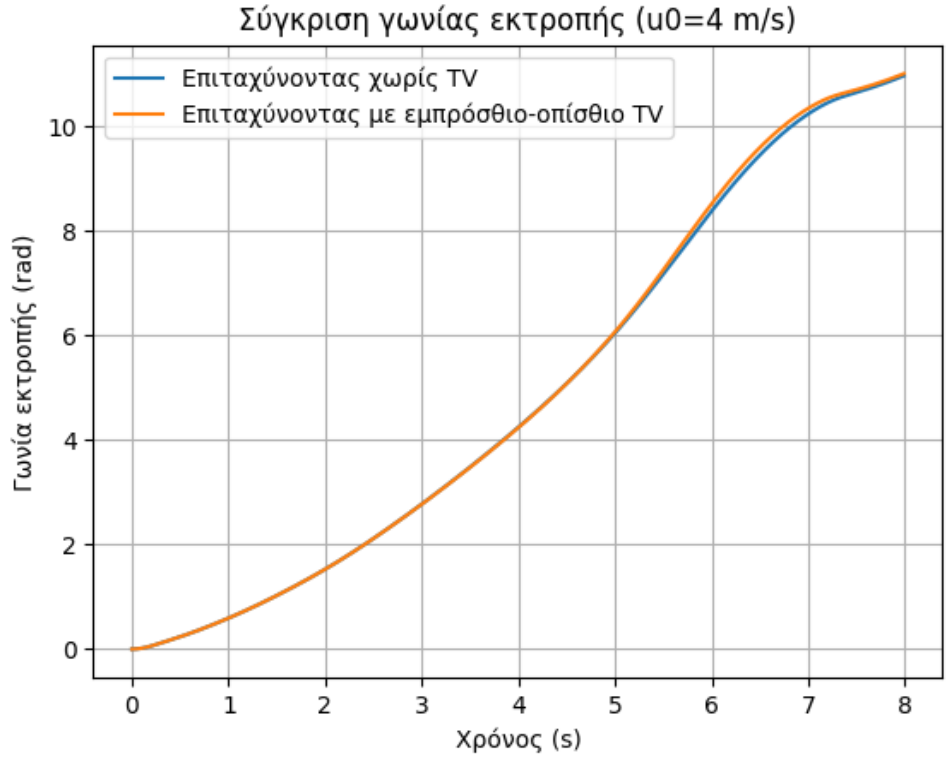
Σχήμα 5.50: Γωνία εκτροπής επιταχυνόμενης αριστερής στροφής με ρυθμό αύξησης γωνίας διεύθυνσης $v_\delta = 0.05\,rad/s$ και επιτάχυνση $a_{\text{long}} = 0.2g$



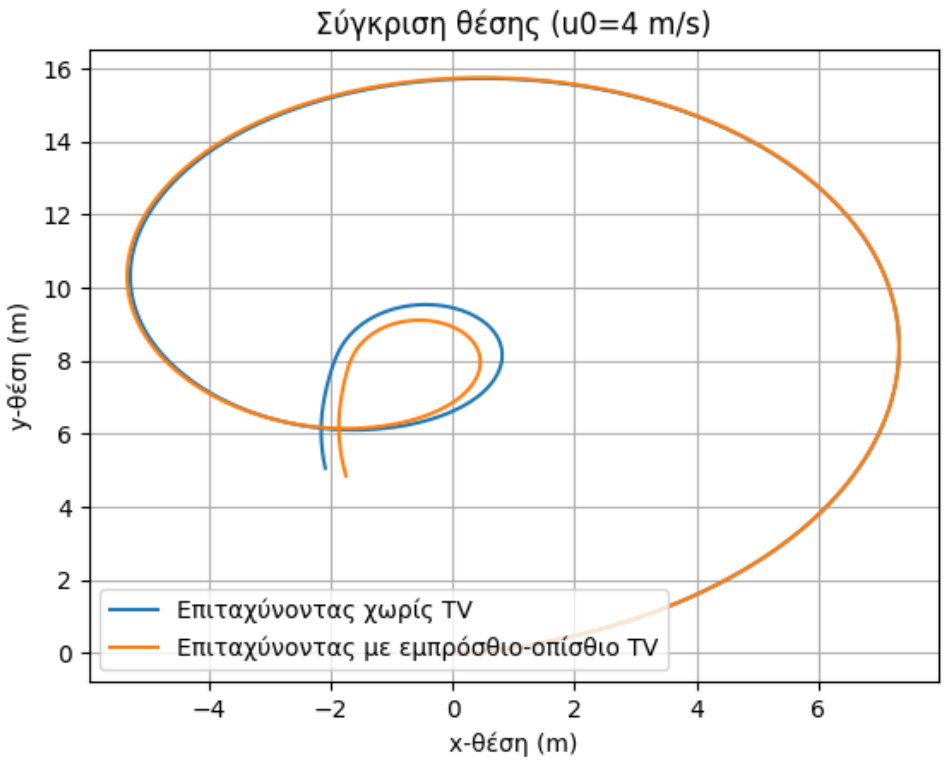
Σχήμα 5.51: Τροχιά επιταχυνόμενης αριστερής στροφής με ρυθμό αύξησης γωνίας διεύθυνσης $v_\delta = 0.05\,rad/s$ και επιτάχυνση $a_{\text{long}} = 0.2g$



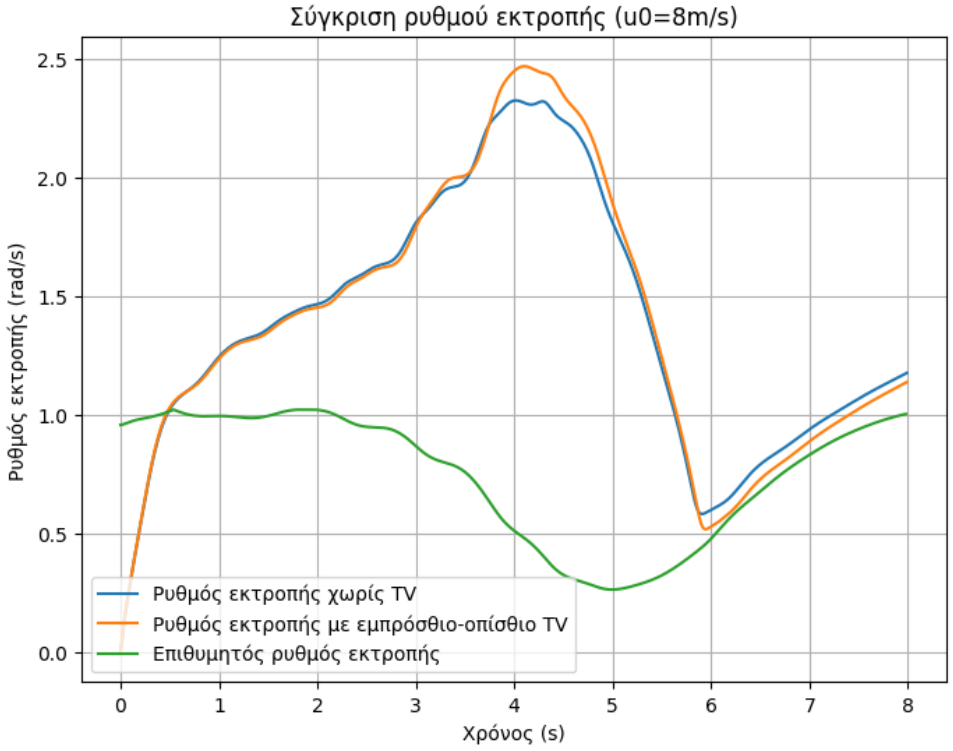
Σχήμα 5.52: Ρυθμός εκτροπής ελιγμού step-steer με επιβολή ακαριαίας γωνίας διεύθυνσης $\delta = 0.314\,rad$ και επιτάχυνση $a_{\text{long}} = 0.3g$



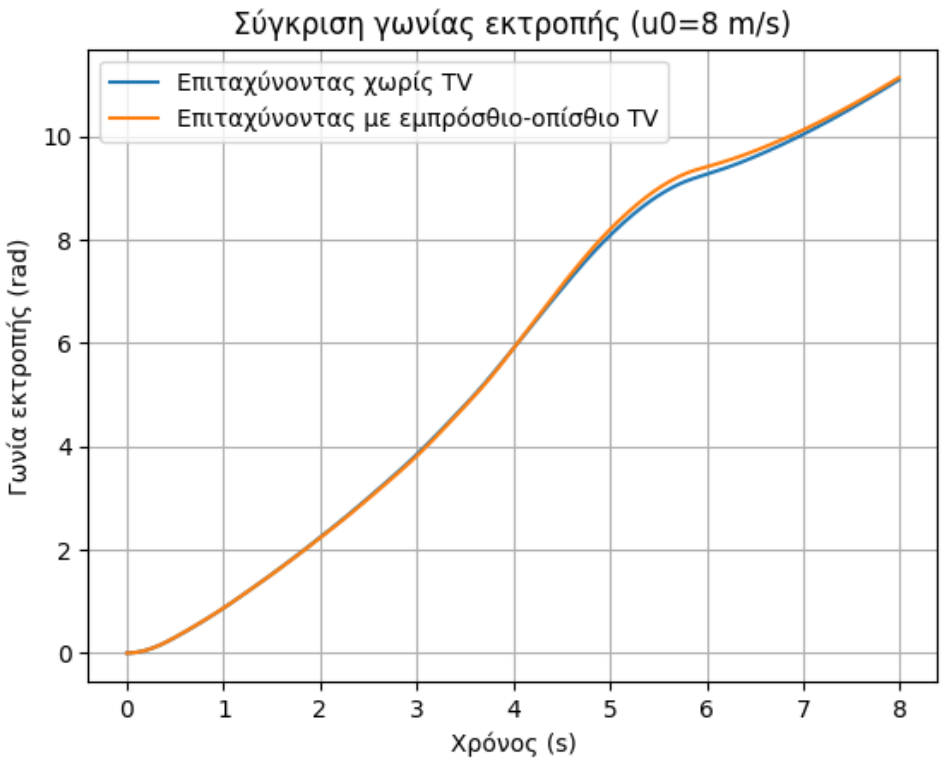
Σχήμα 5.53: Γωνία εκτροπής ελιγμού step-steer με επιβολή ακαριαίας γωνίας διεύθυνσης $\delta = 0.314\,rad$ και επιτάχυνση $a_{\text{long}} = 0.3g$



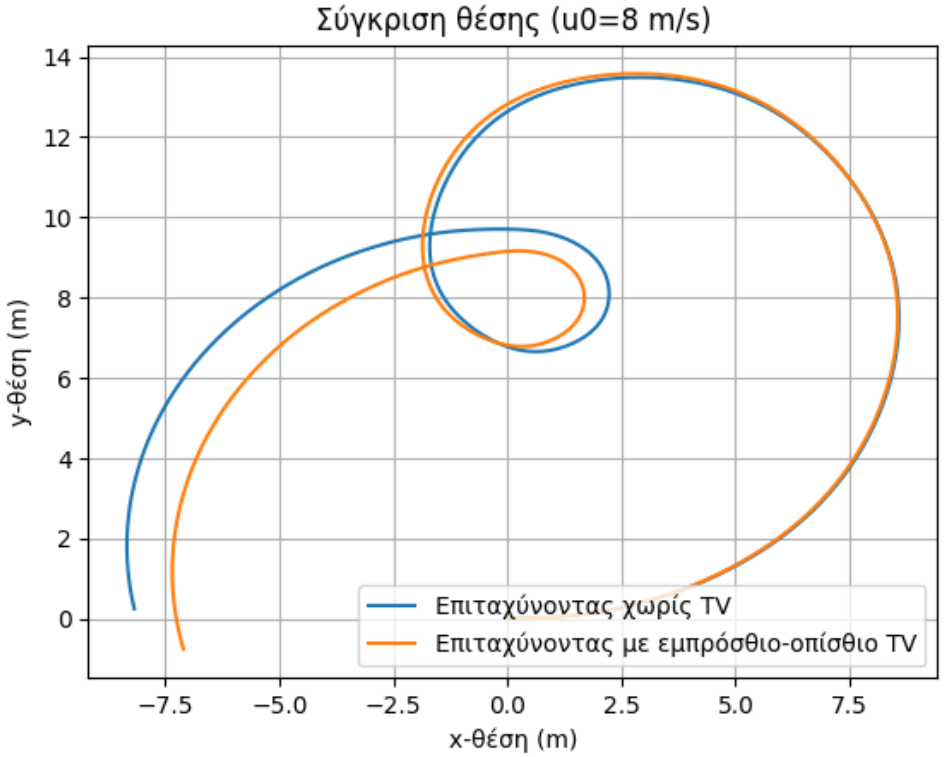
Σχήμα 5.54: Τροχιά ελιγμού step-steer με επιβολή ακαριαίας γωνίας διεύθυνσης $\delta = 0.314\,rad$ και επιτάχυνση $a_{\text{long}} = 0.3g$



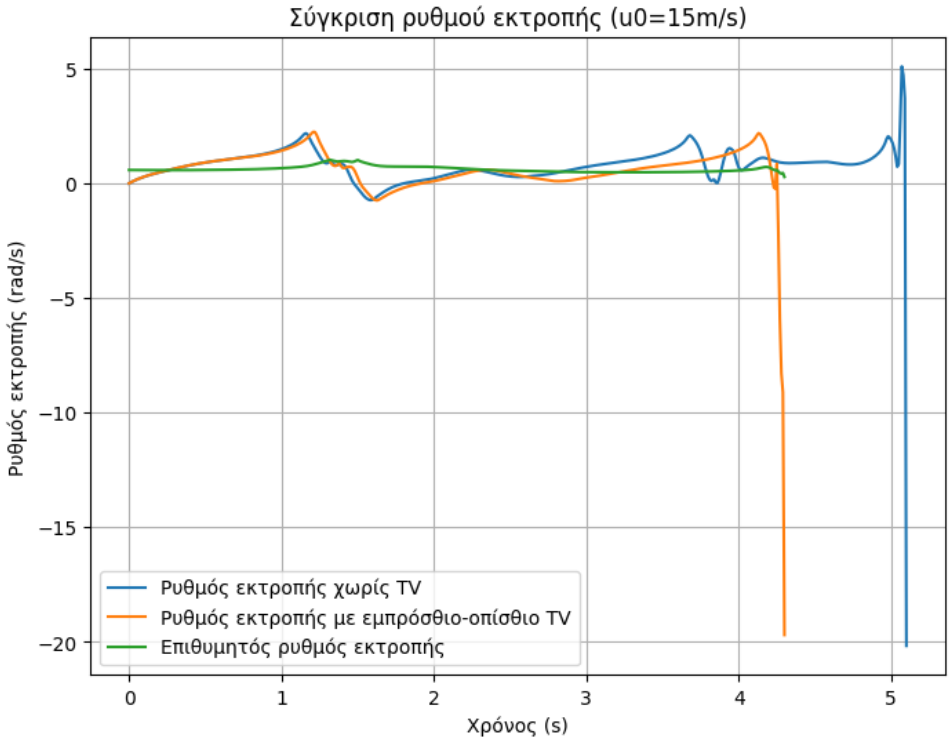
Σχήμα 5.55: Ρυθμός εκτροπής ελιγμού step-steer με επιβολή ακαριαίας γωνίας διεύθυνσης $\delta = 0.314\,rad$ και επιτάχυνση $a_{\text{long}} = 0.3g$



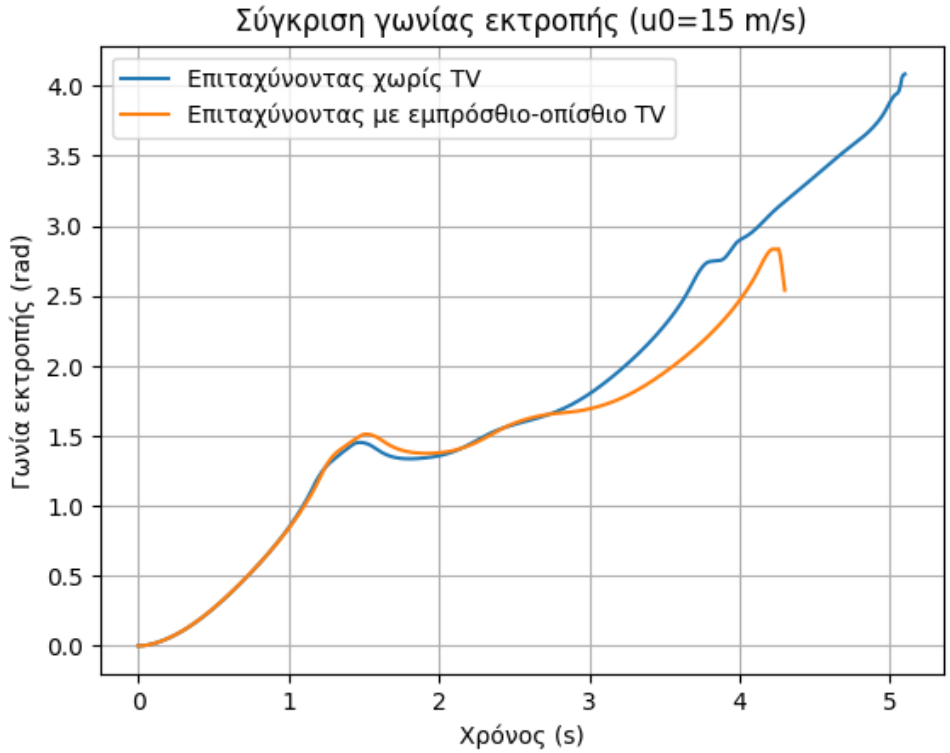
Σχήμα 5.56: Γωνία εκτροπής ελιγμού step-steer με επιβολή ακαριαίας γωνίας διεύθυνσης $\delta = 0.314\,rad$ και επιτάχυνση $a_{\text{long}} = 0.3g$



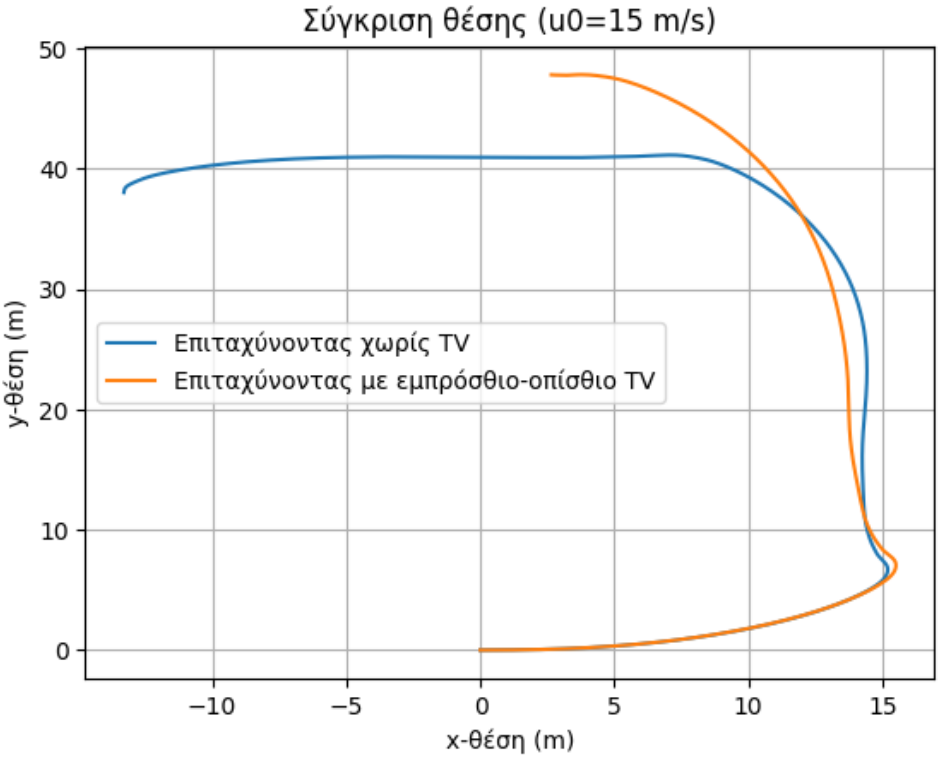
Σχήμα 5.57: Τροχιά ελιγμού step-steer με επιβολή ακαριαίας γωνίας διεύθυνσης $\delta = 0.314\,rad$ και επιτάχυνση $a_{\text{long}} = 0.3g$



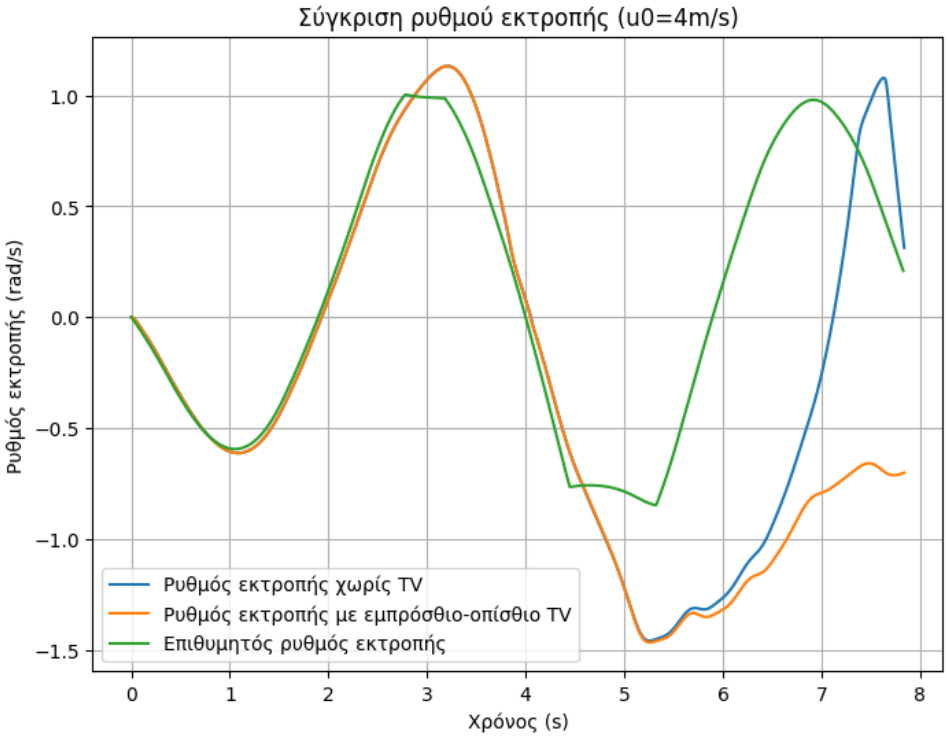
Σχήμα 5.58: Ρυθμός εκτροπής ελιγμού step-steer με επιβολή ακαριαίας γωνίας διεύθυνσης $\delta = 0.314\,rad$ και επιτάχυνση $a_{\text{long}} = 0.3g$



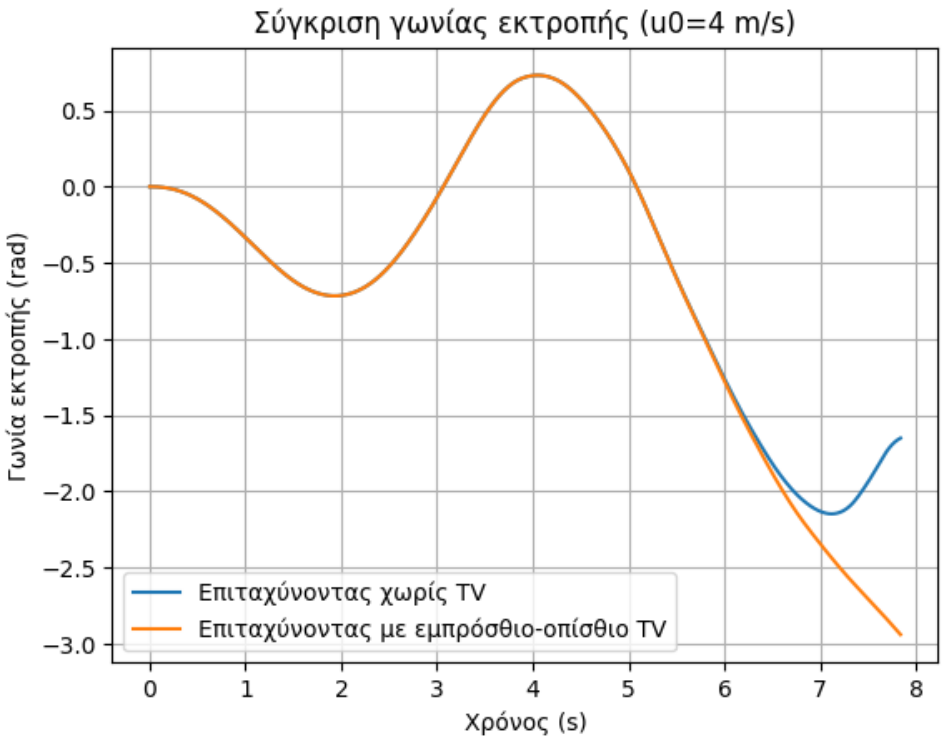
Σχήμα 5.59: Γωνία εκτροπής ελιγμού step-steer με επιβολή ακαριαίας γωνίας διεύθυνσης $\delta = 0.314\,rad$ και επιτάχυνση $a_{\text{long}} = 0.3g$



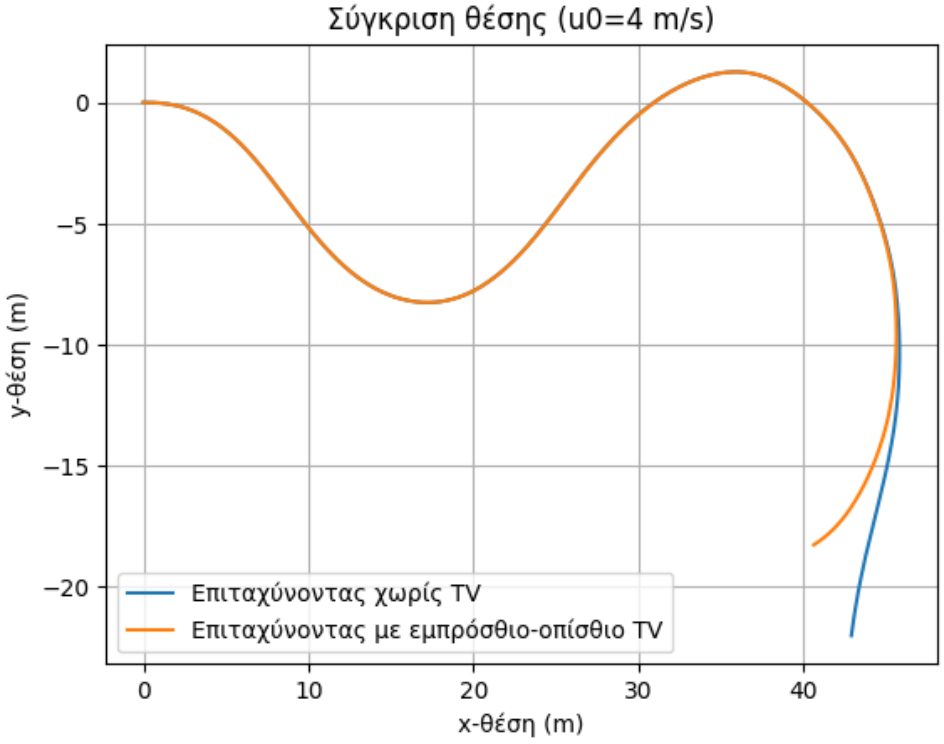
Σχήμα 5.60: Τροχιά ελιγμού step-steer με επιβολή ακαριαίας γωνίας διεύθυνσης $\delta = 0.314\,rad$ και επιτάχυνση $a_{\text{long}} = 0.3g$



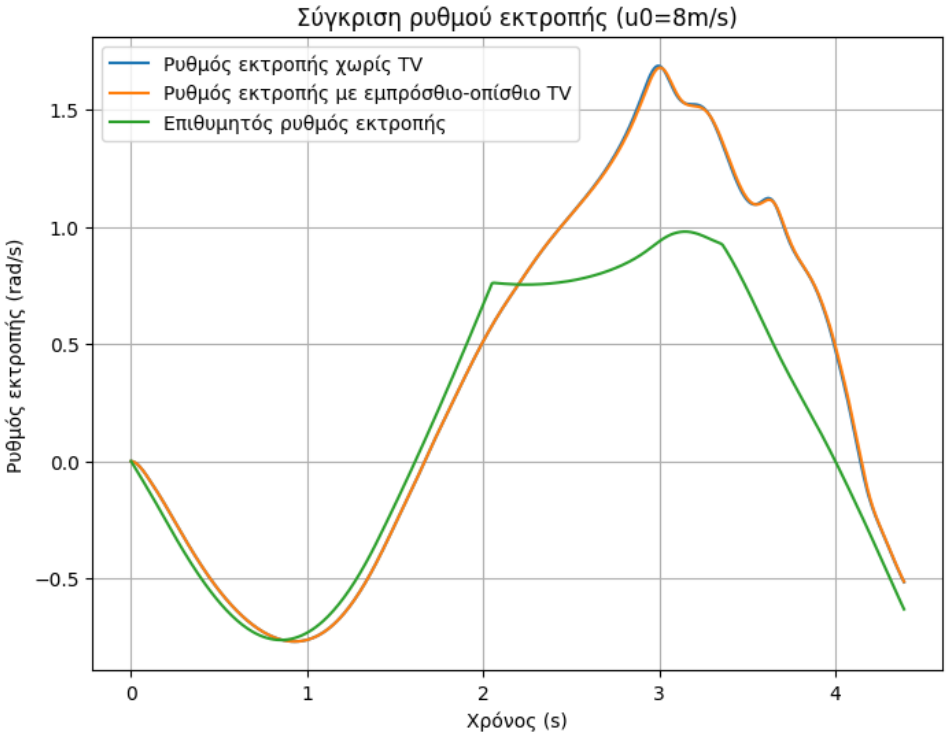
Σχήμα 5.61: Ρυθμός εκτροπής ελιγμού αποφυγής εμποδίου με επιτάχυνση $a_{\text{long}} = 0.2g$



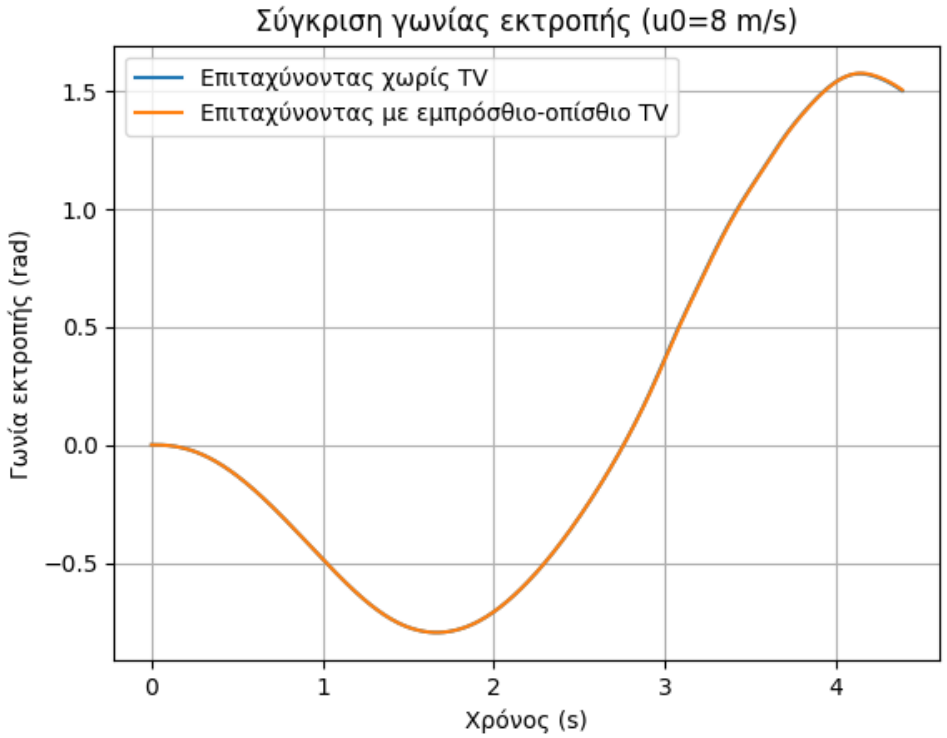
Σχήμα 5.62: Γωνία εκτροπής ελιγμού αποφυγής εμποδίου με επιτάχυνση $a_{\text{long}} = 0.2g$



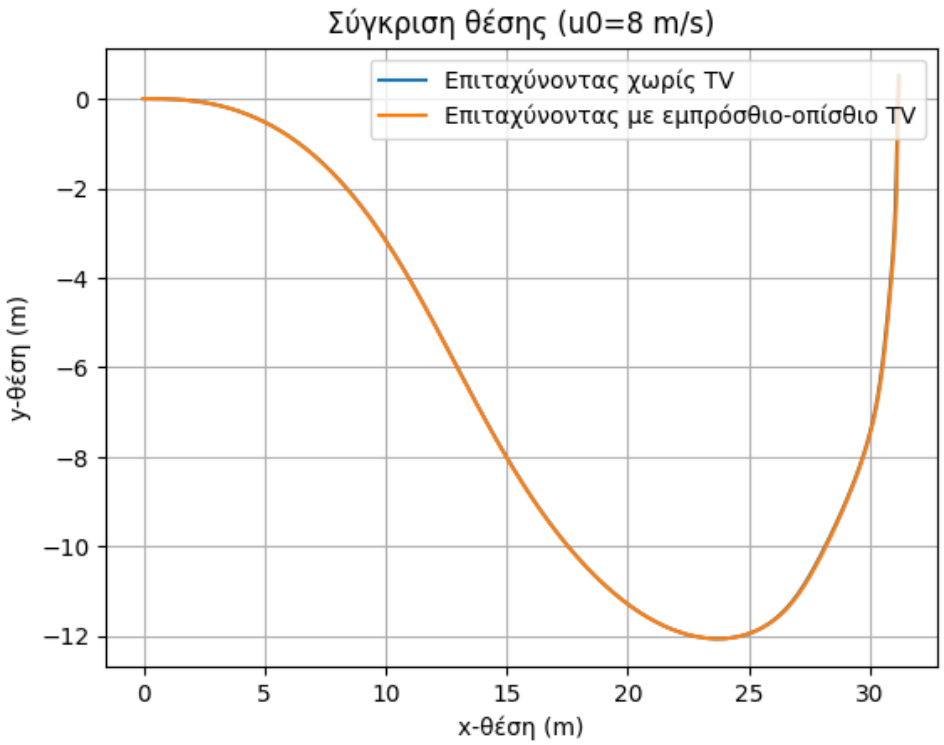
Σχήμα 5.63: Τροχιά ελιγμού αποφυγής εμποδίου με επιτάχυνση $a_{\text{long}} = 0.2g$



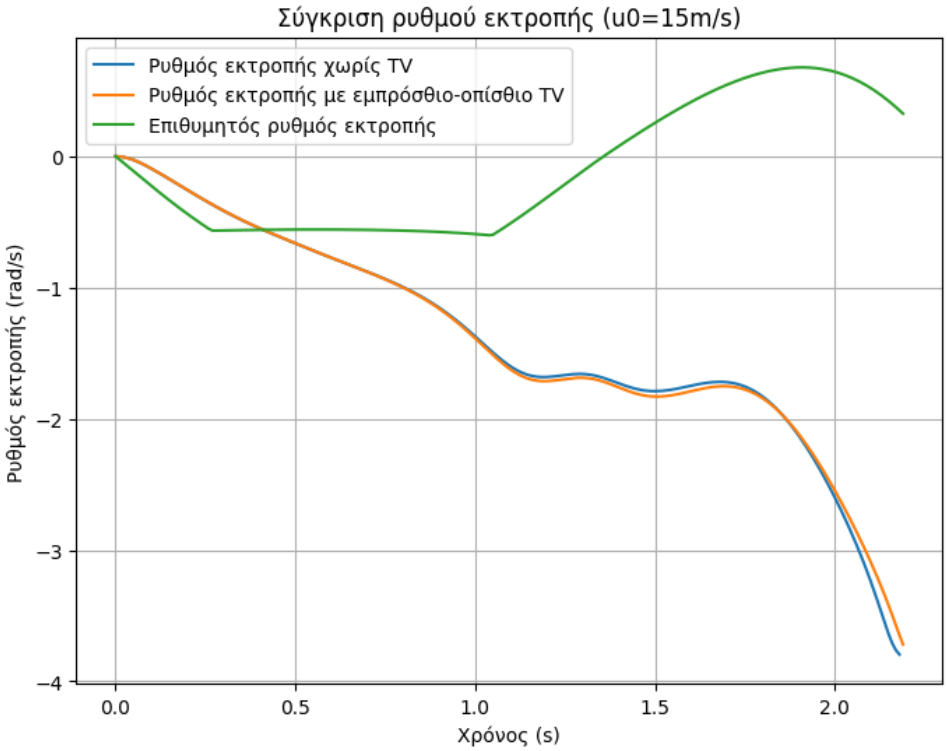
Σχήμα 5.64: Ρυθμός εκτροπής ελιγμού αποφυγής εμποδίου με επιτάχυνση $a_{\text{long}} = 0.2g$



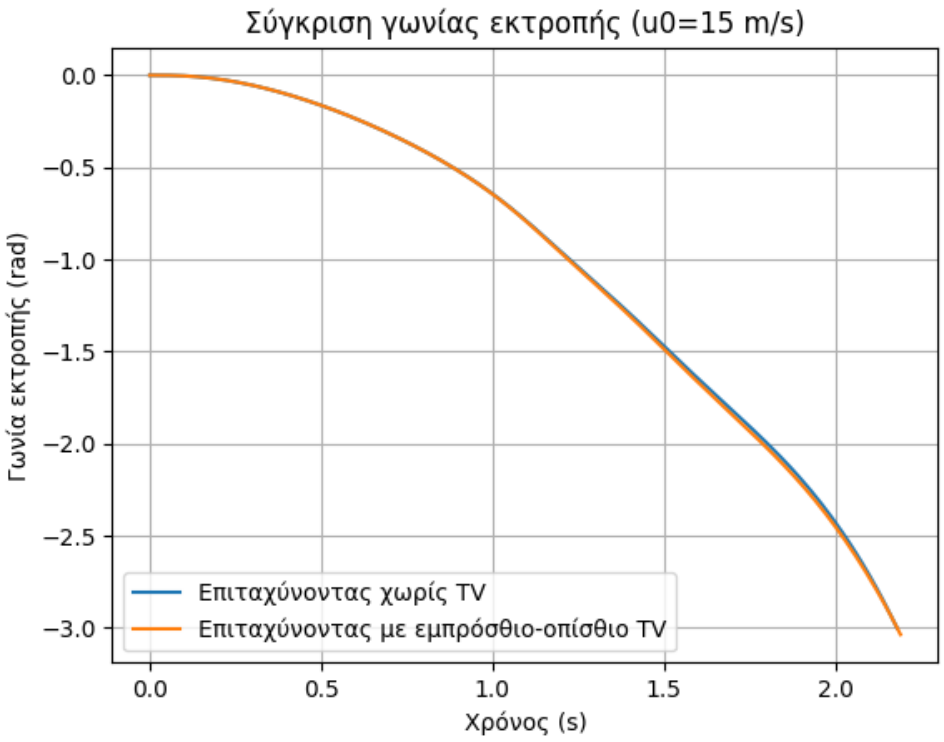
Σχήμα 5.65: Γωνία εκτροπής ελιγμού αποφυγής εμποδίου με επιτάχυνση $a_{\text{long}} = 0.2g$



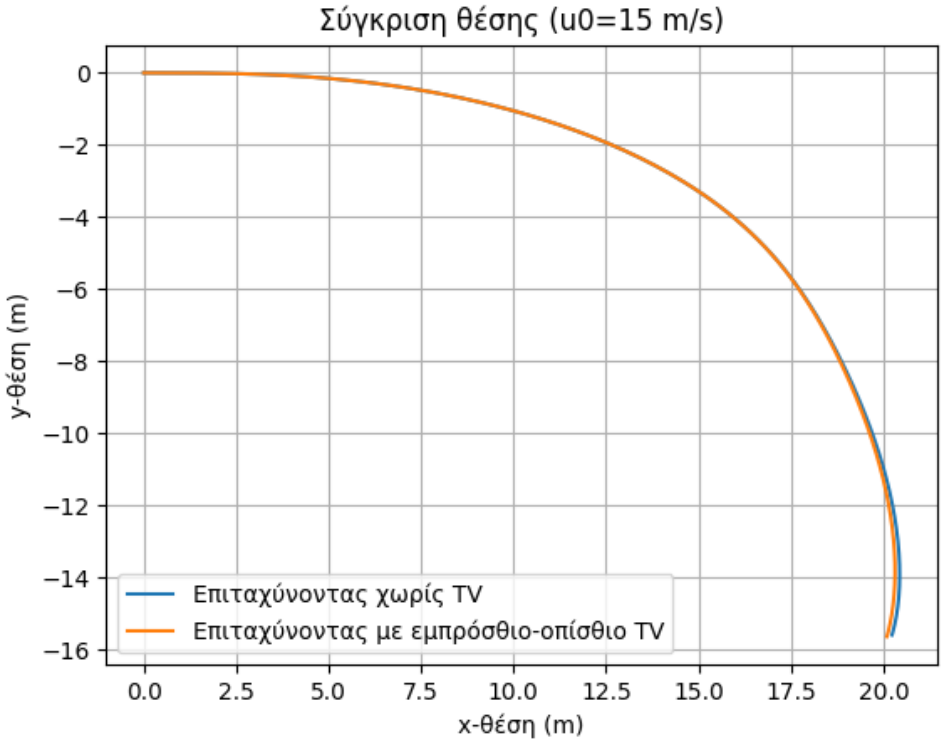
Σχήμα 5.66: Τροχιά ελιγμού αποφυγής εμποδίου με επιτάχυνση $a_{\text{long}} = 0.2g$



Σχήμα 5.67: Ρυθμός εκτροπής ελιγμού αποφυγής εμποδίου με επιτάχυνση $a_{\text{long}} = 0.2g$



Σχήμα 5.68: Γωνία εκτροπής ελιγμού αποφυγής εμποδίου με επιτάχυνση $a_{\text{long}} = 0.2g$



Σχήμα 5.69: Τροχιά ελιγμού αποφυγής εμποδίου με επιτάχυνση $a_{\text{long}} = 0.2g$

Ύστερα από προσεκτική μελέτη των παραπάνω διαγραμμάτων, είναι εμφανής η επιδείνωση της συμπεριφοράς του οχήματος κατά τη διάρκεια των περισσότερων ελιγμών με τη δράση του εμπρόσθιου-οπίσθιου TV μεμονωμένα. Μάλιστα, όσο η αρχική ταχύτητα αυξάνεται, τόσο πιο δραματική είναι η επιδείνωση.

Πιο συγκεκριμένα, για την περίπτωση της χαμηλότερης αρχικής ταχύτητας των $v_0 = 4\,m/s$, δεν παρατηρήθηκε καμία αλλαγή στα αποτελέσματα των προσομοιώσεων της επιταχυνόμενης αριστερής στροφής, όπως φαίνεται στα σχήματα 5.43 - 5.45. Στον ελιγμό step-steer υπήρξε μικρή επιδείνωση, αφού παρατηρήθηκε απόκλιση μεγαλύτερη κατά 0.2 rad/s από τον επιθυμητό ρυθμό εκτροπής λίγο πριν τη χρονική στιγμή $t = 6\,s$, όπως φαίνεται στο Σχ. 5.52. Κατά τη δοκιμή αποφυγής εμποδίου, τα αποτελέσματα ήταν εμφανώς χειρότερα. Από τη χρονική στιγμή $t = 6\,s$ κι έπειτα εμφανίζεται μεγάλη διακύμανση στο διάγραμμα του ρυθμού εκτροπής, η οποία σύντομα θα οδηγούσε σε απώλεια ελέγχου (βλ. σχήμα 5.61). Το συμπέρασμα αυτό ενισχύεται από τα Σχ. 5.62 και 5.63 της γωνίας εκτροπής και της τροχιάς.

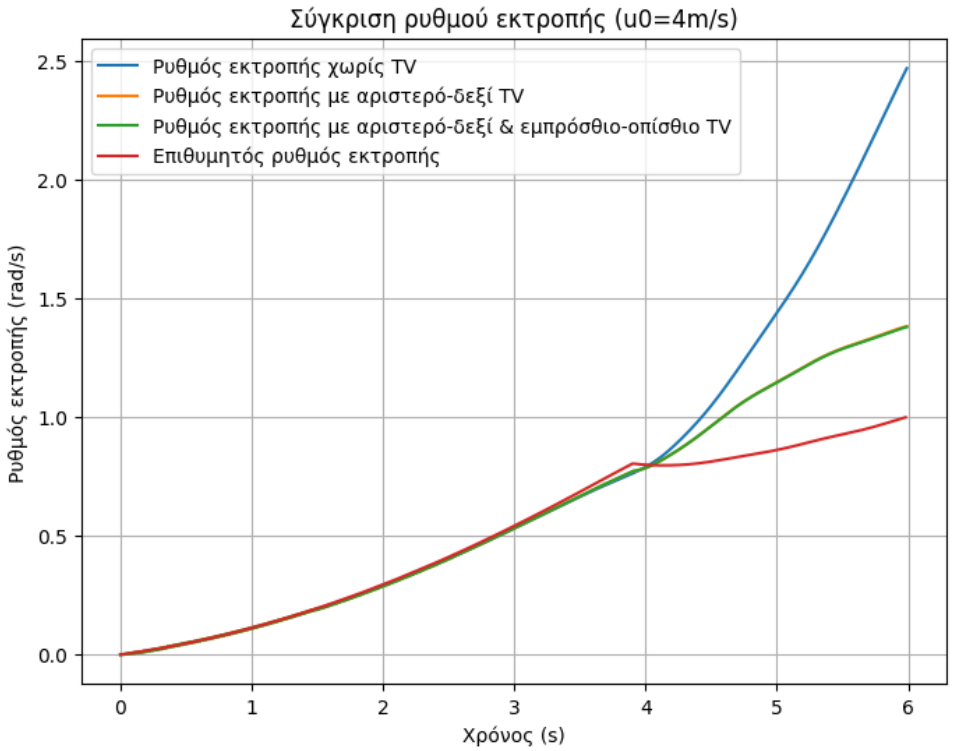
Για την περίπτωση των $v_0 = 8\,m/s$, παρατηρήθηκε ελάχιστη επιδείνωση στον ελιγμό της επιταχυνόμενης στροφής, όπως φαίνεται στο Σχ. 5.46, όπου παρουσιάζεται μικρή αύξηση του σφάλματος. Η αυξηση αυτή ωστόσο δε μεταφράζεται σε αλλαγή στην τροχιά του οχήματος (Σχ. 5.48). Στον ελιγμό step-steer υπήρξε μικρή επιδείνωση, αφού παρατηρήθηκε μεγαλύτερη απόκλιση από τον επιθυμητό ρυθμό εκτροπής κατά 0.3 rad/s τη χρονική στιγμή $t = 4\,s$, όπως φαίνεται στο Σχ. 5.55. Κατά τη δοκιμή αποφυγής εμποδίου, δεν παρατηρήθηκε καμία αλλαγή στα αποτελέσματα των προσομοιώσεων (βλ. Σχ. 5.64 - 5.66).

Τέλος, κατά τις δοκιμές με την υψηλότερη αρχική ταχύτητα των $v_0 = 15\,m/s$, παρατηρείται επίσης μικρή επιδείνωση στον ελιγμό της επιταχυνόμενης στροφής, όπως προδίδει το Σχ. 5.49, όπου το όχημα φαίνεται να χάνει τον έλεγχο οριακά νωρίτερα. Όσον αφορά τον ελιγμό step-steer, τα αποτελέσματα της προσομοίωσης είναι εμφανώς χειρότερα, δεδομένου ότι το όχημα εκτρέπεται της πορείας του σχεδόν 1 δευτερόλεπτο νωρίτερα σε σχέση με τον ελιγμό με το σύστημα ανενεργό. Τα παραπάνω αποδεικνύονται από τα σχήματα 5.58 και 5.59 του ρυθμού και της γωνίας εκτροπής, αφού λίγο μετά τη χρονική στιγμή $t = 4\,s$ παρατηρούνται τεράστιες αλλαγές και έπειτα σταματούν να παράγονται τιμές. Το όχημα χωρίς TV είχε καταφέρει να εκτελέσει $\Delta t = 5\,s$ του ελιγμού προτού συναντήσει την ίδια μοίρα. Κατά τη δοκιμή αποφυγής εμποδίου, οι αλλαγές που παρατηρήθηκαν στα αποτελέσματα των προσομοιώσεων ήταν αμελητέες (βλ. Σχ. 5.67 - 5.69).

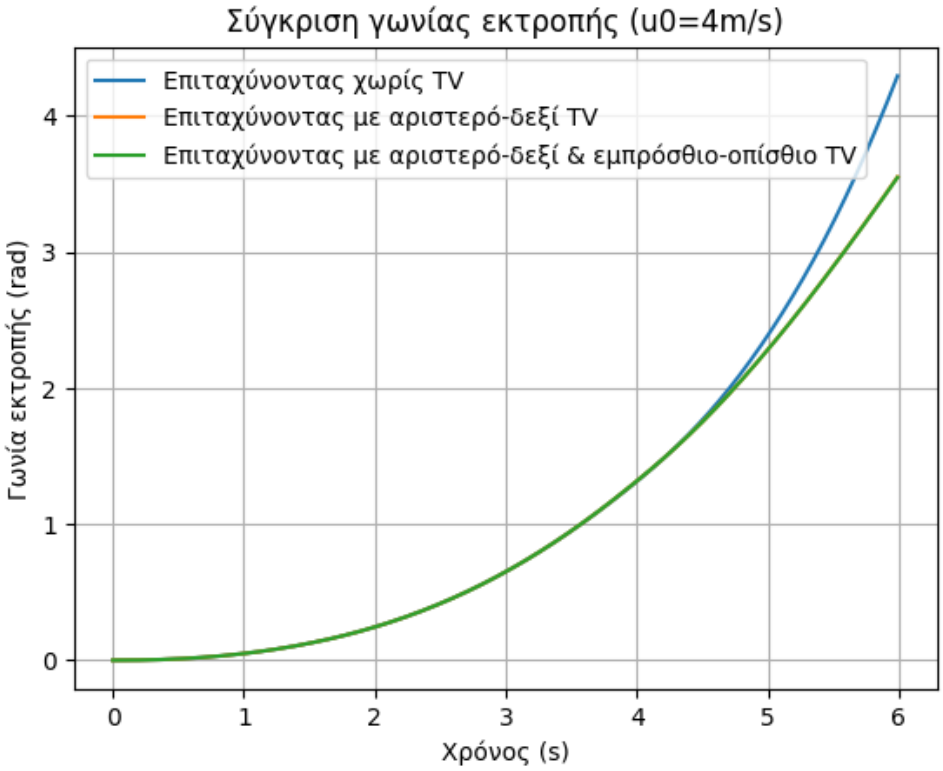
Τα παραπάνω αποτελέσματα επαληθεύουν την ανωτερότητα των συστημάτων τετρακίνησης, σε σχέση με τα συστήματα μετάδοσης κίνησης σε έναν μόνο άξονα (FWD & RWD). Αναδεικνύεται επίσης η αποδοτικότητα του left-right TV, αφού το βασικό μοντέλο στο οποίο εφαρμόστηκε είναι τετρακίνητο, γεγονός που καθιστά τη συμπεριφορά του σε ελιγμούς εξ' ορισμού αρκετά προβλέψιμη και ασφαλή. Όσον αφορά το εμπρόσθιο-οπίσθιο TV, η ρύθμιση ενός τέτοιου συστήματος φαίνεται να είναι πολύ απαιτητική. Σκέψεις προς βελτίωση του υφιστάμενου αλγορίθμου θα παρουσιαστούν παρακάτω.

5.3 Αριστερό - δεξί & εμπρόσθιο-οπίσθιο TV

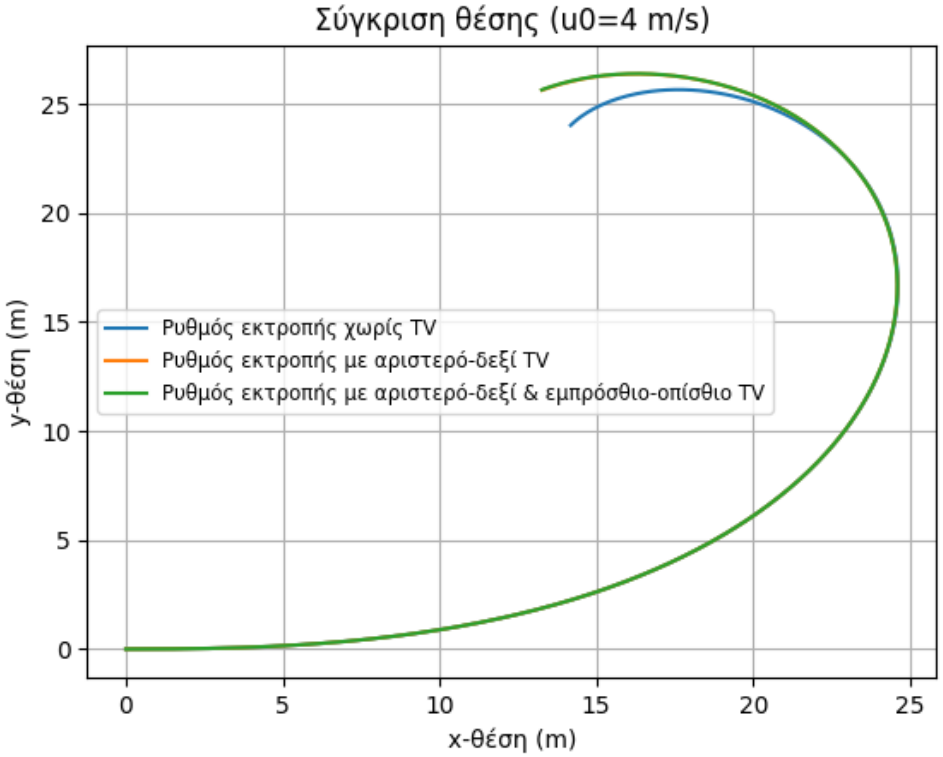
Τα Σχ.5.70 - 5.78, 5.79 - 5.87 και 5.88 - 5.96 που φαίνονται αμέσως μετά παραθέτουν τα αποτελέσματα ταυτόχρονης επιβολής εμπρόσθιου-οπίσθιου και αριστερού-δεξιού TV στους ελιγμούς επιταχυνόμενης αριστερής στροφής, step-steer και αποφυγής εμποδίου αντιστοίχως.



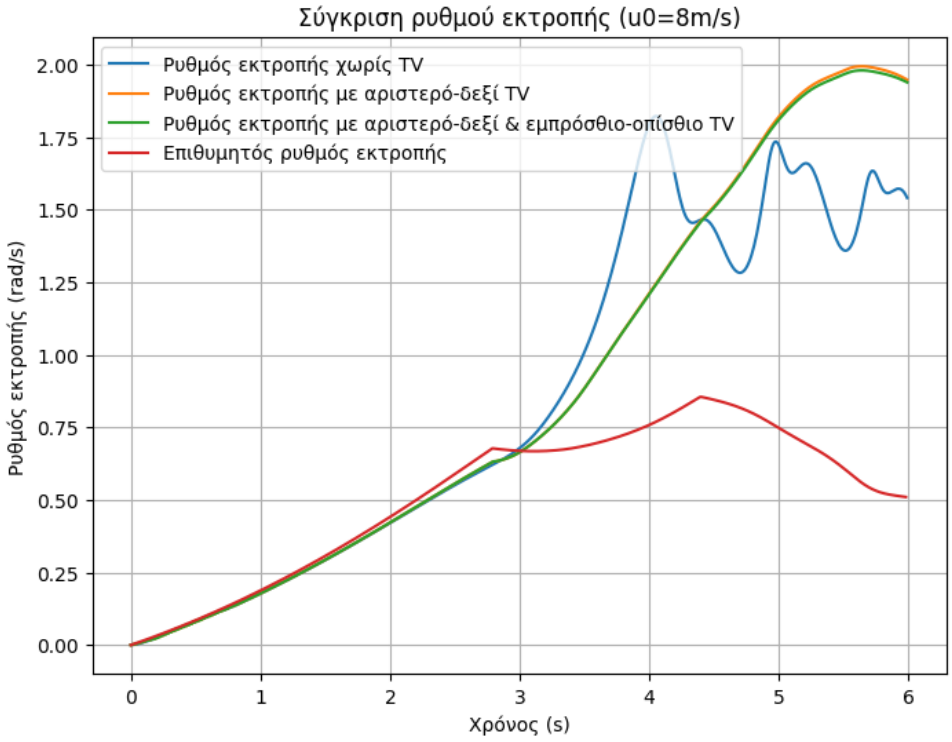
Σχήμα 5.70: Ρυθμός εκτροπής επιταχυνόμενης αριστερής στροφής με ρυθμό αύξησης γωνίας διεύθυνσης $v_\delta = 0.05\,rad/s$ και επιτάχυνση $a_{\text{long}} = 0.2g$



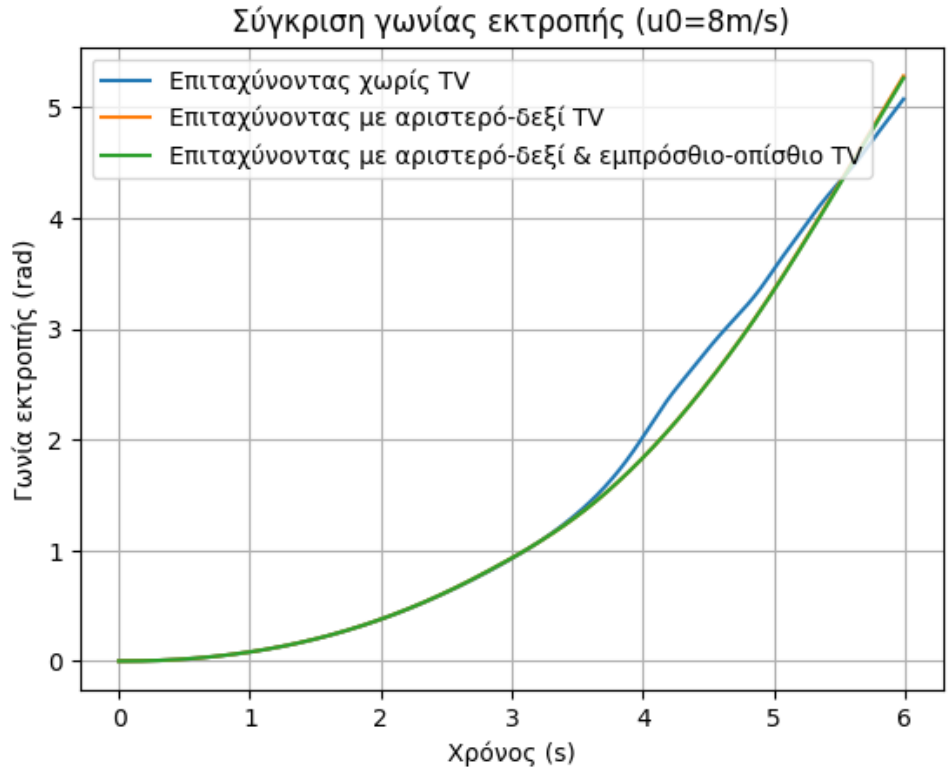
Σχήμα 5.71: Γωνία εκτροπής επιταχυνόμενης αριστερής στροφής με ρυθμό αύξησης γωνίας διεύθυνσης $v_\delta = 0.05\,rad/s$ και επιτάχυνση $a_{\text{long}} = 0.2g$



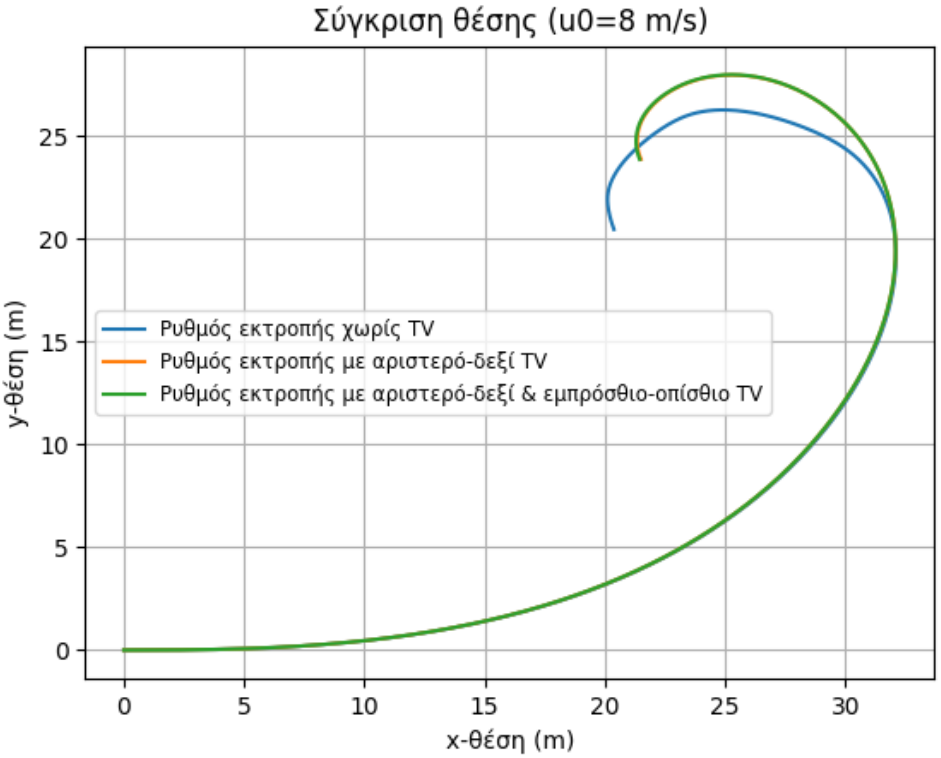
Σχήμα 5.72: Τροχιά επιταχυνόμενης αριστερής στροφής με ρυθμό αύξησης γωνίας διεύθυνσης $v_\delta = 0.05\,rad/s$ και επιτάχυνση $a_{\text{long}} = 0.2g$



Σχήμα 5.73: Ρυθμός εκτροπής επιταχυνόμενης αριστερής στροφής με ρυθμό αύξησης γωνίας διεύθυνσης $v_\delta = 0.05\,rad/s$ και επιτάχυνση $a_{\text{long}} = 0.2g$

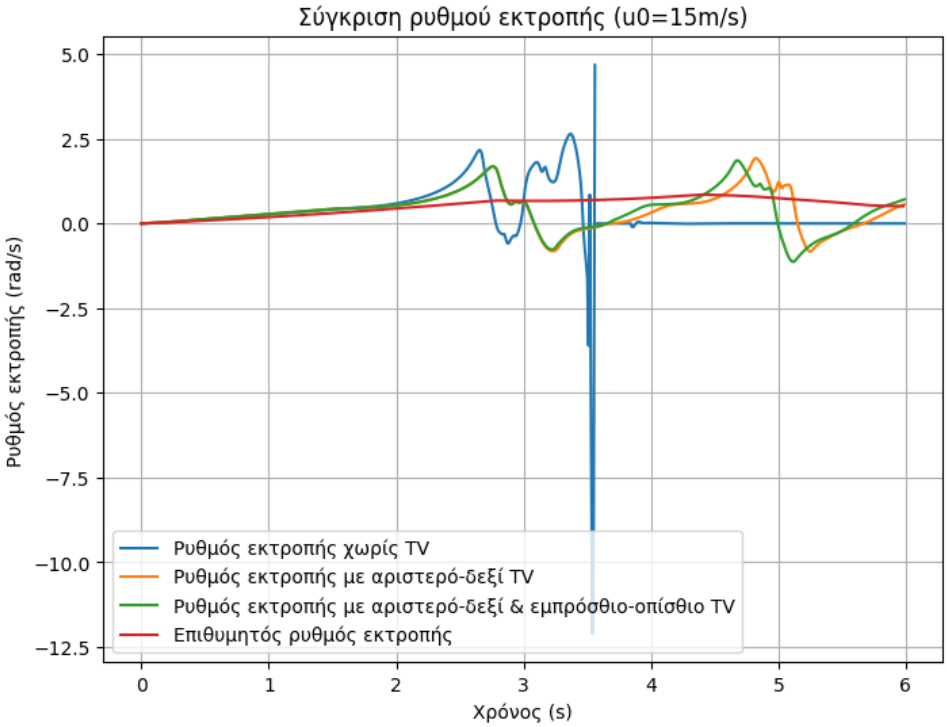


Σχήμα 5.74: Γωνία εκτροπής επιταχυνόμενης αριστερής στροφής με ρυθμό αύξησης γωνίας διεύθυνσης $v_\delta = 0.05\,rad/s$ και επιτάχυνση $a_{\text{long}} = 0.2g$

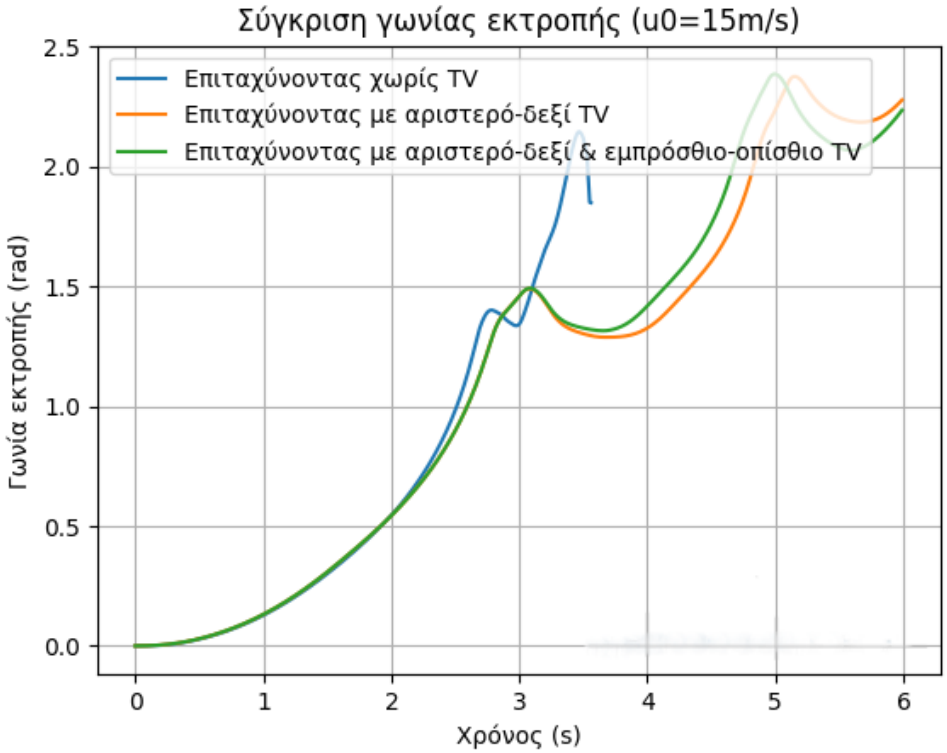


Σχήμα 5.75: Τροχιά επιταχυνόμενης αριστερής στροφής με ρυθμό αύξησης γωνίας διεύθυνσης $v_\delta = 0.05\,rad/s$ και επιτάχυνση $a_{\text{long}} = 0.2g$

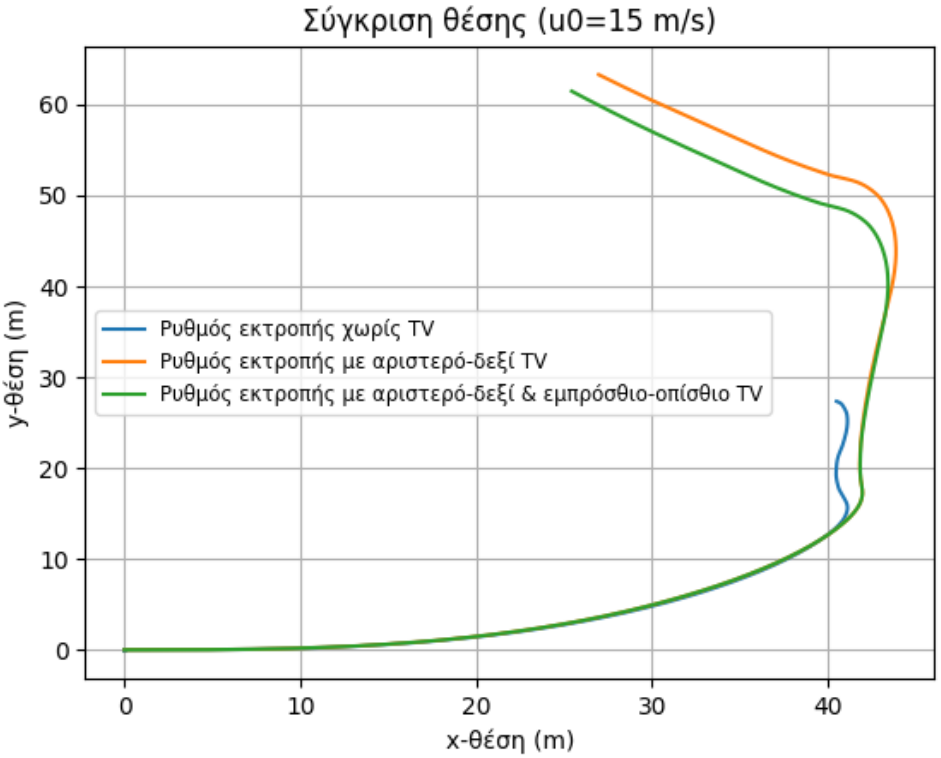
Όσον αφορά τον πρώτο ελιγμό, για $v_0 = 4\,m/s$ και $v_0 = 8\,m/s$ τα αποτελέσματα είναι ίδια σε σχέση με αυτά του αριστερού-δεξιά TV (βλ. σχήματα 5.70 - 5.75).



Σχήμα 5.76: Ρυθμός εκτροπής επιταχυνόμενης αριστερής στροφής με ρυθμό αύξησης γωνίας διεύθυνσης $v_\delta = 0.05\,rad/s$ και επιτάχυνση $a_{\text{long}} = 0.2g$

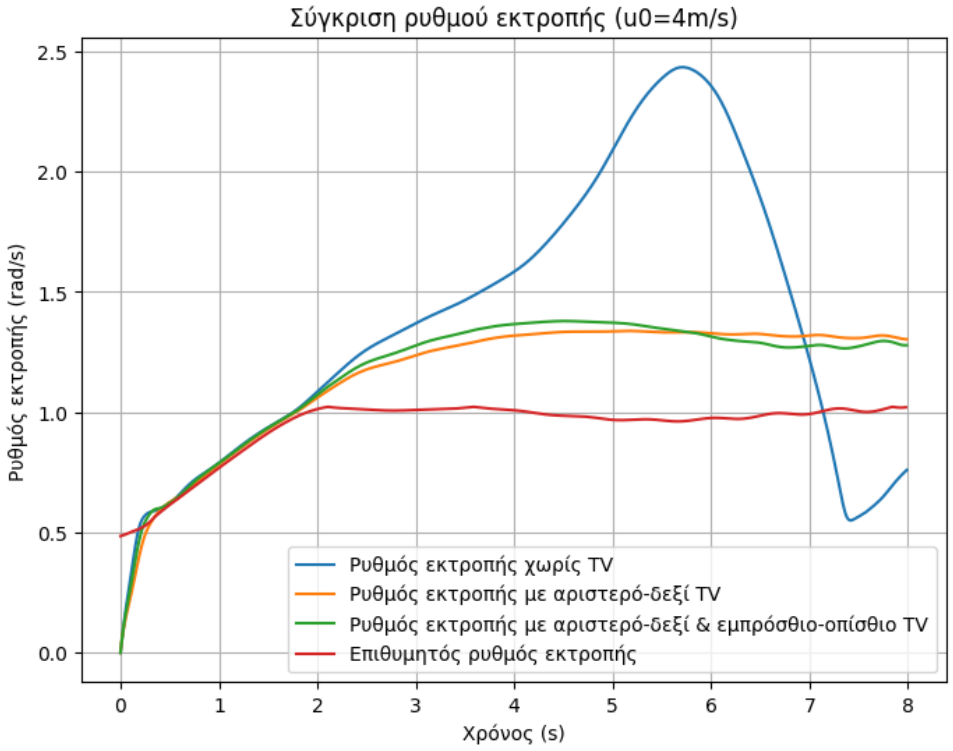


Σχήμα 5.77: Γωνία εκτροπής επιταχυνόμενης αριστερής στροφής με ρυθμό αύξησης γωνίας διεύθυνσης $v_\delta = 0.05\,rad/s$ και επιτάχυνση $a_{\text{long}} = 0.2g$

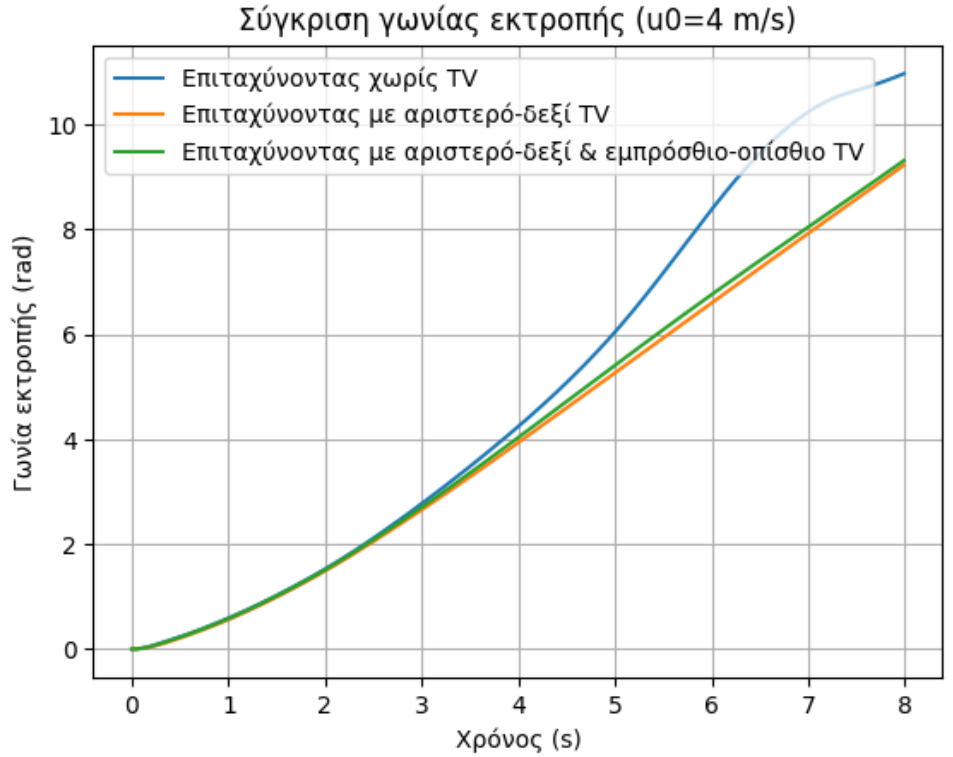


Σχήμα 5.78: Τροχιά επιταχυνόμενης αριστερής στροφής με ρυθμό αύξησης γωνίας διεύθυνσης $v_\delta = 0.05\,rad/s$ και επιτάχυνση $a_{\text{long}} = 0.2g$

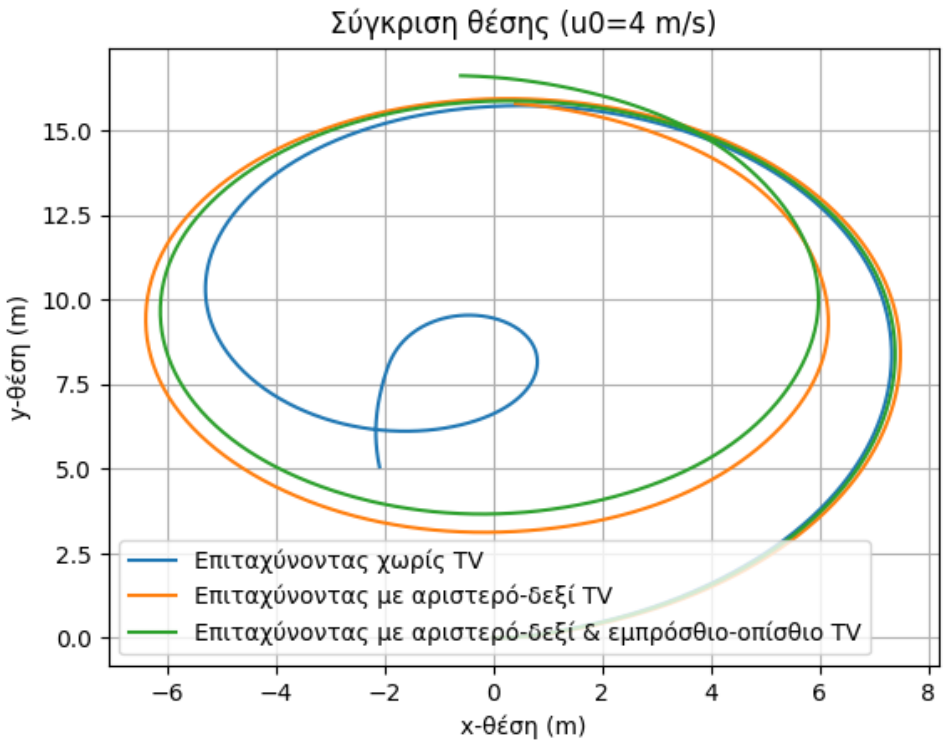
Για $v_0 = 15\,m/s$, παρατηρείται μία ετεροχρονισμένη διόρθωση στο σχήμα 5.76 του ρυθμού εκτροπής λίγο πριν τη στιγμή $t = 5\,s$, η οποία μεταφράζεται σε λίγο καθυστερημένη εκτέλεση της αριστερής στροφής μεταξύ του χρονικού διαστήματος $t = 4\,s$ - $t = 5\,s$, όπως φαίνεται στο σχήμα 5.78.



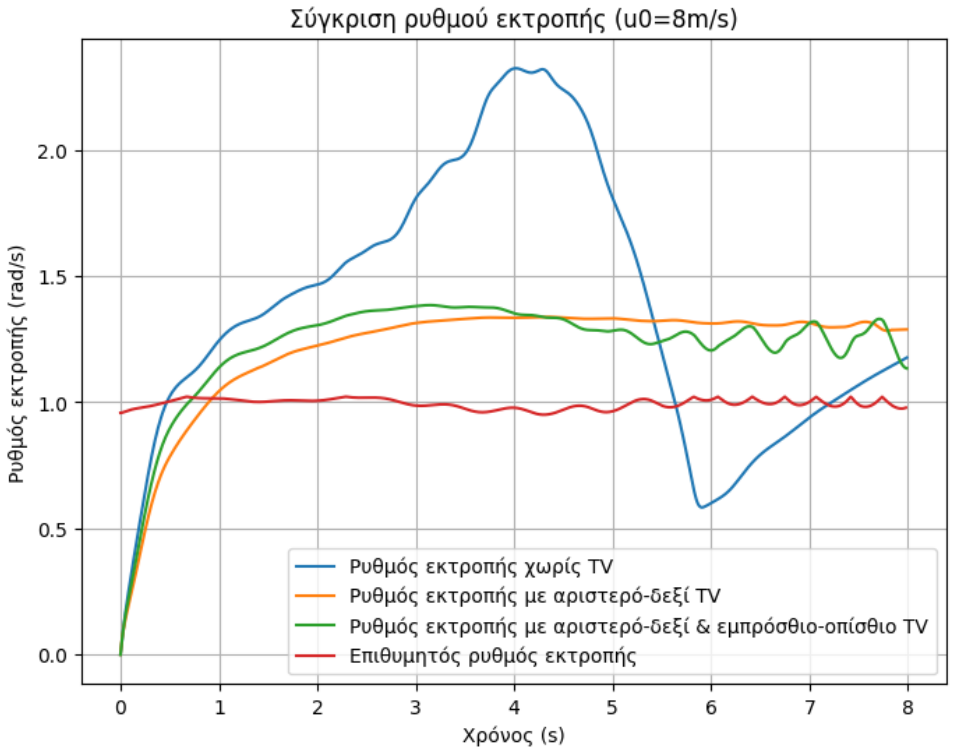
Σχήμα 5.79: Ρυθμός εκτροπής ελιγμού step-steer με επιβολή ακαριαίας γωνίας διεύθυνσης $\delta = 0.314\,rad$ και επιτάχυνση $a_{\text{long}} = 0.3g$



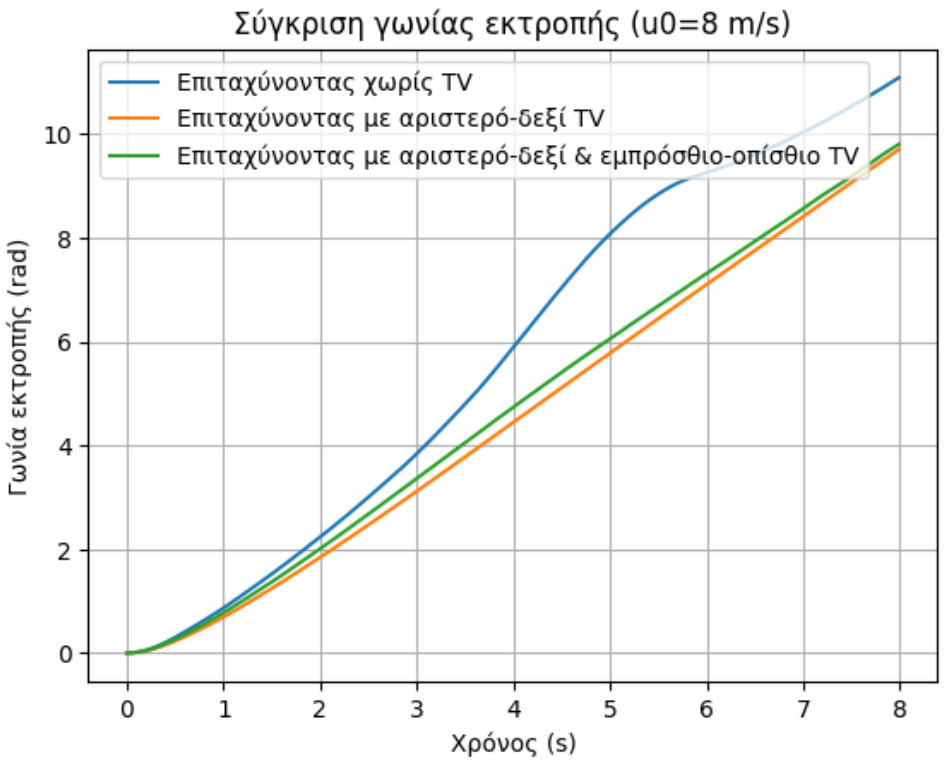
Σχήμα 5.80: Γωνία εκτροπής ελιγμού step-steer με επιβολή ακαριαίας γωνίας διεύθυνσης $\delta = 0.314\,rad$ και επιτάχυνση $a_{\text{long}} = 0.3g$



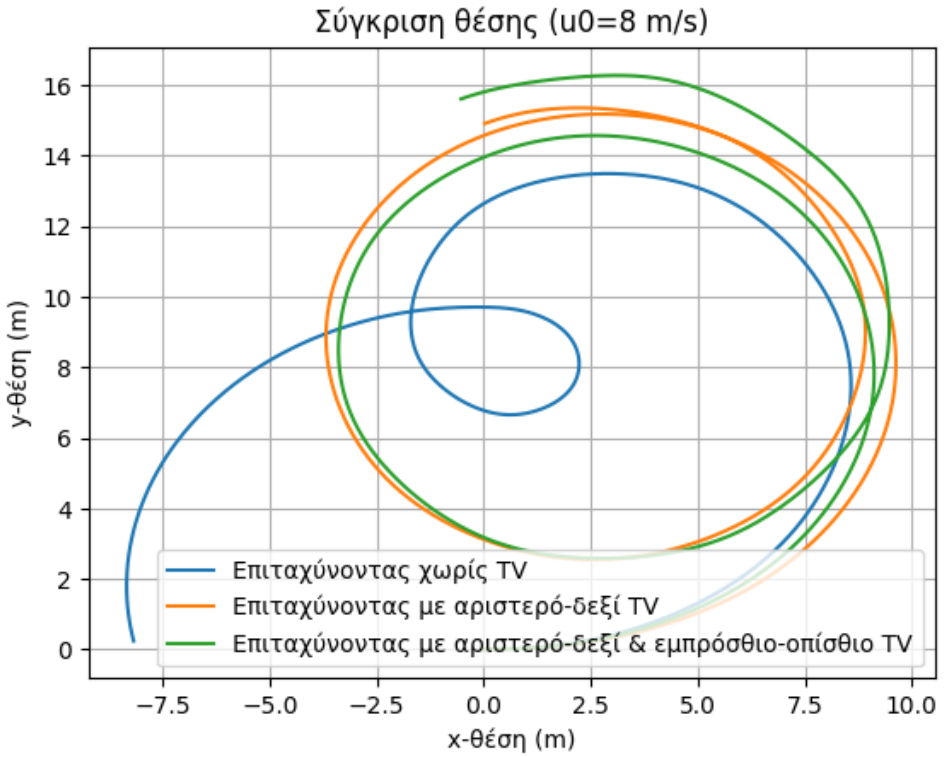
Σχήμα 5.81: Τροχιά ελιγμού step-steer με επιβολή ακαριαίας γωνίας διεύθυνσης $\delta = 0.314\,rad$ και επιτάχυνση $a_{\text{long}} = 0.3g$



Σχήμα 5.82: Ρυθμός εκτροπής ελιγμού step-steer με επιβολή ακαριαίας γωνίας διεύθυνσης $\delta = 0.314\,rad$ και επιτάχυνση $a_{\text{long}} = 0.3g$

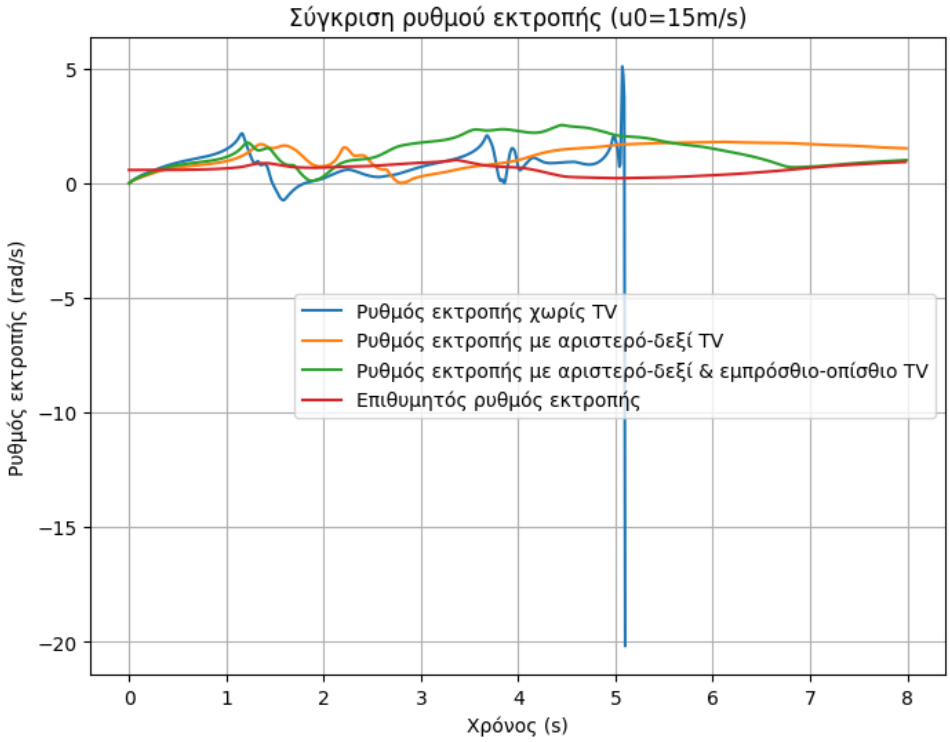


Σχήμα 5.83: Γωνία εκτροπής ελιγμού step-steer με επιβολή ακαριαίας γωνίας διεύθυνσης $\delta = 0.314\,rad$ και επιτάχυνση $a_{\text{long}} = 0.3g$

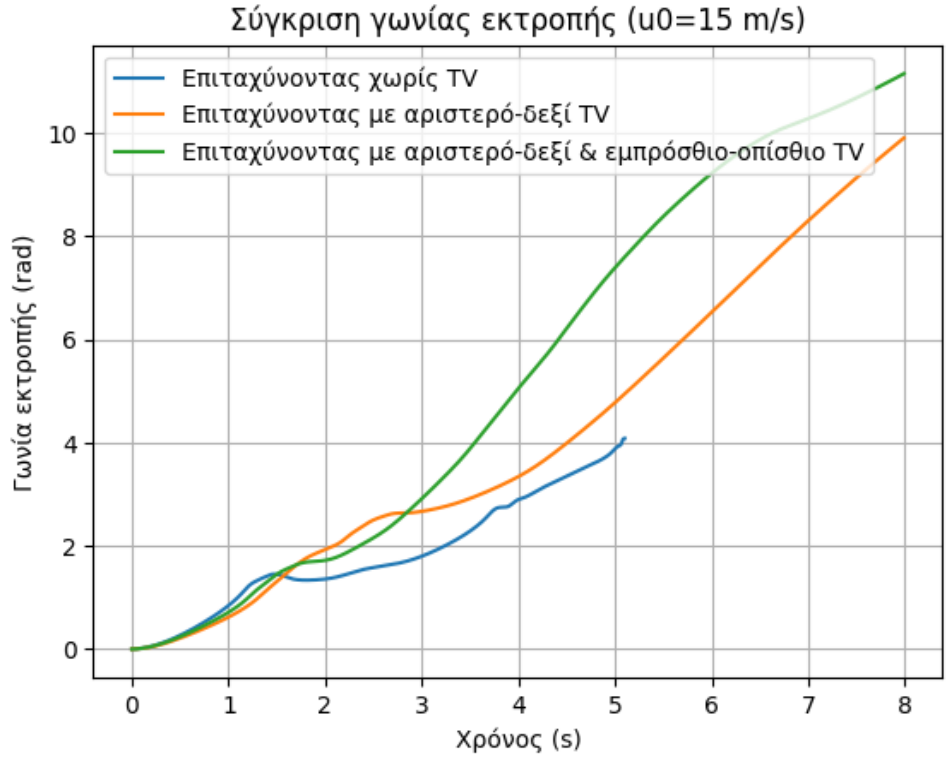


Σχήμα 5.84: Τροχιά ελιγμού step-steer με επιβολή ακαριαίας γωνίας διεύθυνσης $\delta = 0.314\,rad$ και επιτάχυνση $a_{\text{long}} = 0.3g$

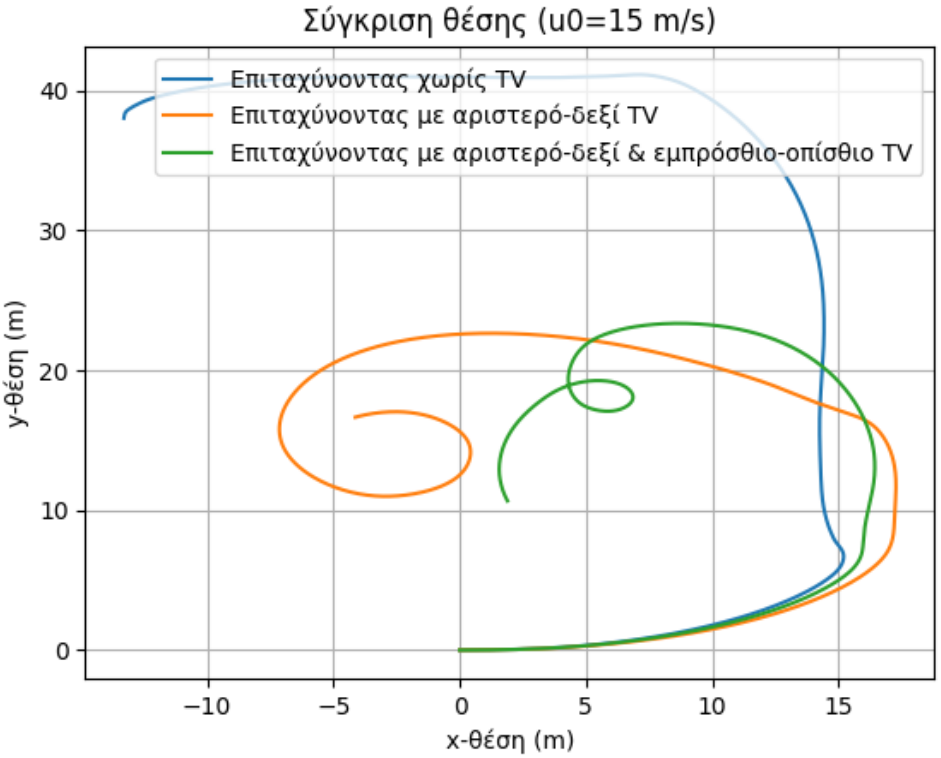
Στους ελιγμούς step-steer, τα αποτελέσματα του συνδυαστικού TV για $v_0 = 4\,m/s$ και $v_0 = 8\,m/s$ είναι παραπλήσια με αυτά του αριστερού-δεξιού TV. Στα σχήματα των ρυθμών εκτροπής 5.79 και 5.82 παρατηρείται αρχικά παραπάνω υπερστροφική συμπεριφορά, ακολουθούμενη από μικρές διακυμάνσεις. Αυτό έχει ως αποτέλεσμα τη δημιουργία ελάχιστα στενότερης τροχιάς, όπως προκύπτει από τα Σχ. 5.81 και 5.84. Η εκτέλεση στενότερης τροχιάς τεκμηριώνεται και από τα σχήματα των γωνιών εκτροπής, όπου το σύστημα με τις συνδυαστικές τακτικές TV παρουσιάζει μικρή αύξηση στις τιμές (βλ. σχήματα 5.80 και 5.83).



Σχήμα 5.85: Ρυθμός εκτροπής ελιγμού step-steer με επιβολή ακαριαίας γωνίας διεύθυνσης $\delta = 0.314\,rad$ και επιτάχυνση $a_{\text{long}} = 0.3g$

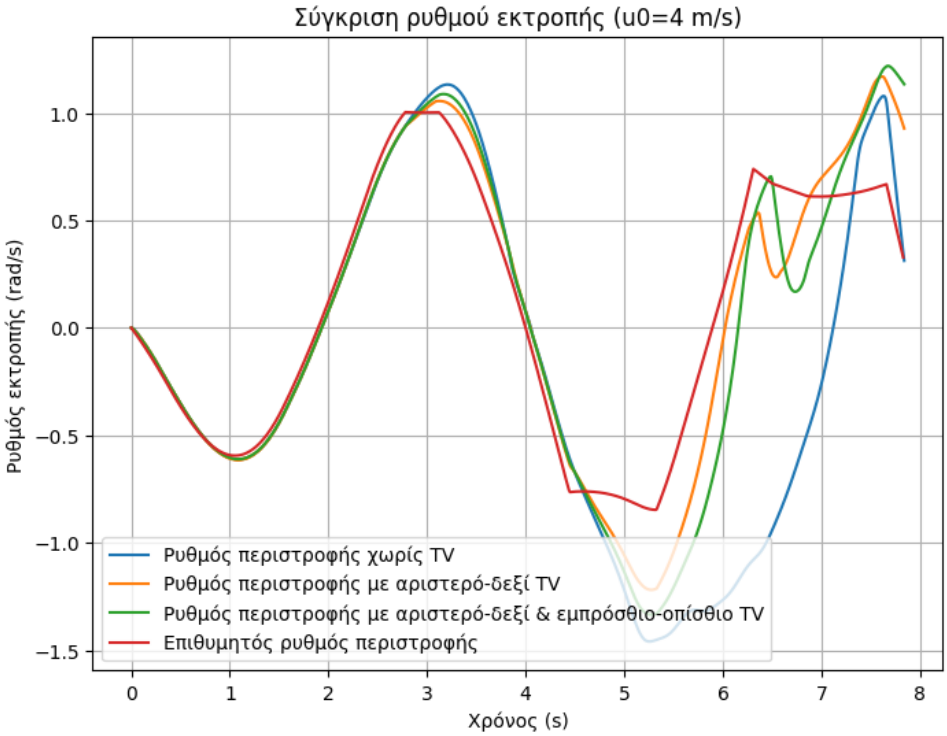


Σχήμα 5.86: Γωνία εκτροπής ελιγμού step-steer με επιβολή ακαριαίας γωνίας διεύθυνσης $\delta = 0.314\,rad$ και επιτάχυνση $a_{\text{long}} = 0.3g$

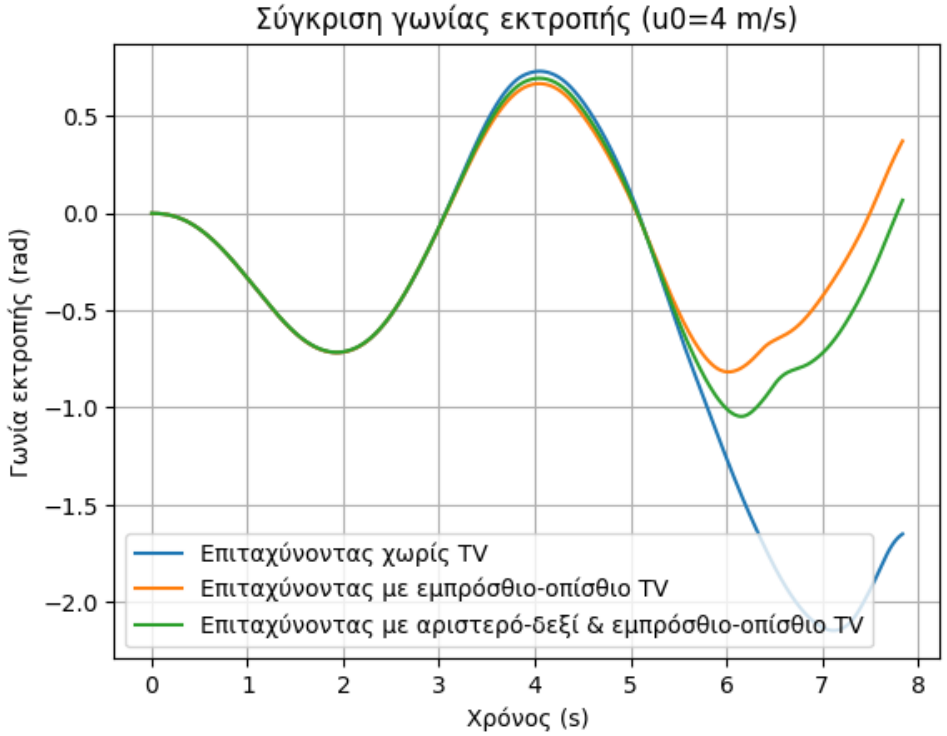


Σχήμα 5.87: Τροχιά ελιγμού step-steer με επιβολή ακαριαίας γωνίας διεύθυνσης $\delta = 0.314\,rad$ και επιτάχυνση $a_{\text{long}} = 0.3g$

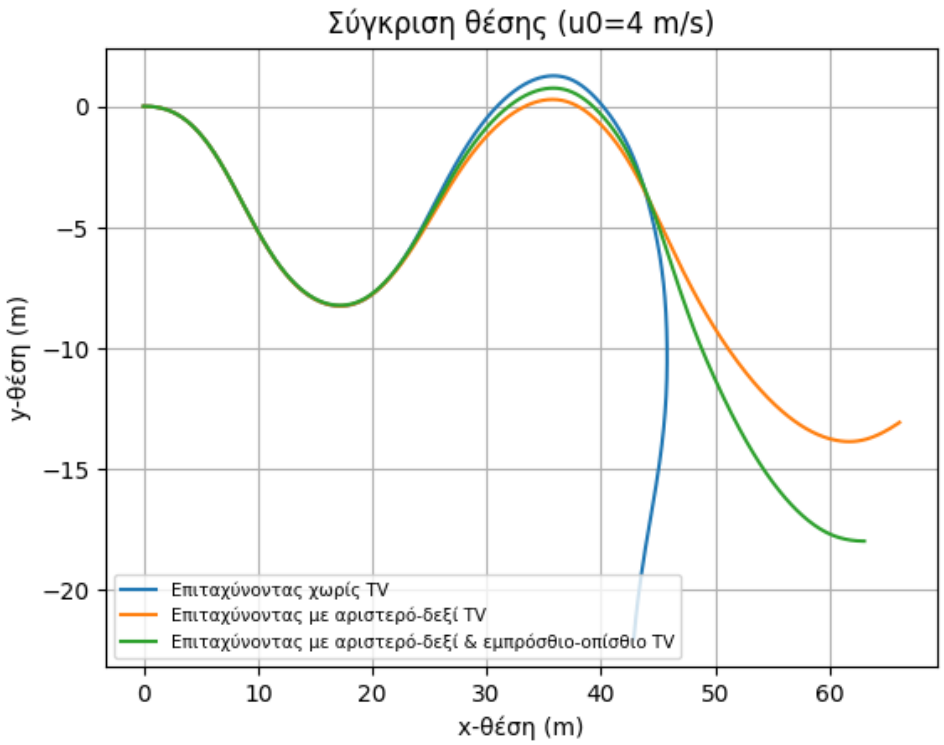
Για $v_0 = 15\,m/s$, το σύστημα καταφέρνει και πάλι να αποτρέψει την επικίνδυνη κατάσταση απώλειας ελέγχου, το οποίο φαίνεται στο σχήμα 5.85 του ρυθμού εκτροπής. Ωστόσο, η διαδικασία ελέγχου δρα λίγο διαφορετικά στην περίπτωση αυτή, με αποτέλεσμα να παρατηρείται αυξημένος ρυθμός εκτροπής έως τη στιγμή $t = 5.7\,s$, όπως φαίνεται στο σχήμα 5.85 και μειωμένος ύστερα. Αυτή η διαφορά αποτυπώνεται πολύ καθαρά και στο διάγραμμα γωνίας εκτροπής (Σχ. 5.86). Συνεπώς, παρατηρείται μεταβολή και στην τροχιά του σχήματος 5.87 υπό την επίδραση του συνδυαστικού TV, η οποία φέρει ένα πιο στρογγυλό σχήμα, αλλά παρουσιάζει και μία απότομη αλλαγή κατευθύνσεως προς το τέλος.



Σχήμα 5.88: Ρυθμός εκτροπής ελιγμού αποφυγής εμποδίου με επιτάχυνση $a_{\text{long}} = 0.2g$

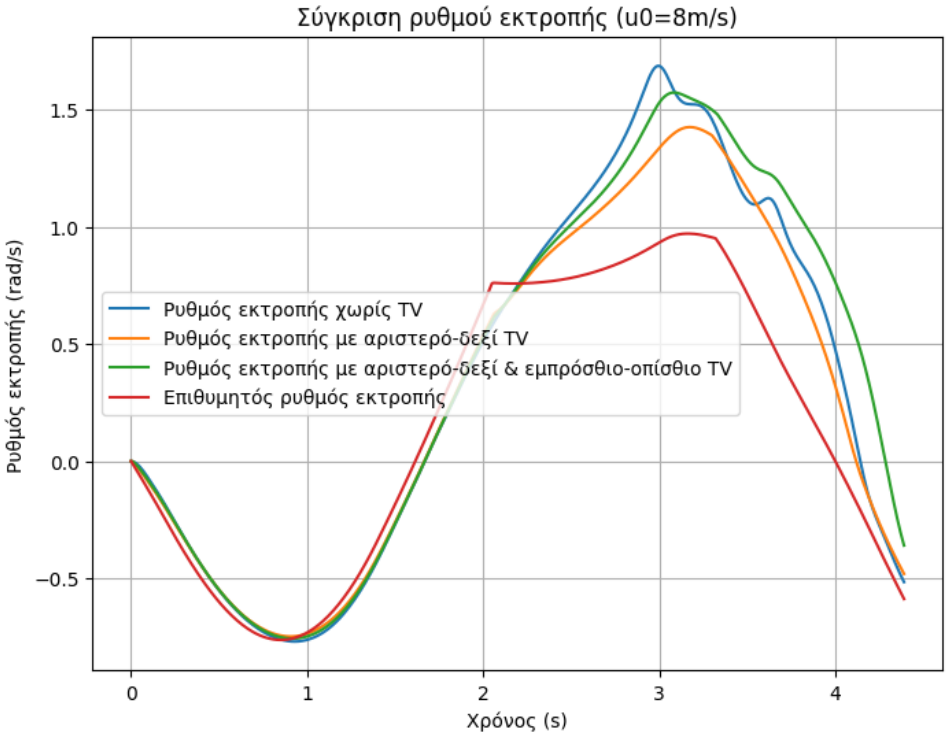


Σχήμα 5.89: Γωνία εκτροπής ελιγμού αποφυγής εμποδίου με επιτάχυνση $a_{\text{long}} = 0.2g$

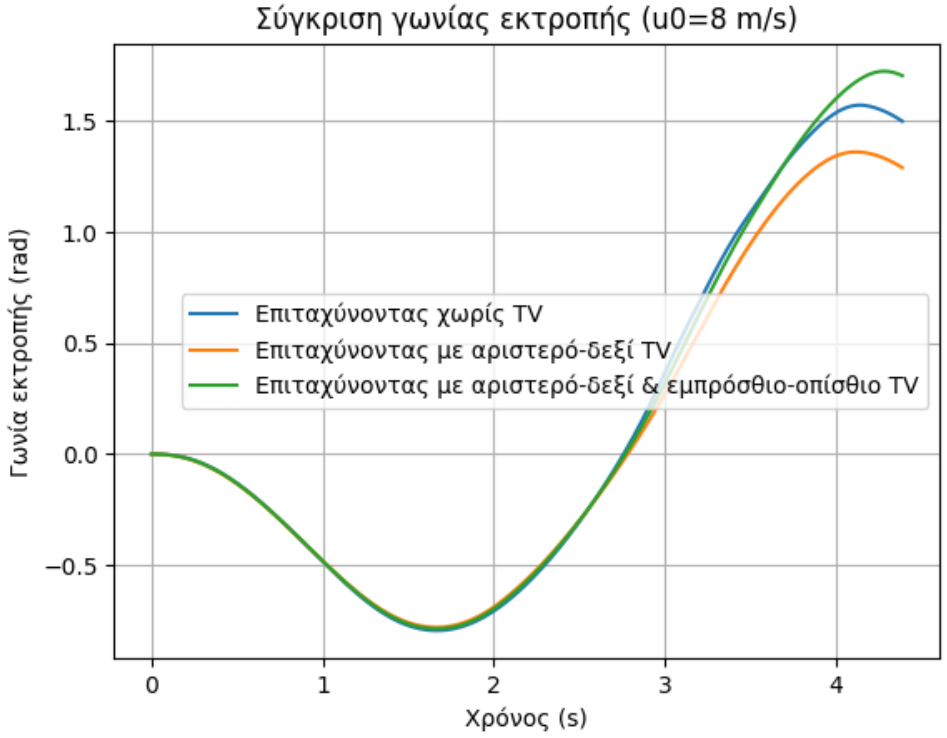


Σχήμα 5.90: Τροχιά ελιγμού αποφυγής εμποδίου με επιτάχυνση $a_{\text{long}} = 0.2g$

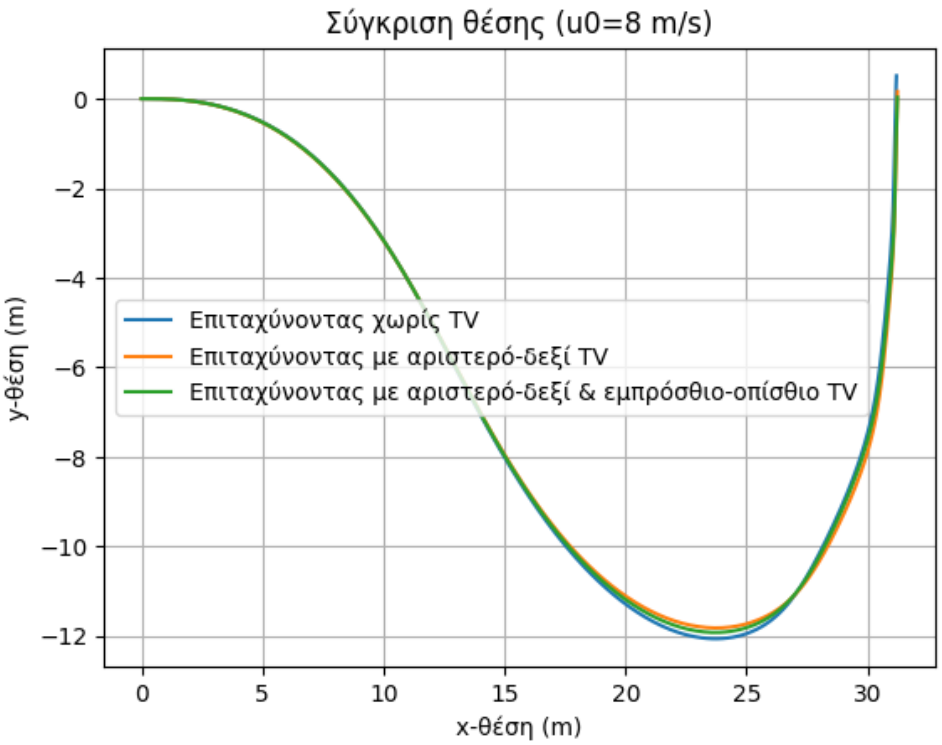
Στον ελιγμό αποφυγής εμποδίου με $v_0 = 4\,m/s$ δεν παρατηρήθηκε βελτίωση των αποτελεσμάτων σε σχέση με το αριστερό-δεξί TV, αφού ο ρυθμός εκτροπής που κατάφερε το όχημα δεν ήταν πιο κοντά στον επιθυμητό. Το παραπάνω φαίνεται στο σχήμα ρυθμού εκτροπής 5.88. Ωστόσο, το σύστημα καταπολέμησε ξανά το φαινόμενο υπερστροφής που είναι ορατό στο σχήμα 5.90 της τροχιάς του οχήματος.



Σχήμα 5.91: Ρυθμός εκτροπής ελιγμού αποφυγής εμποδίου με επιτάχυνση $a_{\text{long}} = 0.2g$

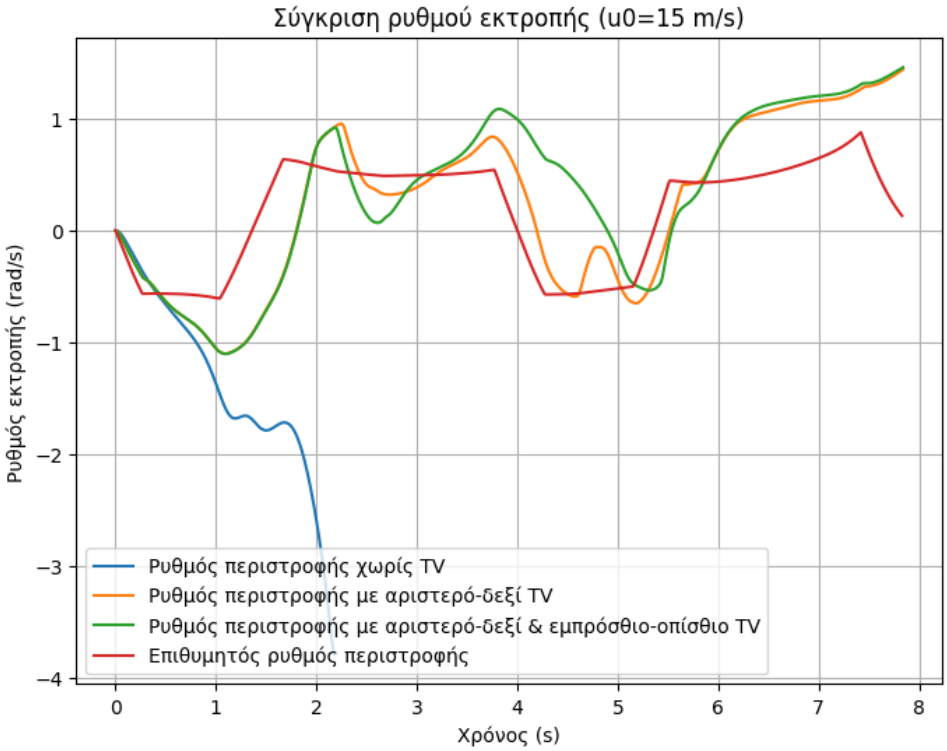


Σχήμα 5.92: Γωνία εκτροπής ελιγμού αποφυγής εμποδίου με επιτάχυνση $a_{\text{long}} = 0.2g$

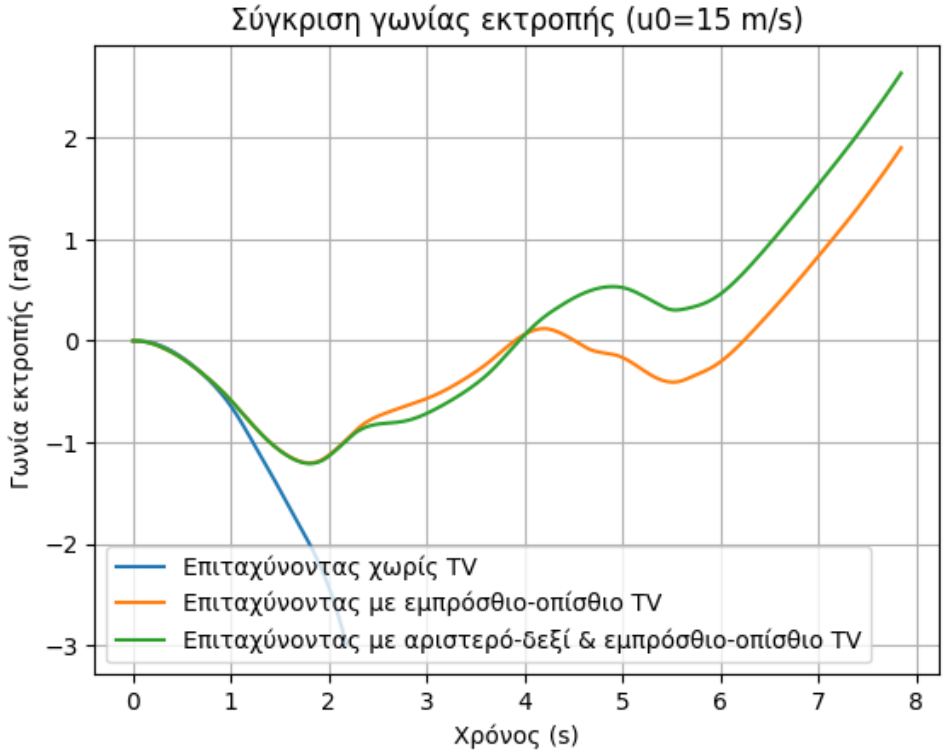


Σχήμα 5.93: Τροχιά ελιγμού αποφυγής εμποδίου με επιτάχυνση $a_{\text{long}} = 0.2g$

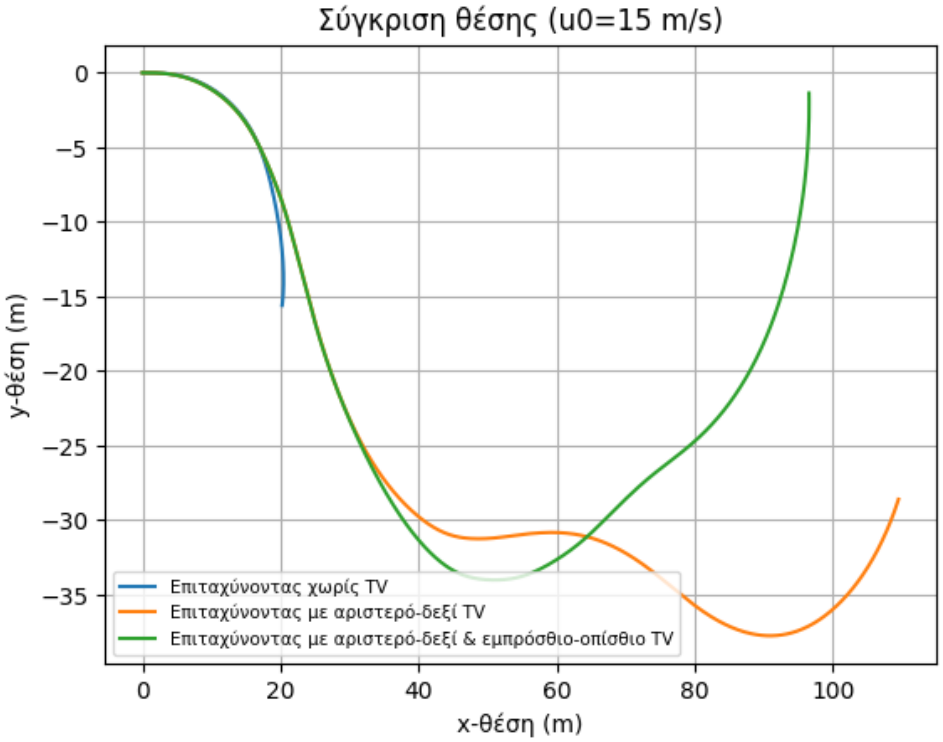
Ομοίως, για αρχική ταχύτητα $v_0 = 8\,m/s$ δεν υπήρξε περαιτέρω βελτίωση των αποτελεσμάτων. Παρατηρήθηκαν μερικές αποκλίσεις στα σχήματα του ρυθμού και της γωνίας εκτροπής (5.91 και 5.92).



Σχήμα 5.94: Ρυθμός εκτροπής ελιγμού αποφυγής εμποδίου με επιτάχυνση $a_{\text{long}} = 0.2g$



Σχήμα 5.95: Γωνία εκτροπής ελιγμού αποφυγής εμποδίου με επιτάχυνση $a_{\text{long}} = 0.2g$



Σχήμα 5.96: Τροχιά ελιγμού αποφυγής εμποδίου με επιτάχυνση $a_{\text{long}} = 0.2g$

Η συνδυαστική χρήση των συστημάτων TV αποδεικνύεται κερδοφόρα στον ελιγμό αποφυγής εμποδίου με την υψηλότερη αρχική ταχύτητα $v_0 = 15\,m/s$. Σύμφωνα με το Σχ. 5.96, το όχημα είναι σε θέση να επιστρέψει στη λωρίδα του επιτυχώς. Η μεταφορά ροπής στον οπίσθιο άξονα κατά την αριστερή στροφή οδήγησε σε επιπλέον δημιουργία ροπής στρέψης, κατευθύνοντας το όχημα πιο κοντά στην επιθυμητή πορεία.

Η τακτική αυτή παρουσιάζει ιδιαίτερο ενδιαφέρον. Η συμπεριφορά του συστήματος είναι πολύ παραπλήσια με αυτή του αριστερού-δεξιού TV, αποτρέποντας κάθε επικίνδυνη κατάσταση υπερστροφής ή υποστροφής. Η βασική διαφορά έγκειται στον τελευταίο ελιγμό, τον οποίο το αριστερό-δεξί TV δεν ήταν σε θέση να ολοκληρώσει. Εξετάζοντας το διάγραμμα τροχιάς του οχήματος στο Σχ. 5.96, παρατηρούνται πολύ πιο "επιθετικές" αλλαγές κατευθύνσεων.

Συμπεράσματα

Ένα από τα βασικά συμπεράσματα της παρούσας Διπλωματικής Εργασίας είναι ο κρίσιμος ρόλος του αριστερού-δεξιού Torque Vectoring στη σταθεροποίηση του οχήματος κατά τη διάρκεια διαφόρων ελιγμών. Η ικανότητά του να εξουδετερώνει τις τάσεις υπερστροφής και υποστροφής συνάδει με τις προσδοκίες για βελτιωμένη απόδοση του οχήματος, μειώνοντας αποτελεσματικά τον κίνδυνο πιθανής απώλειας ελέγχου. Σε πρακτικό επίπεδο, το αποτέλεσμα αυτό υπογραμμίζει τη σημασία της τελειοποίησης των στρατηγικών αριστερού-δεξιού TV για τη βελτιστοποίηση της δυναμικής του οχήματος.

Αντίθετα, η εφαρμογή του συστήματος εμπρόσθιου-οπίσθιου TV μεμονωμένα κατέδειξε μια αξιοσημείωτη πρόκληση. Η επιδείνωση των φαινομένων υπερστροφής και υποστροφής που προέκυψαν από τη μεταβολή της κατανομής της ροπής μεταξύ του εμπρόσθιου και του οπίσθιου άξονα τεκμηριώνει περαιτέρω την ακρίβεια και την προσοχή που απαιτείται κατά τη βαθμονόμησή του.

Το μεγαλύτερο ενδιαφέρον έγκειται στην αρμονική αλληλεπίδραση των συστημάτων αριστερού-δεξιού και εμπρόσθιου-οπίσθιου TV, όταν αυτά συνδυάζονται. Σε αυτή τη διαμόρφωση, οι δύο στρατηγικές αλληλοσυμπληρώνονται, αμβλύνοντας τις δυσμενείς επιδράσεις του εμπρόσθιου-οπίσθιου TV όταν αυτό δρα μόνο του. Οι βελτιωμένες επιδόσεις που παρατηρήθηκαν κατά τη διάρκεια των δοκιμών υπογραμμίζουν τη σημασία της συνδυαστικής λειτουργίας των στρατηγικών ελέγχου της κατανομής ροπής.

Καθώς ολοκληρώνεται η παρούσα έρευνα, είναι προφανές ότι ο δρόμος για την πλήρη αξιοποίηση των δυνατοτήτων του Torque Vectoring είναι γεμάτος με ευκαιρίες για περαιτέρω διερεύνηση και βελτίωση. Η ανάγκη για λεπτομερή ρύθμιση, βελτιστοποίηση των στρατηγικών και επίτευξη μιας αρμονικής ισορροπίας μεταξύ των συστατικών στοιχείων του συστήματος γίνεται ακόμη πιο εμφανής και τονίζει την αναγκαιότητα μιας συνεχούς δέσμευσης για την τελειοποίησή του, ανοίγοντας τελικά το δρόμο για ασφαλέστερες και πιο απολαυστικές οδηγικές εμπειρίες.

5.4 Μελλοντική εργασία

Επιτεύχθηκαν αξιόλογα αποτελέσματα από τη δημιουργία και την προσαρμογή ενός συστήματος αυτομάτου ελέγχου κατανομής ροπής στο όχημα. Ωστόσο, έπειτα από μελέτη των αποτελεσμάτων που παρουσιάστηκαν προηγουμένως, γεννήθηκαν σκέψεις όσον αφορά τους τρόπους βελτίωσης του συστήματος. Πιθανές κατευθύνσεις προς διερεύνηση και τακτικές που θα μπορούσαν μελλοντικά να εφαρμοστούν στο υφιστάμενο σύστημα προκειμένου να βελτιώσουν τη συμπεριφορά του, περιλαμβάνουν μεταξύ άλλων τις ακόλουθες:

- Χρήση μη-γραμμικού ελεγκτή για το εμπρόσθιο-οπίσθιο TV (βλ. Παράρτημα Α, Α.13). Με αυτόν τον τρόπο, η μεταφορά ισχύος μεταξύ του εμπρόσθιου και του οπίσθιου άξονα θα εκτελείται μόνο σε περιπτώσεις όπου το σφάλμα του ρυθμού εκτροπής υπερβαίνει ορισμένα όρια, αφού η έξοδος που θα παράγει για χαμηλές τιμές θα είναι μηδαμινή. Η φιλοσοφία πίσω από την πρόταση αυτή είναι η εξής: Μελετώντας τα αποτελέσματα του εμπρόσθιου-οπίσθιου TV, είναι σαφές ότι η τακτική αυτή αποδίδει περισσότερο σε περιπτώσεις όπου το όχημα βρίσκεται κοντά σε καταστάσεις απώλειας ελέγχου, άρα χρειάζεται επιπλέον δημιουργία διορθωτικής ροπής στρέψης προκειμένου να επιτευχθεί ο επιθυμητός ρυθμός εκτροπής.

- Ανάπτυξη πιο εξελιγμένου αλγορίθμου, ο οποίος θα λαμβάνει υπόψη και τη διαθέσιμη πρόσφυση κάθε τροχού λόγω μεταφοράς βάρους (μεταβολή κατακόρυφης δύναμης ελαστικών) κατά τη διάρκεια των ελιγμών. Ο αλγόριθμος που αναπτύχθηκε σε αυτή τη διατριβή λαμβάνει υπόψη μόνο τη διαθέσιμη πρόσφυση κάθε τροχού λόγω αλλαγών του συντελεστή τριβής, που μπορεί να προκληθεί από καιρικά φαινόμενα ή επιδείνωση ποιότητας του δρόμου (βλ. (3.12) και 2.4).

- Εισαγωγή ακόμη ενός μεγέθους ελέγχου, της γωνίας ολίσθησης (slip angle). Με την προσθήκη αυτή θα λαμβάνονται υπόψη ακόμα περισσότερες πληροφορίες για την κατάσταση που βρίσκεται το όχημα, βελτιώνοντας την αποδοτικότητα του αλγορίθμου.

Παράρτημα Α

Κώδικας

Απόσπασμα Κώδικα Α.1: Υλοποίηση μοντέλου ελαστικών σε Python (tire_model.py)

```python
# sign function
def sign(x):
    if x > 0:
        return 1
    elif x < 0:
        return -1
    else:
        return 0

# longitudinal tire forces
def formula_longitudinal(kappa, gamma, F_z, p):
    # turn slip is neglected, so xi_i=1
    # all scaling factors lambda = 1

    # coordinate system transformation
    kappa = -kappa

    S_hx = p.p_hx1
    S_vx = F_z * p.p_vx1

    kappa_x = kappa + S_hx
    mu_x = p.p_dx1 * (1 - p.p_dx3 * gamma ** 2)

    C_x = p.p_cx1
    D_x = mu_x * F_z
    E_x = p.p_ex1
    K_x = F_z * p.p_kx1
    B_x = K_x / (C_x * D_x)

     # magic tire formula
    return D_x * math.sin(C_x * math.atan(B_x * kappa_x - E_x
      ↪ * (B_x * kappa_x - math.atan(B_x * kappa_x))) + S_vx)

```

```
33
34  # lateral tire forces
35  def formula_lateral(alpha, gamma, F_z, p):
36      # turn slip is neglected, so xi_i=1
37      # all scaling factors lambda = 1
38
39      # coordinate system transformation
40      # alpha = -alpha
41
42      S_hy = sign(gamma) * (p.p_hy1 + p.p_hy3 * math.fabs(gamma)
            ↪ )
43      S_vy = sign(gamma) * F_z * (p.p_vy1 + p.p_vy3 * math.fabs(
            ↪ gamma))
44
45      alpha_y = alpha + S_hy
46      mu_y = p.p_dy1 * (1 - p.p_dy3 * gamma ** 2)
47
48      C_y = p.p_cy1
49      D_y = mu_y * F_z
50      E_y = p.p_ey1
51      K_y = F_z * p.p_ky1  # simplify K_y0 to p.p_ky1*F_z
52      B_y = K_y / (C_y * D_y)
53
54      # magic tire formula
55      F_y = D_y * math.sin(C_y * math.atan(B_y * alpha_y - E_y *
            ↪  (B_y * alpha_y - math.atan(B_y * alpha_y)))) + S_vy
56
57      res = []
58      res.append(F_y)
59      res.append(mu_y)
60      return res
61
62
63  # longitudinal tire forces for combined slip
64  def formula_longitudinal_comb(kappa, alpha, F0_x, p):
65      # turn slip is neglected, so xi_i=1
66      # all scaling factors lambda = 1
67
68      S_hxalpha = p.r_hx1
69
70      alpha_s = alpha + S_hxalpha
71
72      B_xalpha = p.r_bx1 * math.cos(math.atan(p.r_bx2 * kappa))
73      C_xalpha = p.r_cx1
74      E_xalpha = p.r_ex1
75      D_xalpha = F0_x / (math.cos(C_xalpha * math.atan(
76          B_xalpha * S_hxalpha - E_xalpha * (B_xalpha *
                ↪ S_hxalpha - math.atan(B_xalpha * S_hxalpha)))))
```

```
77
78      # magic tire formula
79      return D_xalpha * math.cos(
80          C_xalpha * math.atan(B_xalpha * alpha_s - E_xalpha * (
                ↪ B_xalpha * alpha_s - math.atan(B_xalpha * alpha_s
                ↪ ))))
81
82
83  # lateral tire forces for combined slip
84  def formula_lateral_comb(kappa, alpha, gamma, mu_y, F_z, F0_y,
        ↪  p):
85      # turn slip is neglected, so xi_i=1
86      # all scaling factors lambda = 1
87
88      S_hykappa = p.r_hy1
89
90      kappa_s = kappa + S_hykappa
91
92      B_ykappa = p.r_by1 * math.cos(math.atan(p.r_by2 * (alpha -
            ↪  p.r_by3)))
93      C_ykappa = p.r_cy1
94      E_ykappa = p.r_ey1
95      D_ykappa = F0_y / (math.cos(C_ykappa * math.atan(
96          B_ykappa * S_hykappa - E_ykappa * (B_ykappa *
                ↪ S_hykappa - math.atan(B_ykappa * S_hykappa)))))
97
98      D_vykappa = mu_y * F_z * (p.r_vy1 + p.r_vy3 * gamma) *
            ↪ math.cos(math.atan(p.r_vy4 * alpha))
99      S_vykappa = D_vykappa * math.sin(p.r_vy5 * math.atan(p.
            ↪ r_vy6 * kappa))
100
101     # magic tire formula
102     return D_ykappa * math.cos(C_ykappa * math.atan(
103         B_ykappa * kappa_s - E_ykappa * (B_ykappa * kappa_s -
                ↪ math.atan(B_ykappa * kappa_s)))) + S_vykappa
```

⌞ Πατήστε εδώ για να επιστρέψετε στο κυρίως κείμενο.

Απόσπασμα Κώδικα Α.2: Υλοποίηση των περιορισμών γωνίας διεύθυνσης σε Python (steering_constraints.py)

```python
def steering_constraints(steering_angle, steering_velocity, p):
    """
    steering_constraints - adjusts the steering velocity based on steering

    Inputs:
        :param steering_angle - steering angle
        :param steering_velocity - steering velocity
        :params p - steering parameter structure

    Outputs:
        :return steering_velocity - steering velocity
        """

    # steering limit reached?
    if (steering_angle <= p.min and steering_velocity <= 0) or (steering_angle >= p.max and steering_velocity >= 0):
        steering_velocity = 0
    elif steering_velocity <= p.v_min:
        steering_velocity = p.v_min
    elif steering_velocity >= p.v_max:
        steering_velocity = p.v_max

    return steering_velocity


def kappa_dot_dot_constraints(kappa_dot_dot, kappa_dot, p):
    """
    input constraints for kappa_dot_dot: adjusts kappa_dot_dot if kappa_dot limit (i.e., maximum curvature rate)
    or input bounds are reached
    """
    if (kappa_dot < -p.kappa_dot_max and kappa_dot_dot < 0.) \
            or (kappa_dot > p.kappa_dot_max and kappa_dot_dot > 0.):
        # kappa_dot limit reached
        kappa_dot_dot = 0.
    elif abs(kappa_dot_dot) >= p.kappa_dot_dot_max:
        # input bounds reached
        kappa_dot_dot = p.kappa_dot_dot_max
    return kappa_dot_dot
```

⌞ Πατήστε εδώ για να επιστρέψετε στο κυρίως κείμενο.

Απόσπασμα Κώδικα Α.3: Υλοποίηση των περιορισμών επιτάχυνσης σε Python (acceleration_constraints.py)

```python
def acceleration_constraints(velocity, acceleration, p):
    """
    accelerationConstraints - adjusts the acceleration based on acceleration constraints

    Inputs:
        :param acceleration - acceleration in driving direction
        :param velocity - velocity in driving direction
        :params p - longitudinal parameter structure

    Outputs:
        :return acceleration - acceleration in driving direction
        """
    # positive acceleration limit
    if velocity > p.v_switch:
        posLimit = p.a_max * p.v_switch / velocity
    else:
        posLimit = p.a_max

    # acceleration limit reached?
    if (velocity <= p.v_min and acceleration <= 0) or (velocity >= p.v_max and acceleration >= 0):
        acceleration = 0
    elif acceleration <= -p.a_max:
        acceleration = -p.a_max
    elif acceleration >= posLimit:
        acceleration = posLimit

    return acceleration


def jerk_dot_constraints(jerk_dot, jerk, p):
    """
    input constraints for jerk_dot: adjusts jerk_dot if jerk limit or input bounds are reached
    """
    if (jerk_dot < 0. and jerk <= -p.j_max) or (jerk_dot > 0. and jerk >= p.j_max):
        # jerk limit reached
        jerk_dot = 0.
    elif abs(jerk_dot) >= p.j_dot_max:
        # input bounds reached
```

```
39            jerk_dot = p.j_dot_max
40        return jerk_dot
```

└ Πατήστε εδώ για να επιστρέψετε στο κυρίως κείμενο.

Απόσπασμα Κώδικα Α.4: Υλοποίηση της δυναμικής του οχήματος σε Python (vehicle_dynamics_mb.py)

```
import math

from vehiclemodels.utils.steering_constraints import
    ↪ steering_constraints
from vehiclemodels.utils.acceleration_constraints import
    ↪ acceleration_constraints
from vehiclemodels.utils.vehicle_dynamics_ks_cog import
    ↪ vehicle_dynamics_ks_cog
import vehiclemodels.utils.tire_model as tireModel

def vehicle_dynamics_mb(x, uInit, p):
    """
    vehicleDynamics_mb - multi-body vehicle dynamics based on
        ↪ the DOT (department of transportation) vehicle
        ↪ dynamics
    reference point: center of mass

    Syntax:
        f = vehicleDynamics_mb(x,u,p)

    Inputs:
        :param x: vehicle state vector
        :param uInit: vehicle input vector
        :param p: vehicle parameter vector

    Outputs:
        :return f: right-hand side of differential equations
    """

    #------------- BEGIN CODE --------------

    # set gravity constant
    g = 9.81  #[m/s^2]

    #states
    #x1 = x-position in a global coordinate system
    #x2 = y-position in a global coordinate system
    #x3 = steering angle of front wheels
    #x4 = velocity in x-direction
    #x5 = yaw angle
    #x6 = yaw rate

    #x7 = roll angle
    #x8 = roll rate
```

```
    #x9 = pitch angle
    #x10 = pitch rate
    #x11 = velocity in y-direction
    #x12 = z-position
    #x13 = velocity in z-direction

    #x14 = roll angle front
    #x15 = roll rate front
    #x16 = velocity in y-direction front
    #x17 = z-position front
    #x18 = velocity in z-direction front

    #x19 = roll angle rear
    #x20 = roll rate rear
    #x21 = velocity in y-direction rear
    #x22 = z-position rear
    #x23 = velocity in z-direction rear

    #x24 = left front wheel angular speed
    #x25 = right front wheel angular speed
    #x26 = left rear wheel angular speed
    #x27 = right rear wheel angular speed

    #x28 = delta_y_f
    #x29 = delta_y_r

    #u1 = steering angle velocity of front wheels
    #u2 = acceleration

    #consider steering constraints
    u = []
    u.append(steering_constraints(x[2], uInit[0], p.steering))
       ↪  # different name u_init/u due to side effects of u
    #consider acceleration constraints
    u.append(acceleration_constraints(x[3], uInit[1], p.
       ↪ longitudinal)) # different name u_init/u due to side
       ↪ effects of u

    #compute slip angle at cg
    #switch to kinematic model for small velocities
    if abs(x[3]) < 0.1:
        beta = 0.
    else:
        beta = math.atan(x[10]/x[3])
    vel = math.sqrt(x[3]**2 + x[10]**2)
```

```
    #vertical tire forces
    F_z_LF = (x[16] + p.R_w*(math.cos(x[13]) - 1) - 0.5*p.T_f*
       ↪ math.sin(x[13]))*p.K_zt
    F_z_RF = (x[16] + p.R_w*(math.cos(x[13]) - 1) + 0.5*p.T_f*
       ↪ math.sin(x[13]))*p.K_zt
    F_z_LR = (x[21] + p.R_w*(math.cos(x[18]) - 1) - 0.5*p.T_r*
       ↪ math.sin(x[18]))*p.K_zt
    F_z_RR = (x[21] + p.R_w*(math.cos(x[18]) - 1) + 0.5*p.T_r*
       ↪ math.sin(x[18]))*p.K_zt

    #obtain individual tire speeds
    u_w_lf = (x[3] + 0.5*p.T_f*x[5])*math.cos(x[2]) + (x[10] +
       ↪  p.a*x[5])*math.sin(x[2])
    u_w_rf = (x[3] - 0.5*p.T_f*x[5])*math.cos(x[2]) + (x[10] +
       ↪  p.a*x[5])*math.sin(x[2])
    u_w_lr = x[3] + 0.5*p.T_r*x[5]
    u_w_rr = x[3] - 0.5*p.T_r*x[5]

    #negative wheel spin forbidden
    if u_w_lf < 0.0:
       u_w_lf *= 0

    if u_w_rf < 0.0:
       u_w_rf *= 0

    if u_w_lr < 0.0:
       u_w_lr *= 0

    if u_w_rr < 0.0:
       u_w_rr *= 0
    #compute longitudinal slip
    #switch to kinematic model for small velocities
    if abs(x[3]) < 0.1:
        s_lf = 0.
        s_rf = 0.
        s_lr = 0.
        s_rr = 0.
    else:
        s_lf = 1 - p.R_w*x[23]/u_w_lf
        s_rf = 1 - p.R_w*x[24]/u_w_rf
        s_lr = 1 - p.R_w*x[25]/u_w_lr
        s_rr = 1 - p.R_w*x[26]/u_w_rr

    #lateral slip angles
    #switch to kinematic model for small velocities
    if abs(x[3]) < 0.1:
        alpha_LF = 0.
        alpha_RF = 0.
```

```
        alpha_LR = 0.
        alpha_RR = 0.
    else:
        alpha_LF = math.atan((x[10] + p.a*x[5] - x[14]*(p.R_w
            ↪ - x[16]))/(x[3] + 0.5*p.T_f*x[5])) - x[2]
        alpha_RF = math.atan((x[10] + p.a*x[5] - x[14]*(p.R_w
            ↪ - x[16]))/(x[3] - 0.5*p.T_f*x[5])) - x[2]
        alpha_LR = math.atan((x[10] - p.b*x[5] - x[19]*(p.R_w
            ↪ - x[21]))/(x[3] + 0.5*p.T_r*x[5]))
        alpha_RR = math.atan((x[10] - p.b*x[5] - x[19]*(p.R_w
            ↪ - x[21]))/(x[3] - 0.5*p.T_r*x[5]))

    #auxiliary suspension movement
    z_SLF = (p.h_s - p.R_w + x[16] - x[11])/math.cos(x[6]) - p
        ↪ .h_s + p.R_w + p.a*x[8] + 0.5*(x[6] - x[13])*p.T_f
    z_SRF = (p.h_s - p.R_w + x[16] - x[11])/math.cos(x[6]) - p
        ↪ .h_s + p.R_w + p.a*x[8] - 0.5*(x[6] - x[13])*p.T_f
    z_SLR = (p.h_s - p.R_w + x[21] - x[11])/math.cos(x[6]) - p
        ↪ .h_s + p.R_w - p.b*x[8] + 0.5*(x[6] - x[18])*p.T_r
    z_SRR = (p.h_s - p.R_w + x[21] - x[11])/math.cos(x[6]) - p
        ↪ .h_s + p.R_w - p.b*x[8] - 0.5*(x[6] - x[18])*p.T_r

    dz_SLF = x[17] - x[12] + p.a*x[9] + 0.5*(x[7] - x[14])*p.
        ↪ T_f
    dz_SRF = x[17] - x[12] + p.a*x[9] - 0.5*(x[7] - x[14])*p.
        ↪ T_f
    dz_SLR = x[22] - x[12] - p.b*x[9] + 0.5*(x[7] - x[19])*p.
        ↪ T_r
    dz_SRR = x[22] - x[12] - p.b*x[9] - 0.5*(x[7] - x[19])*p.
        ↪ T_r

    #camber angles
    gamma_LF = x[6] + p.D_f*z_SLF + p.E_f*(z_SLF)**2
    gamma_RF = x[6] - p.D_f*z_SRF - p.E_f*(z_SRF)**2
    gamma_LR = x[6] + p.D_r*z_SLR + p.E_r*(z_SLR)**2
    gamma_RR = x[6] - p.D_r*z_SRR - p.E_r*(z_SRR)**2

    #compute longitudinal tire forces using the magic formula
        ↪ for pure slip
    F0_x_LF = tireModel.formula_longitudinal(s_lf, gamma_LF,
        ↪ F_z_LF, p.tire)
    F0_x_RF = tireModel.formula_longitudinal(s_rf, gamma_RF,
        ↪ F_z_RF, p.tire)
    F0_x_LR = tireModel.formula_longitudinal(s_lr, gamma_LR,
        ↪ F_z_LR, p.tire)
    F0_x_RR = tireModel.formula_longitudinal(s_rr, gamma_RR,
        ↪ F_z_RR, p.tire)

```

```
158     #compute lateral tire forces using the magic formula for
            ↪ pure slip
159     res = tireModel.formula_lateral(alpha_LF, gamma_LF, F_z_LF
            ↪ , p.tire)
160     F0_y_LF = res[0]
161     mu_y_LF = res[1]
162     res = tireModel.formula_lateral(alpha_RF, gamma_RF, F_z_RF
            ↪ , p.tire)
163     F0_y_RF = res[0]
164     mu_y_RF = res[1]
165     res = tireModel.formula_lateral(alpha_LR, gamma_LR, F_z_LR
            ↪ , p.tire)
166     F0_y_LR = res[0]
167     mu_y_LR = res[1]
168     res = tireModel.formula_lateral(alpha_RR, gamma_RR, F_z_RR
            ↪ , p.tire)
169     F0_y_RR = res[0]
170     mu_y_RR = res[1]
171
172     #compute longitudinal tire forces using the magic formula
            ↪ for combined slip
173     F_x_LF = tireModel.formula_longitudinal_comb(s_lf,
            ↪ alpha_LF, F0_x_LF, p.tire)
174     F_x_RF = tireModel.formula_longitudinal_comb(s_rf,
            ↪ alpha_RF, F0_x_RF, p.tire)
175     F_x_LR = tireModel.formula_longitudinal_comb(s_lr,
            ↪ alpha_LR, F0_x_LR, p.tire)
176     F_x_RR = tireModel.formula_longitudinal_comb(s_rr,
            ↪ alpha_RR, F0_x_RR, p.tire)
177
178     #compute lateral tire forces using the magic formula for
            ↪ combined slip
179     F_y_LF = tireModel.formula_lateral_comb(s_lf, alpha_LF,
            ↪ gamma_LF, mu_y_LF, F_z_LF, F0_y_LF, p.tire)
180     F_y_RF = tireModel.formula_lateral_comb(s_rf, alpha_RF,
            ↪ gamma_RF, mu_y_RF, F_z_RF, F0_y_RF, p.tire)
181     F_y_LR = tireModel.formula_lateral_comb(s_lr, alpha_LR,
            ↪ gamma_LR, mu_y_LR, F_z_LR, F0_y_LR, p.tire)
182     F_y_RR = tireModel.formula_lateral_comb(s_rr, alpha_RR,
            ↪ gamma_RR, mu_y_RR, F_z_RR, F0_y_RR, p.tire)
183
184     #auxiliary movements for compliant joint equations
185     delta_z_f = p.h_s - p.R_w + x[16] - x[11]
186     delta_z_r = p.h_s - p.R_w + x[21] - x[11]
187
188     delta_phi_f = x[6] - x[13]
189     delta_phi_r = x[6] - x[18]
190
```

```
    dot_delta_phi_f = x[7] - x[14]
    dot_delta_phi_r = x[7] - x[19]

    dot_delta_z_f = x[17] - x[12]
    dot_delta_z_r = x[22] - x[12]

    dot_delta_y_f = x[10] + p.a*x[5] - x[15]
    dot_delta_y_r = x[10] - p.b*x[5] - x[20]

    delta_f = delta_z_f*math.sin(x[6]) - x[27]*math.cos(x[6])
       ↪ - (p.h_raf - p.R_w)*math.sin(delta_phi_f)
    delta_r = delta_z_r*math.sin(x[6]) - x[28]*math.cos(x[6])
       ↪ - (p.h_rar - p.R_w)*math.sin(delta_phi_r)

    dot_delta_f = (delta_z_f*math.cos(x[6]) + x[27]*math.sin(x
       ↪ [6]))*x[7] + dot_delta_z_f*math.sin(x[6]) -
       ↪ dot_delta_y_f*math.cos(x[6]) - (p.h_raf - p.R_w)*math
       ↪ .cos(delta_phi_f)*dot_delta_phi_f
    dot_delta_r = (delta_z_r*math.cos(x[6]) + x[28]*math.sin(x
       ↪ [6]))*x[7] + dot_delta_z_r*math.sin(x[6]) -
       ↪ dot_delta_y_r*math.cos(x[6]) - (p.h_rar - p.R_w)*math
       ↪ .cos(delta_phi_r)*dot_delta_phi_r

    #compliant joint forces
    F_RAF = delta_f*p.K_ras + dot_delta_f*p.K_rad
    F_RAR = delta_r*p.K_ras + dot_delta_r*p.K_rad

    #auxiliary suspension forces (bump stop neglected  squat/
       ↪ lift forces neglected)
    F_SLF = p.m_s*g*p.b/(2*(p.a+p.b)) - z_SLF*p.K_sf - dz_SLF*
       ↪ p.K_sdf + (x[6] - x[13])*p.K_tsf/p.T_f

    F_SRF = p.m_s*g*p.b/(2*(p.a+p.b)) - z_SRF*p.K_sf - dz_SRF*
       ↪ p.K_sdf - (x[6] - x[13])*p.K_tsf/p.T_f

    F_SLR = p.m_s*g*p.a/(2*(p.a+p.b)) - z_SLR*p.K_sr - dz_SLR*
       ↪ p.K_sdr + (x[6] - x[18])*p.K_tsr/p.T_r

    F_SRR = p.m_s*g*p.a/(2*(p.a+p.b)) - z_SRR*p.K_sr - dz_SRR*
       ↪ p.K_sdr - (x[6] - x[18])*p.K_tsr/p.T_r


    #auxiliary variables sprung mass
    sumX = F_x_LR + F_x_RR + (F_x_LF + F_x_RF)*math.cos(x[2])
       ↪ - (F_y_LF + F_y_RF)*math.sin(x[2])

    sumN = (F_y_LF + F_y_RF)*p.a*math.cos(x[2]) + (F_x_LF +
       ↪ F_x_RF)*p.a*math.sin(x[2]) \
```

```
224             + (F_y_RF - F_y_LF)*0.5*p.T_f*math.sin(x[2]) + (
                    ↪ F_x_LF - F_x_RF)*0.5*p.T_f*math.cos(x[2]) \
225             + (F_x_LR - F_x_RR)*0.5*p.T_r - (F_y_LR + F_y_RR)*p
                    ↪ .b
226
227     sumY_s = (F_RAF + F_RAR)*math.cos(x[6]) + (F_SLF + F_SLR +
            ↪  F_SRF + F_SRR)*math.sin(x[6])
228
229     sumL = 0.5*F_SLF*p.T_f + 0.5*F_SLR*p.T_r - 0.5*F_SRF*p.T_f
            ↪  - 0.5*F_SRR*p.T_r \
230             - F_RAF/math.cos(x[6])*(p.h_s - x[11] - p.R_w + x
                    ↪ [16] - (p.h_raf - p.R_w)*math.cos(x[13])) \
231             - F_RAR/math.cos(x[6])*(p.h_s - x[11] - p.R_w + x
                    ↪ [21] - (p.h_rar - p.R_w)*math.cos(x[18]))
232
233     sumZ_s = (F_SLF + F_SLR + F_SRF + F_SRR)*math.cos(x[6]) -
            ↪ (F_RAF + F_RAR)*math.sin(x[6])
234
235     sumM_s = p.a*(F_SLF + F_SRF) - p.b*(F_SLR + F_SRR) + ((
            ↪ F_x_LF + F_x_RF)*math.cos(x[2]) \
236             - (F_y_LF + F_y_RF)*math.sin(x[2]) + F_x_LR +
                    ↪ F_x_RR)*(p.h_s - x[11])
237
238     #auxiliary variables unsprung mass
239     sumL_uf = 0.5*F_SRF*p.T_f - 0.5*F_SLF*p.T_f - F_RAF*(p.
            ↪ h_raf - p.R_w) \
240                 + F_z_LF*(p.R_w*math.sin(x[13]) + 0.5*p.T_f*math
                        ↪ .cos(x[13]) - p.K_lt*F_y_LF) \
241                 - F_z_RF*(-p.R_w*math.sin(x[13]) + 0.5*p.T_f*
                        ↪ math.cos(x[13]) + p.K_lt*F_y_RF) \
242                 - ((F_y_LF + F_y_RF)*math.cos(x[2]) + (F_x_LF +
                        ↪ F_x_RF)*math.sin(x[2]))*(p.R_w - x[16])
243
244     sumL_ur = 0.5*F_SRR*p.T_r - 0.5*F_SLR*p.T_r - F_RAR*(p.
            ↪ h_rar - p.R_w) \
245                 + F_z_LR*(p.R_w*math.sin(x[18]) + 0.5*p.T_r*math
                        ↪ .cos(x[18]) - p.K_lt*F_y_LR) \
246                 - F_z_RR*(-p.R_w*math.sin(x[18]) + 0.5*p.T_r*
                        ↪ math.cos(x[18]) + p.K_lt*F_y_RR) \
247                 - (F_y_LR + F_y_RR)*(p.R_w - x[21])
248
249     sumZ_uf = F_z_LF + F_z_RF + F_RAF*math.sin(x[6]) - (F_SLF
            ↪ + F_SRF)*math.cos(x[6])
250
251     sumZ_ur = F_z_LR + F_z_RR + F_RAR*math.sin(x[6]) - (F_SLR
            ↪ + F_SRR)*math.cos(x[6])
252
253     sumY_uf = (F_y_LF + F_y_RF)*math.cos(x[2]) + (F_x_LF +
```

```
        ↪ F_x_RF)*math.sin(x[2]) \
254             - F_RAF*math.cos(x[6]) - (F_SLF + F_SRF)*math.
                  ↪ sin(x[6])
255
256     sumY_ur = (F_y_LR + F_y_RR) \
257             - F_RAR*math.cos(x[6]) - (F_SLR + F_SRR)*math.
                  ↪ sin(x[6])
258
259
260     #dynamics common with single-track model
261     f = [] # init 'right hand side'
262     #switch to kinematic model for small velocities
263     if abs(x[3]) < 0.1:
264         #wheelbase
265         # lwb = p.a + p.b
266
267         #system dynamics
268         # x_ks = [x[0],  x[1],  x[2],  x[3],  x[4]]
269         # f_ks = vehicle_dynamics_ks(x_ks, u, p)
270         # f.extend(f_ks)
271         # f.append(u[1]*lwb*math.tan(x[2]) + x[3]/(lwb*math.
              ↪ cos(x[2])**2)*u[0])
272
273         # Use kinematic model with reference point at center
              ↪ of mass
274         # wheelbase
275         lwb = p.a + p.b
276         # system dynamics
277         x_ks = [x[0], x[1], x[2], x[3], x[4]]
278         # kinematic model
279         f_ks = vehicle_dynamics_ks_cog(x_ks, u, p)
280         f = [f_ks[0], f_ks[1], f_ks[2], f_ks[3], f_ks[4]]
281         # derivative of slip angle and yaw rate
282         d_beta = (p.b * u[0]) / (lwb * math.cos(x[2]) ** 2 *
              ↪ (1 + (math.tan(x[2]) ** 2 * p.b / lwb) ** 2))
283         dd_psi = 1 / lwb * (u[1] * math.cos(x[6]) * math.tan(x
              ↪ [2]) -
284                             x[3] * math.sin(x[6]) * d_beta *
                                  ↪ math.tan(x[2]) +
285                             x[3] * math.cos(x[6]) * u[0] /
                                  ↪ math.cos(x[2]) ** 2)
286         f.append(dd_psi)
287
288     else:
289         f.append(math.cos(beta + x[4])*vel)
290         f.append(math.sin(beta + x[4])*vel)
291         f.append(u[0])
292         f.append(1/p.m*sumX + x[5]*x[10])
```

```
293         f.append(x[5])
294         f.append(1/(p.I_z - (p.I_xz_s)**2/p.I_Phi_s)*(sumN + p
                ↪ .I_xz_s/p.I_Phi_s*sumL))
295
296
297     # remaining sprung mass dynamics
298     f.append(x[7])
299     f.append(1/(p.I_Phi_s - (p.I_xz_s)**2/p.I_z)*(p.I_xz_s/p.
            ↪ I_z*sumN + sumL))
300     f.append(x[9])
301     f.append(1/p.I_y_s*sumM_s)
302     f.append(1/p.m_s*sumY_s - x[5]*x[3])
303     f.append(x[12])
304     f.append(g - 1/p.m_s*sumZ_s)
305
306     #unsprung mass dynamics (front)
307     f.append(x[14])
308     f.append(1/p.I_uf*sumL_uf)
309     f.append(1/p.m_uf*sumY_uf - x[5]*x[3])
310     f.append(x[17])
311     f.append(g - 1/p.m_uf*sumZ_uf)
312
313     #unsprung mass dynamics (rear)
314     f.append(x[19])
315     f.append(1/p.I_ur*sumL_ur)
316     f.append(1/p.m_ur*sumY_ur - x[5]*x[3])
317     f.append(x[22])
318     f.append(g - 1/p.m_ur*sumZ_ur)
319
320     #convert acceleration input to brake and engine torque
321     if u[1]>0:
322         T_B = 0.0
323         T_E = p.m*p.R_w*u[1]
324     else:
325         T_B = p.m*p.R_w*u[1]
326         T_E = 0.
327
328
329
330     #wheel dynamics (p.T  new parameter for torque splitting)
331     f.append(1/p.I_y_w*(-p.R_w*F_x_LF + 0.5*p.T_sb*T_B + 0.5*p
            ↪ .T_se*T_E))
332     f.append(1/p.I_y_w*(-p.R_w*F_x_RF + 0.5*p.T_sb*T_B + 0.5*p
            ↪ .T_se*T_E))
333     f.append(1/p.I_y_w*(-p.R_w*F_x_LR + 0.5*(1-p.T_sb)*T_B +
            ↪ 0.5*(1-p.T_se)*T_E))
334     f.append(1/p.I_y_w*(-p.R_w*F_x_RR + 0.5*(1-p.T_sb)*T_B +
            ↪ 0.5*(1-p.T_se)*T_E))
```

```
335
336     #negative wheel spin forbidden
337     for iState in range(23, 27):
338         if x[iState] < 0.0:
339             x[iState] = 0.0
340             f[iState] = 0.0
341
342     #compliant joint equations
343     f.append(dot_delta_y_f)
344     f.append(dot_delta_y_r)
345
346     return f
```

⌞ Πατήστε εδώ για να επιστρέψετε στο κυρίως κείμενο.

Απόσπασμα Κώδικα Α.5: Παράμετροι οχήματος (parameters_vehicle2.yaml)

```yaml
# parameters_vehicle2 - parameter set of the multi-body vehicle dynamics
# based on the DOT (department of transportation) vehicle dynamics
# values are taken from a BMW 320i
# vehicle body dimensions
# vehicle length [m]
l: 4.508
# vehicle width [m]
w: 1.61

# steering constraints
steering:
  # minimum steering angle [rad]
  max: 1.066
  # maximum steering angle [rad]
  min: -1.066
  # minimum steering velocity [rad/s]
  v_max: 0.4
  # maximum steering velocity [rad/s]
  v_min: -0.4
  # maximum curvature change
  kappa_dot_max: 0.4
  # maximum curvature rate rate
  kappa_dot_dot_max: 20

# longitudinal constraints
longitudinal:
  # maximum absolute acceleration [m/s^2]
  a_max: 11.5
  # maximum longitudinal jerk [m/s^3]
  j_max: 10.0e+3
  # maximum longitudinal jerk change [m/s^4]
  j_dot_max: 10.0e3
  # maximum velocity [m/s]
  v_max: 50.8
  # minimum velocity [m/s]
  v_min: -13.9
  # switching velocity [m/s]
  v_switch: 7.319

# masses
# vehicle mass [kg]  MASS
m: 1093.2952334674046
# sprung mass [kg]  SMASS
```

```
m_s: 965.7108098804363
# unsprung mass front [kg]  UMASSF
m_uf: 63.7921826056784
# unsprung mass rear [kg]  UMASSR
m_ur: 63.7921826056784

# axes distances
# distance from spring mass center of gravity to front axle [m] ↪  LENA
a: 1.1561957064
# distance from spring mass center of gravity to rear axle [m] ↪  LENB
b: 1.4227170936

# moments of inertia of sprung mass
# moment of inertia for sprung mass in roll [kg m^2]  IXS
I_Phi_s: 207.26524557936952
# moment of inertia for sprung mass in pitch [kg m^2]  IYS
I_y_s: 1565.8178787125541
# moment of inertia for sprung mass in yaw [kg m^2]  IZZ
I_z: 1791.5995300122856
# moment of inertia cross product [kg m^2]  IXZ
I_xz_s: 0.0

# suspension parameters
# suspension spring rate (front) [N/m]  KSF
K_sf: 24453.137879749014
# suspension damping rate (front) [N s/m]  KSDF
K_sdf: 1786.2441002440723
# suspension spring rate (rear) [N/m]  KSR
K_sr: 19635.504745231297
# suspension damping rate (rear) [N s/m]  KSDR
K_sdr: 1649.0833034887382

# geometric parameters
# track width front [m]  TRWF
T_f: 1.38684
# track width rear [m]  TRWB
T_r: 1.36398
# lateral spring rate at compliant compliant pin joint between ↪  M_s and M_u [N/m]  KRAS
K_ras: 175186.65943700788

# auxiliary torsion roll stiffness per axle (normally negative ↪ ) (front) [N m/rad]  KTSF
K_tsf: -6914.881688272133
# auxiliary torsion roll stiffness per axle (normally negative ↪ ) (rear) [N m/rad]  KTSR
```

```
K_tsr: -2643.6009520155308
# damping rate at compliant compliant pin joint between M_s and M_u [N s/m]  KRADP
K_rad: 10215.732056044453
# vertical spring rate of tire [N/m]  KZT
K_zt: 158294.1398119115

# center of gravity height of total mass [m]  HCG (mainly required for conversion to other vehicle models)
h_cg: 0.5748689544000001
# height of roll axis above ground (front) [m]  HRAF
h_raf: 0.0
# height of roll axis above ground (rear) [m]  HRAR
h_rar: 0.0

# M_s center of gravity above ground [m]  HS
h_s: 0.61373004

# moment of inertia for unsprung mass about x-axis (front) [kg m^2]  IXUF
I_uf: 30.673279563178017
# moment of inertia for unsprung mass about x-axis (rear) [kg m^2]  IXUR
I_ur: 29.670408143156248
# wheel inertia, from internet forum for 235/65 R 17 [kg m^2]
I_y_w: 1.7

# lateral compliance rate of tire, wheel, and suspension, per tire [m/N]  KLT
K_lt: 1.6430724599974725e-05
# effective wheel/tire radius  chosen as tire rolling radius RR  taken from ADAMS documentation [m]
R_w: 0.344

# split of brake and engine torque
T_sb: 0.66
T_se: 0

# suspension parameters
# [rad/m]  DF
D_f: -0.39370078740157477
# [rad/m]  DR
D_r: -0.905511811023622
# [needs conversion if nonzero]  EF
E_f: 0
# [needs conversion if nonzero]  ER
E_r: 0
```

⌞ Πατήστε εδώ για να επιστρέψετε στο κυρίως κείμενο.

Απόσπασμα Κώδικα A.6: Παράμετροι ελαστικών (parameters_tire.yaml)

```yaml
# tire parameters from ADAMS handbook

tire:
  # longitudinal coefficients
  p_cx1: 1.6411
  p_dx1: 1.1739
  p_dx3: 0
  p_ex1: 0.46403
  p_kx1: 22.303
  p_hx1: 0.0012297
  p_vx1: -8.8098e-06
  r_bx1: 13.276
  r_bx2: -13.778
  r_cx1: 1.2568
  r_ex1: 0.65225
  r_hx1: 0.0050722

# lateral coefficients
  p_cy1: 1.3507
  p_dy1: 1.0489
  p_dy3: -2.8821
  p_ey1: -0.0074722
  p_ky1: -21.92
  p_hy1: 0.0026747
  p_hy3: 0.031415
  p_vy1: 0.037318
  p_vy3: -0.32931
  r_by1: 7.1433
  r_by2: 9.1916
  r_by3: -0.027856
  r_cy1: 1.0719
  r_ey1: -0.27572
  r_hy1: 5.7448e-06
  r_vy1: -0.027825
  r_vy3: -0.27568
  r_vy4: 12.12
  r_vy5: 1.9
  r_vy6: -10.704
```

⌞ Πατήστε εδώ για να επιστρέψετε στο κυρίως κείμενο.

Απόσπασμα Κώδικα Α.7: Προσομοίωση αριστερής στροφής με $v_0 = 15\,m/s$, $v_\delta = 0.005\,rad/s$ και $a_{\text{long}} = 0.2g$

```
from scipy.integrate import odeint
import numpy as np
import matplotlib.pyplot as plt
import math

from vehiclemodels.parameters_vehicle2 import parameters_vehicle2
from vehiclemodels.init_mb import init_mb
from vehiclemodels.vehicle_dynamics_mb import vehicle_dynamics_mb

def func_MB(x, t, u, p):
    f = vehicle_dynamics_mb(x, u, p)
    return f

# load parameters
p = parameters_vehicle2()
g = 9.81  # [m/s^2]

# set options
# ------------------------------------------------------------------

tStart = 0  # start time
tFinal = 8  # end time
dt = 0.01  # time steps

delta0 = 0
vel0 = 15
Psi0 = 0
dotPsi0 = 0
beta0 = 0
sy0 = 0
initialState = [0, sy0, delta0, vel0, Psi0, dotPsi0, beta0]  # initial state for simulation
x0_MB = init_mb(initialState, p)  # initial state for multi-body model

time = np.arange(0, tFinal, dt)
v_delta = 0.005

# accelerating
u = [v_delta, 0.2 * g]
x_acc = odeint(func_MB, x0_MB, time, args=(u, p))

```

```
# position
plt.title('Θέση')
plt.plot([tmp[0] for tmp in x_acc], [tmp[1] for tmp in x_acc])
plt.legend(['Επιταχυνόμενη αριστερή στροφή'])
plt.xlabel('x-θέση (m)')
plt.ylabel('y-θέση (m)')
plt.show()

# orientation
plt.title('Γωνία εκτροπής')
plt.plot(time, [tmp[4] for tmp in x_acc])
plt.legend(['Επιταχυνόμενη αριστερή στροφή'])
plt.xlabel('Χρόνος (s)')
plt.ylabel('Γωνία εκτροπής (rad)')
plt.show()

# yaw rate
plt.title('Ρυθμός εκτροπής')
plt.plot(time, [tmp[5] for tmp in x_acc])
plt.legend(['Επιταχυνόμενη αριστερή στροφή'])
plt.xlabel('Χρόνος (s)')
plt.ylabel('Ρυθμός εκτροπής (rad/s)')
plt.show()
```

⌞ Πατήστε εδώ για να επιστρέψετε στο κυρίως κείμενο.

Απόσπασμα Κώδικα Α.8: Ενημέρωση του αρχείου δυναμικής του οχήματος (vehicle_dynamics_mb.py) για την υποστήριξη TV

```
import math

from vehiclemodels.utils.steering_constraints import
    ↪ steering_constraints
from vehiclemodels.utils.acceleration_constraints import
    ↪ acceleration_constraints
from vehiclemodels.utils.vehicle_dynamics_ks_cog import
    ↪ vehicle_dynamics_ks_cog
import vehiclemodels.utils.tire_model as tireModel


def vehicle_dynamics_mb(x, uInit, p):
    """
    vehicleDynamics_mb - multi-body vehicle dynamics based on
        ↪ the DOT (department of transportation) vehicle
        ↪ dynamics
    reference point: center of mass

    Syntax:
        f = vehicleDynamics_mb(x,u,p)

    Inputs:
        :param x: vehicle state vector
        :param uInit: vehicle input vector
        :param p: vehicle parameter vector

    Outputs:
        :return f: right-hand side of differential equations
    """
    # set gravity constant
    g = 9.81  #[m/s^2]

    #states
    #x1 = x-position in a global coordinate system
    #x2 = y-position in a global coordinate system
    #x3 = steering angle of front wheels
    #x4 = velocity in x-direction
    #x5 = yaw angle
    #x6 = yaw rate

    #x7 = roll angle
    #x8 = roll rate
    #x9 = pitch angle
    #x10 = pitch rate
```

```
    #x11 = velocity in y-direction
    #x12 = z-position
    #x13 = velocity in z-direction

    #x14 = roll angle front
    #x15 = roll rate front
    #x16 = velocity in y-direction front
    #x17 = z-position front
    #x18 = velocity in z-direction front

    #x19 = roll angle rear
    #x20 = roll rate rear
    #x21 = velocity in y-direction rear
    #x22 = z-position rear
    #x23 = velocity in z-direction rear

    #x24 = left front wheel angular speed
    #x25 = right front wheel angular speed
    #x26 = left rear wheel angular speed
    #x27 = right rear wheel angular speed

    #x28 = delta_y_f
    #x29 = delta_y_r

    #u1 = steering angle velocity of front wheels
    #u2 = acceleration
    #u3 = left-rear corrective torque

    #consider steering constraints
    u = []
    u.append(steering_constraints(x[2], uInit[0], p.steering))
       ↪  # different name u_init/u due to side effects of u
    #consider acceleration constraints
    u.append(acceleration_constraints(x[3], uInit[1], p.
       ↪ longitudinal)) # different name u_init/u due to side
       ↪ effects of u
    u.append(uInit[2])

    #compute slip angle at cg
    #switch to kinematic model for small velocities
    if abs(x[3]) < 0.1:
        beta = 0.
    else:
        beta = math.atan(x[10]/x[3])
    vel = math.sqrt(x[3]**2 + x[10]**2)
```

```
85      #vertical tire forces
86      F_z_LF = (x[16] + p.R_w*(math.cos(x[13]) - 1) - 0.5*p.T_f*
           ↪ math.sin(x[13]))*p.K_zt
87      F_z_RF = (x[16] + p.R_w*(math.cos(x[13]) - 1) + 0.5*p.T_f*
           ↪ math.sin(x[13]))*p.K_zt
88      F_z_LR = (x[21] + p.R_w*(math.cos(x[18]) - 1) - 0.5*p.T_r*
           ↪ math.sin(x[18]))*p.K_zt
89      F_z_RR = (x[21] + p.R_w*(math.cos(x[18]) - 1) + 0.5*p.T_r*
           ↪ math.sin(x[18]))*p.K_zt
90
91      #obtain individual tire speeds
92      u_w_lf = (x[3] + 0.5*p.T_f*x[5])*math.cos(x[2]) + (x[10] +
           ↪  p.a*x[5])*math.sin(x[2])
93      u_w_rf = (x[3] - 0.5*p.T_f*x[5])*math.cos(x[2]) + (x[10] +
           ↪  p.a*x[5])*math.sin(x[2])
94      u_w_lr = x[3] + 0.5*p.T_r*x[5]
95      u_w_rr = x[3] - 0.5*p.T_r*x[5]
96
97      #negative wheel spin forbidden
98      if u_w_lf < 0.0:
99          u_w_lf *= 0
100
101     if u_w_rf < 0.0:
102         u_w_rf *= 0
103
104     if u_w_lr < 0.0:
105         u_w_lr *= 0
106
107     if u_w_rr < 0.0:
108         u_w_rr *= 0
109     #compute longitudinal slip
110     #switch to kinematic model for small velocities
111     if abs(x[3]) < 0.1:
112         s_lf = 0.
113         s_rf = 0.
114         s_lr = 0.
115         s_rr = 0.
116     else:
117         s_lf = 1 - p.R_w*x[23]/u_w_lf
118         s_rf = 1 - p.R_w*x[24]/u_w_rf
119         s_lr = 1 - p.R_w*x[25]/u_w_lr
120         s_rr = 1 - p.R_w*x[26]/u_w_rr
121
122     #lateral slip angles
123     #switch to kinematic model for small velocities
124     if abs(x[3]) < 0.1:
125         alpha_LF = 0.
126         alpha_RF = 0.
```

```
        alpha_LR = 0.
        alpha_RR = 0.
    else:
        alpha_LF = math.atan((x[10] + p.a*x[5] - x[14]*(p.R_w
            ↪ - x[16]))/(x[3] + 0.5*p.T_f*x[5])) - x[2]
        alpha_RF = math.atan((x[10] + p.a*x[5] - x[14]*(p.R_w
            ↪ - x[16]))/(x[3] - 0.5*p.T_f*x[5])) - x[2]
        alpha_LR = math.atan((x[10] - p.b*x[5] - x[19]*(p.R_w
            ↪ - x[21]))/(x[3] + 0.5*p.T_r*x[5]))
        alpha_RR = math.atan((x[10] - p.b*x[5] - x[19]*(p.R_w
            ↪ - x[21]))/(x[3] - 0.5*p.T_r*x[5]))

    #auxiliary suspension movement
    z_SLF = (p.h_s - p.R_w + x[16] - x[11])/math.cos(x[6]) - p
        ↪ .h_s + p.R_w + p.a*x[8] + 0.5*(x[6] - x[13])*p.T_f
    z_SRF = (p.h_s - p.R_w + x[16] - x[11])/math.cos(x[6]) - p
        ↪ .h_s + p.R_w + p.a*x[8] - 0.5*(x[6] - x[13])*p.T_f
    z_SLR = (p.h_s - p.R_w + x[21] - x[11])/math.cos(x[6]) - p
        ↪ .h_s + p.R_w - p.b*x[8] + 0.5*(x[6] - x[18])*p.T_r
    z_SRR = (p.h_s - p.R_w + x[21] - x[11])/math.cos(x[6]) - p
        ↪ .h_s + p.R_w - p.b*x[8] - 0.5*(x[6] - x[18])*p.T_r

    dz_SLF = x[17] - x[12] + p.a*x[9] + 0.5*(x[7] - x[14])*p.
        ↪ T_f
    dz_SRF = x[17] - x[12] + p.a*x[9] - 0.5*(x[7] - x[14])*p.
        ↪ T_f
    dz_SLR = x[22] - x[12] - p.b*x[9] + 0.5*(x[7] - x[19])*p.
        ↪ T_r
    dz_SRR = x[22] - x[12] - p.b*x[9] - 0.5*(x[7] - x[19])*p.
        ↪ T_r

    #camber angles
    gamma_LF = x[6] + p.D_f*z_SLF + p.E_f*(z_SLF)**2
    gamma_RF = x[6] - p.D_f*z_SRF - p.E_f*(z_SRF)**2
    gamma_LR = x[6] + p.D_r*z_SLR + p.E_r*(z_SLR)**2
    gamma_RR = x[6] - p.D_r*z_SRR - p.E_r*(z_SRR)**2

    #compute longitudinal tire forces using the magic formula
        ↪ for pure slip
    F0_x_LF = tireModel.formula_longitudinal(s_lf, gamma_LF,
        ↪ F_z_LF, p.tire)
    F0_x_RF = tireModel.formula_longitudinal(s_rf, gamma_RF,
        ↪ F_z_RF, p.tire)
    F0_x_LR = tireModel.formula_longitudinal(s_lr, gamma_LR,
        ↪ F_z_LR, p.tire)
    F0_x_RR = tireModel.formula_longitudinal(s_rr, gamma_RR,
        ↪ F_z_RR, p.tire)

```

```
158     #compute lateral tire forces using the magic formula for
            ↪ pure slip
159     res = tireModel.formula_lateral(alpha_LF, gamma_LF, F_z_LF
            ↪ , p.tire)
160     F0_y_LF = res[0]
161     mu_y_LF = res[1]
162     res = tireModel.formula_lateral(alpha_RF, gamma_RF, F_z_RF
            ↪ , p.tire)
163     F0_y_RF = res[0]
164     mu_y_RF = res[1]
165     res = tireModel.formula_lateral(alpha_LR, gamma_LR, F_z_LR
            ↪ , p.tire)
166     F0_y_LR = res[0]
167     mu_y_LR = res[1]
168     res = tireModel.formula_lateral(alpha_RR, gamma_RR, F_z_RR
            ↪ , p.tire)
169     F0_y_RR = res[0]
170     mu_y_RR = res[1]
171
172     #compute longitudinal tire forces using the magic formula
            ↪ for combined slip
173     F_x_LF = tireModel.formula_longitudinal_comb(s_lf,
            ↪ alpha_LF, F0_x_LF, p.tire)
174     F_x_RF = tireModel.formula_longitudinal_comb(s_rf,
            ↪ alpha_RF, F0_x_RF, p.tire)
175     F_x_LR = tireModel.formula_longitudinal_comb(s_lr,
            ↪ alpha_LR, F0_x_LR, p.tire)
176     F_x_RR = tireModel.formula_longitudinal_comb(s_rr,
            ↪ alpha_RR, F0_x_RR, p.tire)
177
178     #compute lateral tire forces using the magic formula for
            ↪ combined slip
179     F_y_LF = tireModel.formula_lateral_comb(s_lf, alpha_LF,
            ↪ gamma_LF, mu_y_LF, F_z_LF, F0_y_LF, p.tire)
180     F_y_RF = tireModel.formula_lateral_comb(s_rf, alpha_RF,
            ↪ gamma_RF, mu_y_RF, F_z_RF, F0_y_RF, p.tire)
181     F_y_LR = tireModel.formula_lateral_comb(s_lr, alpha_LR,
            ↪ gamma_LR, mu_y_LR, F_z_LR, F0_y_LR, p.tire)
182     F_y_RR = tireModel.formula_lateral_comb(s_rr, alpha_RR,
            ↪ gamma_RR, mu_y_RR, F_z_RR, F0_y_RR, p.tire)
183
184     #auxiliary movements for compliant joint equations
185     delta_z_f = p.h_s - p.R_w + x[16] - x[11]
186     delta_z_r = p.h_s - p.R_w + x[21] - x[11]
187
188     delta_phi_f = x[6] - x[13]
189     delta_phi_r = x[6] - x[18]
190
```

```
    dot_delta_phi_f = x[7] - x[14]
    dot_delta_phi_r = x[7] - x[19]

    dot_delta_z_f = x[17] - x[12]
    dot_delta_z_r = x[22] - x[12]

    dot_delta_y_f = x[10] + p.a*x[5] - x[15]
    dot_delta_y_r = x[10] - p.b*x[5] - x[20]

    delta_f = delta_z_f*math.sin(x[6]) - x[27]*math.cos(x[6])
       ↪ - (p.h_raf - p.R_w)*math.sin(delta_phi_f)
    delta_r = delta_z_r*math.sin(x[6]) - x[28]*math.cos(x[6])
       ↪ - (p.h_rar - p.R_w)*math.sin(delta_phi_r)

    dot_delta_f = (delta_z_f*math.cos(x[6]) + x[27]*math.sin(x
       ↪ [6]))*x[7] + dot_delta_z_f*math.sin(x[6]) -
       ↪ dot_delta_y_f*math.cos(x[6]) - (p.h_raf - p.R_w)*math
       ↪ .cos(delta_phi_f)*dot_delta_phi_f
    dot_delta_r = (delta_z_r*math.cos(x[6]) + x[28]*math.sin(x
       ↪ [6]))*x[7] + dot_delta_z_r*math.sin(x[6]) -
       ↪ dot_delta_y_r*math.cos(x[6]) - (p.h_rar - p.R_w)*math
       ↪ .cos(delta_phi_r)*dot_delta_phi_r

    #compliant joint forces
    F_RAF = delta_f*p.K_ras + dot_delta_f*p.K_rad
    F_RAR = delta_r*p.K_ras + dot_delta_r*p.K_rad

    #auxiliary suspension forces (bump stop neglected  squat/
       ↪ lift forces neglected)
    F_SLF = p.m_s*g*p.b/(2*(p.a+p.b)) - z_SLF*p.K_sf - dz_SLF*
       ↪ p.K_sdf + (x[6] - x[13])*p.K_tsf/p.T_f

    F_SRF = p.m_s*g*p.b/(2*(p.a+p.b)) - z_SRF*p.K_sf - dz_SRF*
       ↪ p.K_sdf - (x[6] - x[13])*p.K_tsf/p.T_f

    F_SLR = p.m_s*g*p.a/(2*(p.a+p.b)) - z_SLR*p.K_sr - dz_SLR*
       ↪ p.K_sdr + (x[6] - x[18])*p.K_tsr/p.T_r

    F_SRR = p.m_s*g*p.a/(2*(p.a+p.b)) - z_SRR*p.K_sr - dz_SRR*
       ↪ p.K_sdr - (x[6] - x[18])*p.K_tsr/p.T_r


    #auxiliary variables sprung mass
    sumX = F_x_LR + F_x_RR + (F_x_LF + F_x_RF)*math.cos(x[2])
       ↪ - (F_y_LF + F_y_RF)*math.sin(x[2])

    sumN = (F_y_LF + F_y_RF)*p.a*math.cos(x[2]) + (F_x_LF +
       ↪ F_x_RF)*p.a*math.sin(x[2]) \
```

```
            + (F_y_RF - F_y_LF)*0.5*p.T_f*math.sin(x[2]) + (
               ↪ F_x_LF - F_x_RF)*0.5*p.T_f*math.cos(x[2]) \
            + (F_x_LR - F_x_RR)*0.5*p.T_r - (F_y_LR + F_y_RR)*p
               ↪ .b

    sumY_s = (F_RAF + F_RAR)*math.cos(x[6]) + (F_SLF + F_SLR +
       ↪  F_SRF + F_SRR)*math.sin(x[6])

    sumL = 0.5*F_SLF*p.T_f + 0.5*F_SLR*p.T_r - 0.5*F_SRF*p.T_f
       ↪  - 0.5*F_SRR*p.T_r \
            - F_RAF/math.cos(x[6])*(p.h_s - x[11] - p.R_w + x
               ↪ [16] - (p.h_raf - p.R_w)*math.cos(x[13])) \
            - F_RAR/math.cos(x[6])*(p.h_s - x[11] - p.R_w + x
               ↪ [21] - (p.h_rar - p.R_w)*math.cos(x[18]))

    sumZ_s = (F_SLF + F_SLR + F_SRF + F_SRR)*math.cos(x[6]) -
       ↪ (F_RAF + F_RAR)*math.sin(x[6])

    sumM_s = p.a*(F_SLF + F_SRF) - p.b*(F_SLR + F_SRR) + ((
       ↪ F_x_LF + F_x_RF)*math.cos(x[2]) \
            - (F_y_LF + F_y_RF)*math.sin(x[2]) + F_x_LR +
               ↪ F_x_RR)*(p.h_s - x[11])

    #auxiliary variables unsprung mass
    sumL_uf = 0.5*F_SRF*p.T_f - 0.5*F_SLF*p.T_f - F_RAF*(p.
       ↪ h_raf - p.R_w) \
              + F_z_LF*(p.R_w*math.sin(x[13]) + 0.5*p.T_f*math
                 ↪ .cos(x[13]) - p.K_lt*F_y_LF) \
              - F_z_RF*(-p.R_w*math.sin(x[13]) + 0.5*p.T_f*
                 ↪ math.cos(x[13]) + p.K_lt*F_y_RF) \
              - ((F_y_LF + F_y_RF)*math.cos(x[2]) + (F_x_LF +
                 ↪ F_x_RF)*math.sin(x[2]))*(p.R_w - x[16])

    sumL_ur = 0.5*F_SRR*p.T_r - 0.5*F_SLR*p.T_r - F_RAR*(p.
       ↪ h_rar - p.R_w) \
              + F_z_LR*(p.R_w*math.sin(x[18]) + 0.5*p.T_r*math
                 ↪ .cos(x[18]) - p.K_lt*F_y_LR) \
              - F_z_RR*(-p.R_w*math.sin(x[18]) + 0.5*p.T_r*
                 ↪ math.cos(x[18]) + p.K_lt*F_y_RR) \
              - (F_y_LR + F_y_RR)*(p.R_w - x[21])

    sumZ_uf = F_z_LF + F_z_RF + F_RAF*math.sin(x[6]) - (F_SLF
       ↪ + F_SRF)*math.cos(x[6])

    sumZ_ur = F_z_LR + F_z_RR + F_RAR*math.sin(x[6]) - (F_SLR
       ↪ + F_SRR)*math.cos(x[6])

    sumY_uf = (F_y_LF + F_y_RF)*math.cos(x[2]) + (F_x_LF +
```

```
        ↪ F_x_RF)*math.sin(x[2]) \
254             - F_RAF*math.cos(x[6]) - (F_SLF + F_SRF)*math.
                  ↪ sin(x[6])
255
256     sumY_ur = (F_y_LR + F_y_RR) \
257             - F_RAR*math.cos(x[6]) - (F_SLR + F_SRR)*math.
                  ↪ sin(x[6])
258
259
260     #dynamics common with single-track model
261     f = [] # init 'right hand side'
262     #switch to kinematic model for small velocities
263     if abs(x[3]) < 0.1:
264         #wheelbase
265         # lwb = p.a + p.b
266
267         #system dynamics
268         # x_ks = [x[0],  x[1],  x[2],  x[3],  x[4]]
269         # f_ks = vehicle_dynamics_ks(x_ks, u, p)
270         # f.extend(f_ks)
271         # f.append(u[1]*lwb*math.tan(x[2]) + x[3]/(lwb*math.
              ↪ cos(x[2])**2)*u[0])
272
273         # Use kinematic model with reference point at center
              ↪ of mass
274         # wheelbase
275         lwb = p.a + p.b
276         # system dynamics
277         x_ks = [x[0], x[1], x[2], x[3], x[4]]
278         # kinematic model
279         f_ks = vehicle_dynamics_ks_cog(x_ks, u, p)
280         f = [f_ks[0], f_ks[1], f_ks[2], f_ks[3], f_ks[4]]
281         # derivative of slip angle and yaw rate
282         d_beta = (p.b * u[0]) / (lwb * math.cos(x[2]) ** 2 *
              ↪ (1 + (math.tan(x[2]) ** 2 * p.b / lwb) ** 2))
283         dd_psi = 1 / lwb * (u[1] * math.cos(x[6]) * math.tan(x
              ↪ [2]) -
284                             x[3] * math.sin(x[6]) * d_beta *
                                  ↪ math.tan(x[2]) +
285                             x[3] * math.cos(x[6]) * u[0] /
                                  ↪ math.cos(x[2]) ** 2)
286         f.append(dd_psi)
287
288     else:
289         f.append(math.cos(beta + x[4])*vel)
290         f.append(math.sin(beta + x[4])*vel)
291         f.append(u[0])
292         f.append(1/p.m*sumX + x[5]*x[10])
```

```
293         f.append(x[5])
294         f.append(1/(p.I_z - (p.I_xz_s)**2/p.I_Phi_s)*(sumN + p
                ↪ .I_xz_s/p.I_Phi_s*sumL))
295
296
297     # remaining sprung mass dynamics
298     f.append(x[7])
299     f.append(1/(p.I_Phi_s - (p.I_xz_s)**2/p.I_z)*(p.I_xz_s/p.
            ↪ I_z*sumN + sumL))
300     f.append(x[9])
301     f.append(1/p.I_y_s*sumM_s)
302     f.append(1/p.m_s*sumY_s - x[5]*x[3])
303     f.append(x[12])
304     f.append(g - 1/p.m_s*sumZ_s)
305
306     #unsprung mass dynamics (front)
307     f.append(x[14])
308     f.append(1/p.I_uf*sumL_uf)
309     f.append(1/p.m_uf*sumY_uf - x[5]*x[3])
310     f.append(x[17])
311     f.append(g - 1/p.m_uf*sumZ_uf)
312
313     #unsprung mass dynamics (rear)
314     f.append(x[19])
315     f.append(1/p.I_ur*sumL_ur)
316     f.append(1/p.m_ur*sumY_ur - x[5]*x[3])
317     f.append(x[22])
318     f.append(g - 1/p.m_ur*sumZ_ur)
319
320     #convert acceleration input to brake and engine torque
321     if u[1]>0:
322         T_B = 0.0
323         T_E = p.m*p.R_w*u[1]
324     else:
325         T_B = p.m*p.R_w*u[1]
326         T_E = 0.
327
328
329
330     #wheel dynamics (p.T  new parameter for torque splitting)
331     f.append(1/p.I_y_w*(-p.R_w*F_x_LF + 0.5*p.T_sb*T_B + 0.5*p
            ↪ .T_se*T_E + p.T_se*u[2]))
332     f.append(1/p.I_y_w*(-p.R_w*F_x_RF + 0.5*p.T_sb*T_B + 0.5*p
            ↪ .T_se*T_E - p.T_se*u[2]))
333     f.append(1/p.I_y_w*(-p.R_w*F_x_LR + 0.5*(1-p.T_sb)*T_B +
            ↪ 0.5*(1-p.T_se)*T_E + (1-p.T_se)*u[2]))
334     f.append(1/p.I_y_w*(-p.R_w*F_x_RR + 0.5*(1-p.T_sb)*T_B +
            ↪ 0.5*(1-p.T_se)*T_E - (1-p.T_se)*u[2]))
```

```
335
336     #negative wheel spin forbidden
337     for iState in range(23, 27):
338         if x[iState] < 0.0:
339             x[iState] = 0.0
340             f[iState] = 0.0
341
342     #compliant joint equations
343     f.append(dot_delta_y_f)
344     f.append(dot_delta_y_r)
345
346     return f
```

⌞ Πατήστε εδώ για να επιστρέψετε στο κυρίως κείμενο.

Απόσπασμα Κώδικα A.9: Προσομοίωση αριστερής στροφής με $v_0 = 15\,m/s$, $v_\delta = 0.005\,rad/s$ και $a_{\text{long}} = 0.2g$ και κατανομή ροπής $\Delta T = 100\,Nm$

```

x_TV_left = x_acc.copy()
x_TV_right = x_acc.copy()
p = parameters_vehicle2()

for i in range(len(time)-1):

  ΔT_left = -100
  ΔT_right = 100

  # Apply torque vectoring by adding the corrective torque to each wheel
  u_left = (v_delta, 0.2 * g, ΔT_left)
  u_right = (v_delta, 0.2 * g, ΔT_right)

  # Simulate the multi-body kinematic model with the updated control input
  x_TV_left[i + 1] = odeint(func_MB, x_TV_left[i], [0, dt], args=(u_left, p))[1]
  x_TV_right[i + 1] = odeint(func_MB, x_TV_right[i], [0, dt], args=(u_right, p))[1]

# position
plt.title('Σύγκριση θέσης')
plt.plot([tmp[0] for tmp in x_acc], [tmp[1] for tmp in x_acc])
plt.plot([tmp[0] for tmp in x_TV_right], [tmp[1] for tmp in x_TV_right])
plt.plot([tmp[0] for tmp in x_TV_left], [tmp[1] for tmp in x_TV_left])
plt.legend(['επιτάχυνση χωρίς TV', 'επιτάχυνση με πρόσθετη ροπ ή στη δεξιά πλευρά', 'επιτάχυνση με πρόσθετη ροπή στην αρ ιστερή πλευρά'])
plt.xlabel('x-θέση (m)')
plt.ylabel('y-θέση (m)')
plt.show()

# orientation
plt.title('Σύγκριση γωνίας εκτροπής')
plt.plot(time, [tmp[4] for tmp in x_acc])
plt.plot(time, [tmp[4] for tmp in x_TV_right])
plt.plot(time, [tmp[4] for tmp in x_TV_left])
plt.legend(['επιτάχυνση χωρίς TV', 'επιτάχυνση με πρόσθετη ροπ ή στη δεξιά πλευρά', 'επιτάχυνση με πρόσθετη ροπή στην αρ ιστερή πλευρά'])
```

```
35 plt.xlabel('Χρόνος (s)')
36 plt.ylabel('Γωνία εκτροπής (rad)')
37 plt.show()
38
39 # yaw rate
40 plt.title('Σύγκριση ρυθμού εκτροπής')
41 plt.plot(time, [tmp[5] for tmp in x_acc])
42 plt.plot(time, [tmp[5] for tmp in x_TV_right])
43 plt.plot(time, [tmp[5] for tmp in x_TV_left])
44 plt.legend(['επιτάχυνση χωρίς TV', 'επιτάχυνση με πρόσθετη ροπ
   ↪ ή στη δεξιά πλευρά', 'επιτάχυνση με πρόσθετη ροπή στην αρ
   ↪ ιστερή πλευρά'])
45 plt.xlabel('Χρόνος (s)')
46 plt.ylabel('Ρυθμός εκτροπής (rad/s)')
47 plt.show()
```

⌞ Πατήστε εδώ για να επιστρέψετε στο κυρίως κείμενο.

Απόσπασμα Κώδικα A.10: Ενημέρωση του αρχείου δυναμικής του οχήματος (vehicle_dynamics_mb.py) για την υποστήριξη left-right και front-rear TV

```
from vehiclemodels.utils.steering_constraints import
    ↪ steering_constraints
from vehiclemodels.utils.acceleration_constraints import
    ↪ acceleration_constraints
from vehiclemodels.utils.vehicle_dynamics_ks_cog import
    ↪ vehicle_dynamics_ks_cog
import vehiclemodels.utils.tire_model as tireModel

def vehicle_dynamics_mb(x, uInit, p):
    """
    vehicleDynamics_mb - multi-body vehicle dynamics based on
        ↪ the DOT (department of transportation) vehicle
        ↪ dynamics
    reference point: center of mass

    Syntax:
        f = vehicleDynamics_mb(x,u,p)

    Inputs:
        :param x: vehicle state vector
        :param uInit: vehicle input vector
        :param p: vehicle parameter vector

    Outputs:
        :return f: right-hand side of differential equations
    """
    # set gravity constant
    g = 9.81  #[m/s^2]

    #states
    #x1 = x-position in a global coordinate system
    #x2 = y-position in a global coordinate system
    #x3 = steering angle of front wheels
    #x4 = velocity in x-direction
    #x5 = yaw angle
    #x6 = yaw rate

    #x7 = roll angle
    #x8 = roll rate
    #x9 = pitch angle
    #x10 = pitch rate
    #x11 = velocity in y-direction
    #x12 = z-position
    #x13 = velocity in z-direction
```

```
40
41      #x14 = roll angle front
42      #x15 = roll rate front
43      #x16 = velocity in y-direction front
44      #x17 = z-position front
45      #x18 = velocity in z-direction front
46
47      #x19 = roll angle rear
48      #x20 = roll rate rear
49      #x21 = velocity in y-direction rear
50      #x22 = z-position rear
51      #x23 = velocity in z-direction rear
52
53      #x24 = left front wheel angular speed
54      #x25 = right front wheel angular speed
55      #x26 = left rear wheel angular speed
56      #x27 = right rear wheel angular speed
57
58      #x28 = delta_y_f
59      #x29 = delta_y_r
60
61      #u1 = steering angle velocity of front wheels
62      #u2 = acceleration
63      #u3 = left-right corrective torque
64      #u4 = front-rear engine torque distribution
65
66      #consider steering constraints
67      u = []
68      u.append(steering_constraints(x[2], uInit[0], p.steering))
           ↪  # different name u_init/u due to side effects of u
69      #consider acceleration constraints
70      u.append(acceleration_constraints(x[3], uInit[1], p.
           ↪ longitudinal)) # different name u_init/u due to side
           ↪ effects of u
71      u.append(uInit[2])
72      u.append(uInit[3])
73
74      #compute slip angle at cg
75      #switch to kinematic model for small velocities
76      if abs(x[3]) < 0.1:
77          beta = 0.
78      else:
79          beta = math.atan(x[10]/x[3])
80      vel = math.sqrt(x[3]**2 + x[10]**2)
81
82
83
84      #vertical tire forces
```

```
    F_z_LF = (x[16] + p.R_w*(math.cos(x[13]) - 1) - 0.5*p.T_f*
       ↪ math.sin(x[13]))*p.K_zt
    F_z_RF = (x[16] + p.R_w*(math.cos(x[13]) - 1) + 0.5*p.T_f*
       ↪ math.sin(x[13]))*p.K_zt
    F_z_LR = (x[21] + p.R_w*(math.cos(x[18]) - 1) - 0.5*p.T_r*
       ↪ math.sin(x[18]))*p.K_zt
    F_z_RR = (x[21] + p.R_w*(math.cos(x[18]) - 1) + 0.5*p.T_r*
       ↪ math.sin(x[18]))*p.K_zt

    #obtain individual tire speeds
    u_w_lf = (x[3] + 0.5*p.T_f*x[5])*math.cos(x[2]) + (x[10] +
       ↪  p.a*x[5])*math.sin(x[2])
    u_w_rf = (x[3] - 0.5*p.T_f*x[5])*math.cos(x[2]) + (x[10] +
       ↪  p.a*x[5])*math.sin(x[2])
    u_w_lr = x[3] + 0.5*p.T_r*x[5]
    u_w_rr = x[3] - 0.5*p.T_r*x[5]

    #negative wheel spin forbidden
    if u_w_lf < 0.0:
       u_w_lf *= 0

    if u_w_rf < 0.0:
       u_w_rf *= 0

    if u_w_lr < 0.0:
       u_w_lr *= 0

    if u_w_rr < 0.0:
       u_w_rr *= 0
    #compute longitudinal slip
    #switch to kinematic model for small velocities
    if abs(x[3]) < 0.1:
        s_lf = 0.
        s_rf = 0.
        s_lr = 0.
        s_rr = 0.
    else:
        s_lf = 1 - p.R_w*x[23]/u_w_lf
        s_rf = 1 - p.R_w*x[24]/u_w_rf
        s_lr = 1 - p.R_w*x[25]/u_w_lr
        s_rr = 1 - p.R_w*x[26]/u_w_rr

    #lateral slip angles
    #switch to kinematic model for small velocities
    if abs(x[3]) < 0.1:
        alpha_LF = 0.
        alpha_RF = 0.
        alpha_LR = 0.
```

```
127         alpha_RR = 0.
128     else:
129         alpha_LF = math.atan((x[10] + p.a*x[5] - x[14]*(p.R_w
                ↪ - x[16]))/(x[3] + 0.5*p.T_f*x[5])) - x[2]
130         alpha_RF = math.atan((x[10] + p.a*x[5] - x[14]*(p.R_w
                ↪ - x[16]))/(x[3] - 0.5*p.T_f*x[5])) - x[2]
131         alpha_LR = math.atan((x[10] - p.b*x[5] - x[19]*(p.R_w
                ↪ - x[21]))/(x[3] + 0.5*p.T_r*x[5]))
132         alpha_RR = math.atan((x[10] - p.b*x[5] - x[19]*(p.R_w
                ↪ - x[21]))/(x[3] - 0.5*p.T_r*x[5]))
133
134     #auxiliary suspension movement
135     z_SLF = (p.h_s - p.R_w + x[16] - x[11])/math.cos(x[6]) - p
            ↪ .h_s + p.R_w + p.a*x[8] + 0.5*(x[6] - x[13])*p.T_f
136     z_SRF = (p.h_s - p.R_w + x[16] - x[11])/math.cos(x[6]) - p
            ↪ .h_s + p.R_w + p.a*x[8] - 0.5*(x[6] - x[13])*p.T_f
137     z_SLR = (p.h_s - p.R_w + x[21] - x[11])/math.cos(x[6]) - p
            ↪ .h_s + p.R_w - p.b*x[8] + 0.5*(x[6] - x[18])*p.T_r
138     z_SRR = (p.h_s - p.R_w + x[21] - x[11])/math.cos(x[6]) - p
            ↪ .h_s + p.R_w - p.b*x[8] - 0.5*(x[6] - x[18])*p.T_r
139
140     dz_SLF = x[17] - x[12] + p.a*x[9] + 0.5*(x[7] - x[14])*p.
            ↪ T_f
141     dz_SRF = x[17] - x[12] + p.a*x[9] - 0.5*(x[7] - x[14])*p.
            ↪ T_f
142     dz_SLR = x[22] - x[12] - p.b*x[9] + 0.5*(x[7] - x[19])*p.
            ↪ T_r
143     dz_SRR = x[22] - x[12] - p.b*x[9] - 0.5*(x[7] - x[19])*p.
            ↪ T_r
144
145     #camber angles
146     gamma_LF = x[6] + p.D_f*z_SLF + p.E_f*(z_SLF)**2
147     gamma_RF = x[6] - p.D_f*z_SRF - p.E_f*(z_SRF)**2
148     gamma_LR = x[6] + p.D_r*z_SLR + p.E_r*(z_SLR)**2
149     gamma_RR = x[6] - p.D_r*z_SRR - p.E_r*(z_SRR)**2
150
151     #compute longitudinal tire forces using the magic formula
            ↪ for pure slip
152     F0_x_LF = tireModel.formula_longitudinal(s_lf, gamma_LF,
            ↪ F_z_LF, p.tire)
153     F0_x_RF = tireModel.formula_longitudinal(s_rf, gamma_RF,
            ↪ F_z_RF, p.tire)
154     F0_x_LR = tireModel.formula_longitudinal(s_lr, gamma_LR,
            ↪ F_z_LR, p.tire)
155     F0_x_RR = tireModel.formula_longitudinal(s_rr, gamma_RR,
            ↪ F_z_RR, p.tire)
156
157     #compute lateral tire forces using the magic formula for
```

```
    ↪ pure slip
158     res = tireModel.formula_lateral(alpha_LF, gamma_LF, F_z_LF
    ↪ , p.tire)
159     F0_y_LF = res[0]
160     mu_y_LF = res[1]
161     res = tireModel.formula_lateral(alpha_RF, gamma_RF, F_z_RF
    ↪ , p.tire)
162     F0_y_RF = res[0]
163     mu_y_RF = res[1]
164     res = tireModel.formula_lateral(alpha_LR, gamma_LR, F_z_LR
    ↪ , p.tire)
165     F0_y_LR = res[0]
166     mu_y_LR = res[1]
167     res = tireModel.formula_lateral(alpha_RR, gamma_RR, F_z_RR
    ↪ , p.tire)
168     F0_y_RR = res[0]
169     mu_y_RR = res[1]
170
171     #compute longitudinal tire forces using the magic formula
    ↪ for combined slip
172     F_x_LF = tireModel.formula_longitudinal_comb(s_lf,
    ↪ alpha_LF, F0_x_LF, p.tire)
173     F_x_RF = tireModel.formula_longitudinal_comb(s_rf,
    ↪ alpha_RF, F0_x_RF, p.tire)
174     F_x_LR = tireModel.formula_longitudinal_comb(s_lr,
    ↪ alpha_LR, F0_x_LR, p.tire)
175     F_x_RR = tireModel.formula_longitudinal_comb(s_rr,
    ↪ alpha_RR, F0_x_RR, p.tire)
176
177     #compute lateral tire forces using the magic formula for
    ↪ combined slip
178     F_y_LF = tireModel.formula_lateral_comb(s_lf, alpha_LF,
    ↪ gamma_LF, mu_y_LF, F_z_LF, F0_y_LF, p.tire)
179     F_y_RF = tireModel.formula_lateral_comb(s_rf, alpha_RF,
    ↪ gamma_RF, mu_y_RF, F_z_RF, F0_y_RF, p.tire)
180     F_y_LR = tireModel.formula_lateral_comb(s_lr, alpha_LR,
    ↪ gamma_LR, mu_y_LR, F_z_LR, F0_y_LR, p.tire)
181     F_y_RR = tireModel.formula_lateral_comb(s_rr, alpha_RR,
    ↪ gamma_RR, mu_y_RR, F_z_RR, F0_y_RR, p.tire)
182
183     #auxiliary movements for compliant joint equations
184     delta_z_f = p.h_s - p.R_w + x[16] - x[11]
185     delta_z_r = p.h_s - p.R_w + x[21] - x[11]
186
187     delta_phi_f = x[6] - x[13]
188     delta_phi_r = x[6] - x[18]
189
190     dot_delta_phi_f = x[7] - x[14]
```

```
    dot_delta_phi_r = x[7] - x[19]

    dot_delta_z_f = x[17] - x[12]
    dot_delta_z_r = x[22] - x[12]

    dot_delta_y_f = x[10] + p.a*x[5] - x[15]
    dot_delta_y_r = x[10] - p.b*x[5] - x[20]

    delta_f = delta_z_f*math.sin(x[6]) - x[27]*math.cos(x[6])
       ↪ - (p.h_raf - p.R_w)*math.sin(delta_phi_f)
    delta_r = delta_z_r*math.sin(x[6]) - x[28]*math.cos(x[6])
       ↪ - (p.h_rar - p.R_w)*math.sin(delta_phi_r)

    dot_delta_f = (delta_z_f*math.cos(x[6]) + x[27]*math.sin(x
       ↪ [6]))*x[7] + dot_delta_z_f*math.sin(x[6]) -
       ↪ dot_delta_y_f*math.cos(x[6]) - (p.h_raf - p.R_w)*math
       ↪ .cos(delta_phi_f)*dot_delta_phi_f
    dot_delta_r = (delta_z_r*math.cos(x[6]) + x[28]*math.sin(x
       ↪ [6]))*x[7] + dot_delta_z_r*math.sin(x[6]) -
       ↪ dot_delta_y_r*math.cos(x[6]) - (p.h_rar - p.R_w)*math
       ↪ .cos(delta_phi_r)*dot_delta_phi_r

    #compliant joint forces
    F_RAF = delta_f*p.K_ras + dot_delta_f*p.K_rad
    F_RAR = delta_r*p.K_ras + dot_delta_r*p.K_rad

    #auxiliary suspension forces (bump stop neglected  squat/
       ↪ lift forces neglected)
    F_SLF = p.m_s*g*p.b/(2*(p.a+p.b)) - z_SLF*p.K_sf - dz_SLF*
       ↪ p.K_sdf + (x[6] - x[13])*p.K_tsf/p.T_f

    F_SRF = p.m_s*g*p.b/(2*(p.a+p.b)) - z_SRF*p.K_sf - dz_SRF*
       ↪ p.K_sdf - (x[6] - x[13])*p.K_tsf/p.T_f

    F_SLR = p.m_s*g*p.a/(2*(p.a+p.b)) - z_SLR*p.K_sr - dz_SLR*
       ↪ p.K_sdr + (x[6] - x[18])*p.K_tsr/p.T_r

    F_SRR = p.m_s*g*p.a/(2*(p.a+p.b)) - z_SRR*p.K_sr - dz_SRR*
       ↪ p.K_sdr - (x[6] - x[18])*p.K_tsr/p.T_r


    #auxiliary variables sprung mass
    sumX = F_x_LR + F_x_RR + (F_x_LF + F_x_RF)*math.cos(x[2])
       ↪ - (F_y_LF + F_y_RF)*math.sin(x[2])

    sumN = (F_y_LF + F_y_RF)*p.a*math.cos(x[2]) + (F_x_LF +
       ↪ F_x_RF)*p.a*math.sin(x[2]) \
            + (F_y_RF - F_y_LF)*0.5*p.T_f*math.sin(x[2]) + (
```

```
            ↪ F_x_LF - F_x_RF)*0.5*p.T_f*math.cos(x[2]) \
224         + (F_x_LR - F_x_RR)*0.5*p.T_r - (F_y_LR + F_y_RR)*p
            ↪ .b
225
226     sumY_s = (F_RAF + F_RAR)*math.cos(x[6]) + (F_SLF + F_SLR +
        ↪  F_SRF + F_SRR)*math.sin(x[6])
227
228     sumL = 0.5*F_SLF*p.T_f + 0.5*F_SLR*p.T_r - 0.5*F_SRF*p.T_f
        ↪  - 0.5*F_SRR*p.T_r \
229         - F_RAF/math.cos(x[6])*(p.h_s - x[11] - p.R_w + x
            ↪ [16] - (p.h_raf - p.R_w)*math.cos(x[13])) \
230         - F_RAR/math.cos(x[6])*(p.h_s - x[11] - p.R_w + x
            ↪ [21] - (p.h_rar - p.R_w)*math.cos(x[18]))
231
232     sumZ_s = (F_SLF + F_SLR + F_SRF + F_SRR)*math.cos(x[6]) -
        ↪ (F_RAF + F_RAR)*math.sin(x[6])
233
234     sumM_s = p.a*(F_SLF + F_SRF) - p.b*(F_SLR + F_SRR) + ((
        ↪ F_x_LF + F_x_RF)*math.cos(x[2]) \
235         - (F_y_LF + F_y_RF)*math.sin(x[2]) + F_x_LR +
            ↪ F_x_RR)*(p.h_s - x[11])
236
237     #auxiliary variables unsprung mass
238     sumL_uf = 0.5*F_SRF*p.T_f - 0.5*F_SLF*p.T_f - F_RAF*(p.
        ↪ h_raf - p.R_w) \
239             + F_z_LF*(p.R_w*math.sin(x[13]) + 0.5*p.T_f*math
                ↪ .cos(x[13]) - p.K_lt*F_y_LF) \
240             - F_z_RF*(-p.R_w*math.sin(x[13]) + 0.5*p.T_f*
                ↪ math.cos(x[13]) + p.K_lt*F_y_RF) \
241             - ((F_y_LF + F_y_RF)*math.cos(x[2]) + (F_x_LF +
                ↪ F_x_RF)*math.sin(x[2]))*(p.R_w - x[16])
242
243     sumL_ur = 0.5*F_SRR*p.T_r - 0.5*F_SLR*p.T_r - F_RAR*(p.
        ↪ h_rar - p.R_w) \
244             + F_z_LR*(p.R_w*math.sin(x[18]) + 0.5*p.T_r*math
                ↪ .cos(x[18]) - p.K_lt*F_y_LR) \
245             - F_z_RR*(-p.R_w*math.sin(x[18]) + 0.5*p.T_r*
                ↪ math.cos(x[18]) + p.K_lt*F_y_RR) \
246             - (F_y_LR + F_y_RR)*(p.R_w - x[21])
247
248     sumZ_uf = F_z_LF + F_z_RF + F_RAF*math.sin(x[6]) - (F_SLF
        ↪ + F_SRF)*math.cos(x[6])
249
250     sumZ_ur = F_z_LR + F_z_RR + F_RAR*math.sin(x[6]) - (F_SLR
        ↪ + F_SRR)*math.cos(x[6])
251
252     sumY_uf = (F_y_LF + F_y_RF)*math.cos(x[2]) + (F_x_LF +
        ↪ F_x_RF)*math.sin(x[2]) \
```

```
253                 - F_RAF*math.cos(x[6]) - (F_SLF + F_SRF)*math.
                        ↪ sin(x[6])
254
255     sumY_ur = (F_y_LR + F_y_RR) \
256                 - F_RAR*math.cos(x[6]) - (F_SLR + F_SRR)*math.
                        ↪ sin(x[6])
257
258
259     #dynamics common with single-track model
260     f = [] # init 'right hand side'
261     #switch to kinematic model for small velocities
262     if abs(x[3]) < 0.1:
263         #wheelbase
264         # lwb = p.a + p.b
265
266         #system dynamics
267         # x_ks = [x[0],  x[1],  x[2],  x[3],  x[4]]
268         # f_ks = vehicle_dynamics_ks(x_ks, u, p)
269         # f.extend(f_ks)
270         # f.append(u[1]*lwb*math.tan(x[2]) + x[3]/(lwb*math.
                ↪ cos(x[2])**2)*u[0])
271
272         # Use kinematic model with reference point at center
                ↪ of mass
273         # wheelbase
274         lwb = p.a + p.b
275         # system dynamics
276         x_ks = [x[0], x[1], x[2], x[3], x[4]]
277         # kinematic model
278         f_ks = vehicle_dynamics_ks_cog(x_ks, u, p)
279         f = [f_ks[0], f_ks[1], f_ks[2], f_ks[3], f_ks[4]]
280         # derivative of slip angle and yaw rate
281         d_beta = (p.b * u[0]) / (lwb * math.cos(x[2]) ** 2 *
                ↪ (1 + (math.tan(x[2]) ** 2 * p.b / lwb) ** 2))
282         dd_psi = 1 / lwb * (u[1] * math.cos(x[6]) * math.tan(x
                ↪ [2]) -
283                             x[3] * math.sin(x[6]) * d_beta *
                                ↪ math.tan(x[2]) +
284                             x[3] * math.cos(x[6]) * u[0] /
                                ↪ math.cos(x[2]) ** 2)
285         f.append(dd_psi)
286
287     else:
288         f.append(math.cos(beta + x[4])*vel)
289         f.append(math.sin(beta + x[4])*vel)
290         f.append(u[0])
291         f.append(1/p.m*sumX + x[5]*x[10])
292         f.append(x[5])
```

```
293         f.append(1/(p.I_z - (p.I_xz_s)**2/p.I_Phi_s)*(sumN + p
                ↪ .I_xz_s/p.I_Phi_s*sumL))
294
295
296     # remaining sprung mass dynamics
297     f.append(x[7])
298     f.append(1/(p.I_Phi_s - (p.I_xz_s)**2/p.I_z)*(p.I_xz_s/p.
           ↪ I_z*sumN + sumL))
299     f.append(x[9])
300     f.append(1/p.I_y_s*sumM_s)
301     f.append(1/p.m_s*sumY_s - x[5]*x[3])
302     f.append(x[12])
303     f.append(g - 1/p.m_s*sumZ_s)
304
305     #unsprung mass dynamics (front)
306     f.append(x[14])
307     f.append(1/p.I_uf*sumL_uf)
308     f.append(1/p.m_uf*sumY_uf - x[5]*x[3])
309     f.append(x[17])
310     f.append(g - 1/p.m_uf*sumZ_uf)
311
312     #unsprung mass dynamics (rear)
313     f.append(x[19])
314     f.append(1/p.I_ur*sumL_ur)
315     f.append(1/p.m_ur*sumY_ur - x[5]*x[3])
316     f.append(x[22])
317     f.append(g - 1/p.m_ur*sumZ_ur)
318
319     #convert acceleration input to brake and engine torque
320     if u[1]>0:
321         T_B = 0.0
322         T_E = p.m*p.R_w*u[1]
323     else:
324         T_B = p.m*p.R_w*u[1]
325         T_E = 0.
326
327
328
329     #wheel dynamics (p.T  new parameter for torque splitting)
330
331     f.append(1/p.I_y_w*(-p.R_w*F_x_LF + 0.5*p.T_sb*T_B + 0.5*u
           ↪ [3]*T_E + u[3]*u[2]))
332     f.append(1/p.I_y_w*(-p.R_w*F_x_RF + 0.5*p.T_sb*T_B + 0.5*u
           ↪ [3]*T_E - u[3]*u[2]))
333     f.append(1/p.I_y_w*(-p.R_w*F_x_LR + 0.5*(1-p.T_sb)*T_B +
           ↪ 0.5*(1-u[3])*T_E + (1-u[3])*u[2]))
334     f.append(1/p.I_y_w*(-p.R_w*F_x_RR + 0.5*(1-p.T_sb)*T_B +
           ↪ 0.5*(1-u[3])*T_E - (1-u[3])*u[2]))
```

```
335
336     #negative wheel spin forbidden
337     for iState in range(23, 27):
338         if x[iState] < 0.0:
339             x[iState] = 0.0
340             f[iState] = 0.0
341
342     #compliant joint equations
343     f.append(dot_delta_y_f)
344     f.append(dot_delta_y_r)
345
346     return f
```

⌞ Πατήστε εδώ για να επιστρέψετε στο κυρίως κείμενο.

Απόσπασμα Κώδικα A.11: Προσομοίωση αριστερής στροφής με $v_0 = 15\,m/s$, $v_\delta = 0.005\,rad/s$ και $a_{\text{long}} = 0.2g$, κατανομή ροπής $\Delta T = 100\,Nm$ και $u_4 = 0.2/0.8$

```
x_TV_left_fr = x_acc.copy()
x_TV_right_rr = x_acc.copy()
p = parameters_vehicle2()

for i in range(len(time)-1):

  ΔT_left = -100
  ΔT_right = 100

  # Apply torque vectoring by adding the corrective torque to each wheel
  u_left_fr = (v_delta, 0.2 * g, ΔT_left, 0.8)
  u_right_rr = (v_delta, 0.2 * g, ΔT_right, 0.2)

  # Simulate the multi-body kinematic model with the updated control input
  x_TV_left_fr[i + 1] = odeint(func_MB, x_TV_left_fr[i], [0, dt], args=(u_left_fr, p))[1]
  x_TV_right_rr[i + 1] = odeint(func_MB, x_TV_right_rr[i], [0,  dt], args=(u_right_rr, p))[1]

# position
plt.figure(figsize=(8, 6))
plt.title('Σύγκριση θέσης')
plt.plot([tmp[0] for tmp in x_acc], [tmp[1] for tmp in x_acc])
plt.plot([tmp[0] for tmp in x_TV_right], [tmp[1] for tmp in x_TV_right])
plt.plot([tmp[0] for tmp in x_TV_right_rr], [tmp[1] for tmp in  x_TV_right_rr])
plt.plot([tmp[0] for tmp in x_TV_left], [tmp[1] for tmp in x_TV_left])
plt.plot([tmp[0] for tmp in x_TV_left_fr], [tmp[1] for tmp in x_TV_left_fr])
plt.legend(['χωρίς TV', 'με πρόσθετη ροπή στη δεξιά πλευρά', 'με πρόσθετη ροπή στη δεξιά πλευρά + 70% της ροπής \n πίσω ', 'με πρόσθετη ροπή στην αριστερή πλευρά', 'με πρόσθετη ροπή στην αριστερή πλευρά + 70% της \n ροπής εμπρός'], fontsize = 10)
plt.xlabel('x-θέση (m)')
plt.ylabel('y-θέση (m)')
plt.show()

# orientation
plt.figure(figsize=(8, 6))
```

```
plt.title('Σύγκριση γωνίας εκτροπής')
plt.plot(time, [tmp[4] for tmp in x_acc])
plt.plot(time, [tmp[4] for tmp in x_TV_right])
plt.plot(time, [tmp[4] for tmp in x_TV_right_rr])
plt.plot(time, [tmp[4] for tmp in x_TV_left])
plt.plot(time, [tmp[4] for tmp in x_TV_left_fr])
plt.legend(['χωρίς TV', 'με πρόσθετη ροπή στη δεξιά πλευρά', '
    ↪ με πρόσθετη ροπή στη δεξιά πλευρά + 70% της ροπής \n πίσω
    ↪ ', 'με πρόσθετη ροπή στην αριστερή πλευρά', 'με πρόσθετη
    ↪ ροπή στην αριστερή πλευρά + 70% της \n ροπής εμπρός'],
    ↪ fontsize = 10)
plt.xlabel('Χρόνος (s)')
plt.ylabel('Γωνία εκτροπής (rad)')
plt.show()

# yaw rate
plt.figure(figsize=(8, 6))
plt.title('Σύγκριση ρυθμού εκτροπής')
plt.plot(time, [tmp[5] for tmp in x_acc])
plt.plot(time, [tmp[5] for tmp in x_TV_right])
plt.plot(time, [tmp[5] for tmp in x_TV_right_rr])
plt.plot(time, [tmp[5] for tmp in x_TV_left])
plt.plot(time, [tmp[5] for tmp in x_TV_left_fr])
plt.legend(['χωρίς TV', 'με πρόσθετη ροπή στη δεξιά πλευρά', '
    ↪ με πρόσθετη ροπή στη δεξιά πλευρά + 70% της ροπής \n πίσω
    ↪ ', 'με πρόσθετη ροπή στην αριστερή πλευρά', 'με πρόσθετη
    ↪ ροπή στην αριστερή πλευρά + 70% της \n ροπής εμπρός'],
    ↪ fontsize = 10)
plt.xlabel('Χρόνος (s)')
plt.ylabel('Ρυθμός εκτροπής (rad/s)')
plt.show()
```

⌞ Πατήστε εδώ για να επιστρέψετε στο κυρίως κείμενο.

Απόσπασμα Κώδικα A.12: Συνάρτηση εκτίμησης επιθυμητού ρυθμού εκτροπής

```
def desired_values(x, u):

  #Vehicle Parameters
  lf = p.a
  lr = p.b
  h = p.h_s
  m = p.m
  mu = p.tire.p_dy1
  C_S = -p.tire.p_ky1 / p.tire.p_dy1
  F_zf = (m*g*lr-m*u[1]*h)/(lr+lf)
  F_zr = (m*g*lf+m*u[1]*h)/(lr+lf)
  C_af = mu*C_S*F_zf
  C_ar = mu*C_S*F_zr

  psi_dot_des = [] #Desired yaw rate

  nom_psi_dot = x[2]*x[3]
  denom = lf+lr+(m*x[3]**2*(lr*C_ar-lf*C_af))/(2*C_af*C_ar*(lf
    ↪ +lr))
  temp_psi_dot = nom_psi_dot/denom
  if np.abs(temp_psi_dot) <= 0.85*mu*g/x[3]:          # Upper
    ↪ bounded value for target Yaw Rate(Eq. 8.28) from
    ↪ Rajamani Vehicle  Dynamics & Control
    psi_dot_des.append(temp_psi_dot)
  else:
    psi_dot_des.append((0.85*mu*g/x[3])*np.sign(temp_psi_dot))

  return psi_dot_des
```

⌞ Πατήστε εδώ για να επιστρέψετε στο κυρίως κείμενο.

Απόσπασμα Κώδικα A.13: Συνάρτηση ελέγχου της παραμέτρου $T_{s,e}$

```
from scipy.interpolate import interp1d

def tune_T_se(error):

  x = [-1.5, 1.5]
  y = [0.1,0.9]
  t = interp1d(x, y, bounds_error=False, fill_value=(0.1,0.9))
  t_eng = t(error)

  return t_eng.item()
```

⌞ Πατήστε εδώ για να επιστρέψετε στο κυρίως κείμενο.

Απόσπασμα Κώδικα Α.14: Επιταχυνόμενη αριστερή στροφή με ρυθμό αύξησης γωνίας διεύθυνσης $v_\delta = 0.05\,rad/s$, επιτάχυνση $a_{\text{long}} = 0.2g$ και αριστερό-δεξί TV

```
from vehiclemodels.utils.steering_constraints import
    ↪ steering_constraints
from vehiclemodels.utils.acceleration_constraints import
    ↪ acceleration_constraints

# Initialize arrays, variables & lists for the control loop
x_lr_TV_4 = x_acc_4.copy()
x_lr_TV_8 = x_acc_8.copy()
x_lr_TV_15 = x_acc_15.copy()
e_psidot_prev_4 = 0
e_psidot_prev_8 = 0
e_psidot_prev_15 = 0
yaw_rate_desired_4 = []
yaw_rate_desired_8 = []
yaw_rate_desired_15 = []

kp = 2
kd = 400

for i in range(len(time)-1):

  acc_4 = acceleration_constraints(x_lr_TV_4[i,3],  0.2 * g, p
    ↪ .longitudinal)
  acc_8 = acceleration_constraints(x_lr_TV_8[i,3],  0.2 * g, p
    ↪ .longitudinal)
  acc_15 = acceleration_constraints(x_lr_TV_15[i,3],  0.2 * g,
    ↪  p.longitudinal)

  T_E_4 =  p.m * p.R_w * acc_4
  T_E_8 =  p.m * p.R_w * acc_8
  T_E_15 =  p.m * p.R_w * acc_15

  # Calculate errors between desired and actual yaw rate
  e_psidot_4 = desired_values(x_lr_TV_4[i], u) - x_lr_TV_4[i
    ↪ ,5]
  e_psidot_8 = desired_values(x_lr_TV_8[i], u) - x_lr_TV_8[i
    ↪ ,5]
  e_psidot_15 = desired_values(x_lr_TV_15[i], u) - x_lr_TV_15[
    ↪ i,5]

  # Calculate control input feedbacks from yaw rate error
    ↪ using a PD controller
  ∆T_psidot_4 = kp * e_psidot_4 + kd * (e_psidot_4 -
    ↪ e_psidot_prev_4)/dt
```

```
    ΔT_psidot_8 = kp * e_psidot_8 + kd * (e_psidot_8 -
       ↪ e_psidot_prev_8)/dt
    ΔT_psidot_15 = kp * e_psidot_15 + kd * (e_psidot_15 -
       ↪ e_psidot_prev_15)/dt

    # Apply saturation to ensure the total torque adjustment is
       ↪ within the desired range (-T_E/4 <= ΔT <= T_E/4)
    ΔT_4 = max(- T_E_4/4 , min(ΔT_psidot_4, T_E_4/4 ))
    ΔT_8 = max(- T_E_8/4 , min(ΔT_psidot_8, T_E_8/4 ))
    ΔT_15 = max(- T_E_15/4 , min(ΔT_psidot_15, T_E_15/4 ))

    # Update the previous errors for the next iteration
    e_psidot_prev_4 = e_psidot_4
    e_psidot_prev_8 = e_psidot_8
    e_psidot_prev_15 = e_psidot_15

    # Apply torque vectoring by adding the corrective torque to
       ↪ each wheel
    u_controlled_4 = (v_delta, 0.2 * g, 2*ΔT_4, 0.5)
    u_controlled_8 = (v_delta, 0.2 * g, 2*ΔT_8, 0.5)
    u_controlled_15 = (v_delta, 0.2 * g, 2*ΔT_15, 0.5)

    # Simulate the multi-body kinematic model with the updated
       ↪ control input
    x_lr_TV_4[i + 1] = odeint(func_MB, x_lr_TV_4[i], [0, dt],
       ↪ args=(u_controlled_4, p))[1]
    x_lr_TV_8[i + 1] = odeint(func_MB, x_lr_TV_8[i], [0, dt],
       ↪ args=(u_controlled_8, p))[1]
    x_lr_TV_15[i + 1] = odeint(func_MB, x_lr_TV_15[i], [0, dt],
       ↪ args=(u_controlled_15, p))[1]

    yaw_rate_desired_4.append(desired_values(x_lr_TV_4[i], u))
    yaw_rate_desired_8.append(desired_values(x_lr_TV_8[i], u))
    yaw_rate_desired_15.append(desired_values(x_lr_TV_15[i], u))
```

⌞ Πατήστε εδώ για να επιστρέψετε στο κυρίως κείμενο.

Απόσπασμα Κώδικα Α.15: Επιταχυνόμενη αριστερή στροφή με ρυθμό αύξησης γωνίας διεύθυνσης $v_\delta = 0.05\,rad/s$, επιτάχυνση $a_{\text{long}} = 0.2g$ και εμπρόσθιο-οπίσθιο TV

```
from vehiclemodels.utils.steering_constraints import
    ↪ steering_constraints
from vehiclemodels.utils.acceleration_constraints import
    ↪ acceleration_constraints

# Initialize arrays, variables & lists for the control loop
x_fr_TV_4 = x_acc_4.copy()
x_fr_TV_8 = x_acc_8.copy()
x_fr_TV_15 = x_acc_15.copy()
e_psidot_prev_4 = 0
e_psidot_prev_8 = 0
e_psidot_prev_15 = 0
yaw_rate_desired_4 = []
yaw_rate_desired_8 = []
yaw_rate_desired_15 = []

kp = 2
kd = 400

for i in range(len(time)-1):

  acc_4 = acceleration_constraints(x_fr_TV_4[i,3],  0.2 * g, p
    ↪ .longitudinal)
  acc_8 = acceleration_constraints(x_fr_TV_8[i,3],  0.2 * g, p
    ↪ .longitudinal)
  acc_15 = acceleration_constraints(x_fr_TV_15[i,3],  0.2 * g,
    ↪  p.longitudinal)

  T_E_4 =  p.m * p.R_w * acc_4
  T_E_8 =  p.m * p.R_w * acc_8
  T_E_15 =  p.m * p.R_w * acc_15

  # Calculate errors between desired and actual yaw rate
  e_psidot_4 = desired_values(x_fr_TV_4[i], u) - x_fr_TV_4[i
    ↪ ,5]
  e_psidot_8 = desired_values(x_fr_TV_8[i], u) - x_fr_TV_8[i
    ↪ ,5]
  e_psidot_15 = desired_values(x_fr_TV_15[i], u) - x_fr_TV_15[
    ↪ i,5]

  # Calculate control input feedbacks from yaw rate error
    ↪ using a PD controller
  ΔT_psidot_4 = kp * e_psidot_4 + kd * (e_psidot_4 -
    ↪ e_psidot_prev_4)/dt
```

```
    ΔT_psidot_8 = kp * e_psidot_8 + kd * (e_psidot_8 -
        ↪ e_psidot_prev_8)/dt
    ΔT_psidot_15 = kp * e_psidot_15 + kd * (e_psidot_15 -
        ↪ e_psidot_prev_15)/dt

    # Apply saturation to ensure the total torque adjustment is
        ↪ within the desired range (-T_E/4 <= ΔT <= T_E/4)
    ΔT_4 = max(- T_E_4/4 , min(ΔT_psidot_4, T_E_4/4 ))
    ΔT_8 = max(- T_E_8/4 , min(ΔT_psidot_8, T_E_8/4 ))
    ΔT_15 = max(- T_E_15/4 , min(ΔT_psidot_15, T_E_15/4 ))

    eng_tsplit_4 = tune_T_se(e_psidot_4)
    eng_tsplit_8 = tune_T_se(e_psidot_8)
    eng_tsplit_15 = tune_T_se(e_psidot_15)

    # Update the previous errors for the next iteration
    e_psidot_prev_4 = e_psidot_4
    e_psidot_prev_8 = e_psidot_8
    e_psidot_prev_15 = e_psidot_15

    # Apply torque vectoring by adding the corrective torque to
        ↪ each wheel

    u_controlled_4 = (v_delta, 0.2 * g, 0, eng_tsplit_4)
    u_controlled_8 = (v_delta, 0.2 * g, 0, eng_tsplit_8)
    u_controlled_15 = (v_delta, 0.2 * g, 0, eng_tsplit_15)

    # Simulate the multi-body kinematic model with the updated
        ↪ control input
    x_fr_TV_4[i + 1] = odeint(func_MB, x_fr_TV_4[i], [0, dt],
        ↪ args=(u_controlled_4, p))[1]
    x_fr_TV_8[i + 1] = odeint(func_MB, x_fr_TV_8[i], [0, dt],
        ↪ args=(u_controlled_8, p))[1]
    x_fr_TV_15[i + 1] = odeint(func_MB, x_fr_TV_15[i], [0, dt],
        ↪ args=(u_controlled_15, p))[1]

    yaw_rate_desired_4.append(desired_values(x_fr_TV_4[i], u))
    yaw_rate_desired_8.append(desired_values(x_fr_TV_8[i], u))
    yaw_rate_desired_15.append(desired_values(x_fr_TV_15[i], u))
```

⌞ Πατήστε εδώ για να επιστρέψετε στο κυρίως κείμενο.

Απόσπασμα Κώδικα Α.16: Επιταχυνόμενη αριστερή στροφή με ρυθμό αύξησης γωνίας διεύθυνσης $v_\delta = 0.05\,rad/s$, επιτάχυνση $a_{\text{long}} = 0.2g$ και αριστερό-δεξί & εμπρόσθιο-οπίσθιο TV

```
from vehiclemodels.utils.steering_constraints import
    ↪ steering_constraints
from vehiclemodels.utils.acceleration_constraints import
    ↪ acceleration_constraints

# Initialize arrays, variables & lists for the control loop
x_lr_fr_TV_4 = x_acc_4.copy()
x_lr_fr_TV_8 = x_acc_8.copy()
x_lr_fr_TV_15 = x_acc_15.copy()
e_psidot_prev_4 = 0
e_psidot_prev_8 = 0
e_psidot_prev_15 = 0
yaw_rate_desired_4 = []
yaw_rate_desired_8 = []
yaw_rate_desired_15 = []
t_split = []

kp = 2
kd = 400

for i in range(len(time)-1):

  acc_4 = acceleration_constraints(x_lr_fr_TV_4[i,3],  0.2 * g
    ↪ , p.longitudinal)
  acc_8 = acceleration_constraints(x_lr_fr_TV_8[i,3],  0.2 * g
    ↪ , p.longitudinal)
  acc_15 = acceleration_constraints(x_lr_fr_TV_15[i,3],  0.2 *
    ↪  g, p.longitudinal)

  T_E_4 =  p.m * p.R_w * acc_4
  T_E_8 =  p.m * p.R_w * acc_8
  T_E_15 =  p.m * p.R_w * acc_15

  # Calculate errors between desired and actual yaw rate
  e_psidot_4 = desired_values(x_lr_fr_TV_4[i], u) -
    ↪ x_lr_fr_TV_4[i,5]
  e_psidot_8 = desired_values(x_lr_fr_TV_8[i], u) -
    ↪ x_lr_fr_TV_8[i,5]
  e_psidot_15 = desired_values(x_lr_fr_TV_15[i], u) -
    ↪ x_lr_fr_TV_15[i,5]

  # Calculate control input feedbacks from yaw rate error
    ↪ using a PD controller
```

```
35  ΔT_psidot_4 = kp * e_psidot_4 + kd * (e_psidot_4 -
       ↪ e_psidot_prev_4)/dt
36  ΔT_psidot_8 = kp * e_psidot_8 + kd * (e_psidot_8 -
       ↪ e_psidot_prev_8)/dt
37  ΔT_psidot_15 = kp * e_psidot_15 + kd * (e_psidot_15 -
       ↪ e_psidot_prev_15)/dt
38
39  # Apply saturation to ensure the total torque adjustment is
       ↪ within the desired range (-T_E/4 <= ΔT <= T_E/4)
40  ΔT_4 = max(- T_E_4/4 , min(ΔT_psidot_4, T_E_4/4 ))
41  ΔT_8 = max(- T_E_8/4 , min(ΔT_psidot_8, T_E_8/4 ))
42  ΔT_15 = max(- T_E_15/4 , min(ΔT_psidot_15, T_E_15/4 ))
43
44  eng_tsplit_4 = tune_T_se(e_psidot_4)
45  eng_tsplit_8 = tune_T_se(e_psidot_8)
46  eng_tsplit_15 = tune_T_se(e_psidot_15)
47
48  # Update the previous errors for the next iteration
49  e_psidot_prev_4 = e_psidot_4
50  e_psidot_prev_8 = e_psidot_8
51  e_psidot_prev_15 = e_psidot_15
52
53  # Apply torque vectoring by adding the corrective torque to
       ↪ each wheel
54  u_controlled_4 = (v_delta, 0.2 * g, 2*ΔT_4, eng_tsplit_4)
55  u_controlled_8 = (v_delta, 0.2 * g, 2*ΔT_8, eng_tsplit_8)
56  u_controlled_15 = (v_delta, 0.2 * g, 2*ΔT_15, eng_tsplit_15)
57
58  # Simulate the multi-body kinematic model with the updated
       ↪ control input
59  x_lr_fr_TV_4[i + 1] = odeint(func_MB, x_lr_fr_TV_4[i], [0,
       ↪ dt], args=(u_controlled_4, p))[1]
60  x_lr_fr_TV_8[i + 1] = odeint(func_MB, x_lr_fr_TV_8[i], [0,
       ↪ dt], args=(u_controlled_8, p))[1]
61  x_lr_fr_TV_15[i + 1] = odeint(func_MB, x_lr_fr_TV_15[i], [0,
       ↪  dt], args=(u_controlled_15, p))[1]
62
63  yaw_rate_desired_4.append(desired_values(x_lr_fr_TV_4[i], u)
       ↪ )
64  yaw_rate_desired_8.append(desired_values(x_lr_fr_TV_8[i], u)
       ↪ )
65  yaw_rate_desired_15.append(desired_values(x_lr_fr_TV_15[i],
       ↪ u))
```

⌞ Πατήστε εδώ για να επιστρέψετε στο κυρίως κείμενο.

Απόσπασμα Κώδικα Α.17: Υλοποίηση ελιγμού step-steer με επιβολή ακαριαίας γωνίας διεύθυνσης $\delta = 0.314\,rad$ και επιτάχυνση $a_{\text{long}} = 0.3g$

```
from scipy.integrate import odeint
import numpy as np
import matplotlib.pyplot as plt
import math

from vehiclemodels.parameters_vehicle2 import parameters_vehicle2
from vehiclemodels.init_mb import init_mb
from vehiclemodels.vehicle_dynamics_mb import vehicle_dynamics_mb

def func_MB(x, t, u, p):
    f = vehicle_dynamics_mb(x, u, p)
    return f

# load parameters
p = parameters_vehicle2()
g = 9.81  # [m/s^2]

# set options
# ------------------------------------------------------------------

tStart = 0  # start time
tFinal = 8  # end time
dt = 0.01  # time steps
time = np.arange(0, tFinal, dt)

delta0 = 0.314
vel0 = 4
Psi0 = 0
dotPsi0 = 0
beta0 = 0
sy0 = 0
initialState_4 = [0, sy0, delta0, vel0, Psi0, dotPsi0, beta0]  # initial state for simulation
x0_MB_4 = init_mb(initialState_4, p)  # initial state for multi-body model
x_ss_4 = np.full((len(time), 29), x0_MB_4)
vel0 = 8
initialState_8 = [0, sy0, delta0, vel0, Psi0, dotPsi0, beta0]  # initial state for simulation
x0_MB_8 = init_mb(initialState_8, p)  # initial state for multi-body model
x_ss_8 = np.full((len(time), 29), x0_MB_8)
```

```
vel0 = 15
initialState_15 = [0, sy0, delta0, vel0, Psi0, dotPsi0, beta0]   # initial state for simulation
x0_MB_15 = init_mb(initialState_15, p)  # initial state for multi-body model
x_ss_15 = np.full((len(time), 29), x0_MB_15)

v_delta = 0

for i in range(len(time)-1):

  # Set vehicle input parameters for each time step
  u_ss = [v_delta, 0.3*g, 0, 0.5]

  # Simulate the multi-body kinematic model for each time step
  x_ss_4[i+1] = odeint(func_MB, x_ss_4[i], [0, dt], args=(u_ss, p))[-1]
  x_ss_8[i+1] = odeint(func_MB, x_ss_8[i], [0, dt], args=(u_ss, p))[-1]
  x_ss_15[i+1] = odeint(func_MB, x_ss_15[i], [0, dt], args=(u_ss, p))[-1]

# position
plt.title('Θέση')
plt.plot([tmp[0] for tmp in x_ss_4], [tmp[1] for tmp in x_ss_4])
plt.plot([tmp[0] for tmp in x_ss_8], [tmp[1] for tmp in x_ss_8])
plt.plot([tmp[0] for tmp in x_ss_15], [tmp[1] for tmp in x_ss_15])
plt.legend(['u0 = 4 m/s', 'u0 = 8 m/s', 'u0 = 15 m/s'])
plt.xlabel('x-θέση (m)')
plt.ylabel('y-θέση (m)')
plt.grid(True)
plt.show()

# orientation
plt.title('Εκτροπή')
plt.plot(time, [tmp[4] for tmp in x_ss_4])
plt.plot(time, [tmp[4] for tmp in x_ss_8])
plt.plot(time, [tmp[4] for tmp in x_ss_15])
plt.legend(['u0 = 4 m/s', 'u0 = 8 m/s', 'u0 = 15 m/s'])
plt.xlabel('Χρόνος (s)')
plt.ylabel('Εκτροπή (rad)')
plt.grid(True)
plt.show()

# yaw rate
```

```
77 plt.title('Ρυθμός εκτροπής')
78 plt.plot(time, [tmp[5] for tmp in x_ss_4])
79 plt.plot(time, [tmp[5] for tmp in x_ss_8])
80 plt.plot(time, [tmp[5] for tmp in x_ss_15])
81 plt.legend(['u0 = 4 m/s', 'u0 = 8 m/s', 'u0 = 15 m/s'])
82 plt.xlabel('Χρόνος (s)')
83 plt.ylabel('Ρυθμός εκτροπής (rad/s)')
84 plt.grid(True)
85 plt.show()
```

⌞ Πατήστε εδώ για να επιστρέψετε στο κυρίως κείμενο.

Απόσπασμα Κώδικα Α.18: Υλοποίηση ελιγμού step-steer με επιβολή ακαριαίας γωνίας διεύθυνσης $\delta = 0.314\,rad$, επιτάχυνση $a_{\text{long}} = 0.3g$ και αριστερό-δεξί TV

```
from vehiclemodels.utils.steering_constraints import
    ↪ steering_constraints
from vehiclemodels.utils.acceleration_constraints import
    ↪ acceleration_constraints

# Initialize arrays, variables & lists for the control loop
x_ss_lr_TV_4 = x_ss_4.copy()
x_ss_lr_TV_8 = x_ss_8.copy()
x_ss_lr_TV_15 = x_ss_15.copy()
e_psidot_prev_4 = 0
e_psidot_prev_8 = 0
e_psidot_prev_15 = 0
yaw_rate_desired_4 = []
yaw_rate_desired_8 = []
yaw_rate_desired_15 = []

kp = 2
kd = 400

for i in range(len(time)-1):

  acc_4 = acceleration_constraints(x_ss_lr_TV_4[i,3],  0.3 * g
     ↪ , p.longitudinal)
  acc_8 = acceleration_constraints(x_ss_lr_TV_8[i,3],  0.3 * g
     ↪ , p.longitudinal)
  acc_15 = acceleration_constraints(x_ss_lr_TV_15[i,3],  0.3 *
     ↪  g, p.longitudinal)

  T_E_4 =  p.m * p.R_w * acc_4
  T_E_8 =  p.m * p.R_w * acc_8
  T_E_15 =  p.m * p.R_w * acc_15

  # Calculate errors between desired and actual yaw rate
  e_psidot_4 = desired_values(x_ss_lr_TV_4[i], u_ss) -
     ↪ x_ss_lr_TV_4[i,5]
  e_psidot_8 = desired_values(x_ss_lr_TV_8[i], u_ss) -
     ↪ x_ss_lr_TV_8[i,5]
  e_psidot_15 = desired_values(x_ss_lr_TV_15[i], u_ss) -
     ↪ x_ss_lr_TV_15[i,5]

  # Calculate control input feedbacks from yaw rate error
     ↪ using a PD controller
  ∆T_psidot_4 = kp * e_psidot_4 + kd * (e_psidot_4 -
     ↪ e_psidot_prev_4)/dt
```

```
    ΔT_psidot_8 = kp * e_psidot_8 + kd * (e_psidot_8 -
        ↪ e_psidot_prev_8)/dt
    ΔT_psidot_15 = kp * e_psidot_15 + kd * (e_psidot_15 -
        ↪ e_psidot_prev_15)/dt

    # Apply saturation to ensure the total torque adjustment is
        ↪ within the desired range (-T_E/4 <= ΔT <= T_E/4)
    ΔT_4 = max(- T_E_4/4 , min(ΔT_psidot_4, T_E_4/4 ))
    ΔT_8 = max(- T_E_8/4 , min(ΔT_psidot_8, T_E_8/4 ))
    ΔT_15 = max(- T_E_15/4 , min(ΔT_psidot_15, T_E_15/4 ))

    # Update the previous errors for the next iteration
    e_psidot_prev_4 = e_psidot_4
    e_psidot_prev_8 = e_psidot_8
    e_psidot_prev_15 = e_psidot_15

    # Apply torque vectoring by adding the corrective torque to
        ↪ each wheel
    u_controlled_4 = (v_delta, 0.3 * g, 2*ΔT_4, 0.5)
    u_controlled_8 = (v_delta, 0.3 * g, 2*ΔT_8, 0.5)
    u_controlled_15 = (v_delta, 0.3 * g, 2*ΔT_15, 0.5)

    # Simulate the multi-body kinematic model with the updated
        ↪ control input
    x_ss_lr_TV_4[i + 1] = odeint(func_MB, x_ss_lr_TV_4[i], [0,
        ↪ dt], args=(u_controlled_4, p))[1]
    x_ss_lr_TV_8[i + 1] = odeint(func_MB, x_ss_lr_TV_8[i], [0,
        ↪ dt], args=(u_controlled_8, p))[1]
    x_ss_lr_TV_15[i + 1] = odeint(func_MB, x_ss_lr_TV_15[i], [0,
        ↪  dt], args=(u_controlled_15, p))[1]

    yaw_rate_desired_4.append(desired_values(x_ss_lr_TV_4[i],
        ↪ u_ss))
    yaw_rate_desired_8.append(desired_values(x_ss_lr_TV_8[i],
        ↪ u_ss))
    yaw_rate_desired_15.append(desired_values(x_ss_lr_TV_15[i],
        ↪ u_ss))
```

⌞ Πατήστε εδώ για να επιστρέψετε στο κυρίως κείμενο.

Απόσπασμα Κώδικα A.19: Υλοποίηση ελιγμού step-steer με επιβολή ακαριαίας γωνίας διεύθυνσης $\delta = 0.314\,rad$, επιτάχυνση $a_{\text{long}} = 0.3g$ και εμπρόσθιο-οπίσθιο TV

```
from vehiclemodels.utils.steering_constraints import
    ↪ steering_constraints
from vehiclemodels.utils.acceleration_constraints import
    ↪ acceleration_constraints

# Initialize arrays, variables & lists for the control loop
x_ss_fr_TV_4 = x_ss_4.copy()
x_ss_fr_TV_8 = x_ss_8.copy()
x_ss_fr_TV_15 = x_ss_15.copy()
e_psidot_prev_4 = 0
e_psidot_prev_8 = 0
e_psidot_prev_15 = 0
yaw_rate_desired_4 = []
yaw_rate_desired_8 = []
yaw_rate_desired_15 = []

kp = 2
kd = 400

for i in range(len(time)-1):

  acc_4 = acceleration_constraints(x_ss_fr_TV_4[i,3],  0.3 * g
     ↪ , p.longitudinal)
  acc_8 = acceleration_constraints(x_ss_fr_TV_8[i,3],  0.3 * g
     ↪ , p.longitudinal)
  acc_15 = acceleration_constraints(x_ss_fr_TV_15[i,3],  0.3 *
     ↪  g, p.longitudinal)

  T_E_4 =  p.m * p.R_w * acc_4
  T_E_8 =  p.m * p.R_w * acc_8
  T_E_15 =  p.m * p.R_w * acc_15

  # Calculate errors between desired and actual yaw rate
  e_psidot_4 = desired_values(x_ss_fr_TV_4[i], u_ss) -
     ↪ x_ss_fr_TV_4[i,5]
  e_psidot_8 = desired_values(x_ss_fr_TV_8[i], u_ss) -
     ↪ x_ss_fr_TV_8[i,5]
  e_psidot_15 = desired_values(x_ss_fr_TV_15[i], u_ss) -
     ↪ x_ss_fr_TV_15[i,5]

  # Calculate control input feedbacks from yaw rate error
     ↪ using a PD controller
  ∆T_psidot_4 = kp * e_psidot_4 + kd * (e_psidot_4 -
     ↪ e_psidot_prev_4)/dt
```

```
    ΔT_psidot_8 = kp * e_psidot_8 + kd * (e_psidot_8 -
        ↪ e_psidot_prev_8)/dt
    ΔT_psidot_15 = kp * e_psidot_15 + kd * (e_psidot_15 -
        ↪ e_psidot_prev_15)/dt

    # Apply saturation to ensure the total torque adjustment is
        ↪ within the desired range (-T_E/4 <= ΔT <= T_E/4)
    ΔT_4 = max(- T_E_4/4 , min(ΔT_psidot_4, T_E_4/4 ))
    ΔT_8 = max(- T_E_8/4 , min(ΔT_psidot_8, T_E_8/4 ))
    ΔT_15 = max(- T_E_15/4 , min(ΔT_psidot_15, T_E_15/4 ))

    eng_tsplit_4 = tune_T_se(e_psidot_4)
    eng_tsplit_8 = tune_T_se(e_psidot_8)
    eng_tsplit_15 = tune_T_se(e_psidot_15)

    # Update the previous errors for the next iteration
    e_psidot_prev_4 = e_psidot_4
    e_psidot_prev_8 = e_psidot_8
    e_psidot_prev_15 = e_psidot_15

    # Apply torque vectoring by adding the corrective torque to
        ↪ each wheel
    u_controlled_4 = (v_delta, 0.3 * g, 0, eng_tsplit_4)
    u_controlled_8 = (v_delta, 0.3 * g, 0, eng_tsplit_8)
    u_controlled_15 = (v_delta, 0.3 * g, 0, eng_tsplit_15)

    # Simulate the multi-body kinematic model with the updated
        ↪ control input
    x_ss_fr_TV_4[i + 1] = odeint(func_MB, x_ss_fr_TV_4[i], [0,
        ↪ dt], args=(u_controlled_4, p))[1]
    x_ss_fr_TV_8[i + 1] = odeint(func_MB, x_ss_fr_TV_8[i], [0,
        ↪ dt], args=(u_controlled_8, p))[1]
    x_ss_fr_TV_15[i + 1] = odeint(func_MB, x_ss_fr_TV_15[i], [0,
        ↪  dt], args=(u_controlled_15, p))[1]

    yaw_rate_desired_4.append(desired_values(x_ss_fr_TV_4[i],
        ↪ u_ss))
    yaw_rate_desired_8.append(desired_values(x_ss_fr_TV_8[i],
        ↪ u_ss))
    yaw_rate_desired_15.append(desired_values(x_ss_fr_TV_15[i],
        ↪ u_ss))
```

⌞ Πατήστε εδώ για να επιστρέψετε στο κυρίως κείμενο.

Απόσπασμα Κώδικα Α.20: Υλοποίηση ελιγμού step-steer με επιβολή ακαριαίας γωνίας διεύθυνσης $\delta = 0.314\,rad$, επιτάχυνση $a_{\text{long}} = 0.3g$ και αριστερό-δεξί & εμπρόσθιο-οπίσθιο TV

```
from vehiclemodels.utils.steering_constraints import
    ↪ steering_constraints
from vehiclemodels.utils.acceleration_constraints import
    ↪ acceleration_constraints

# Initialize arrays, variables & lists for the control loop
x_ss_lr_fr_TV_4 = x_ss_4.copy()
x_ss_lr_fr_TV_8 = x_ss_8.copy()
x_ss_lr_fr_TV_15 = x_ss_15.copy()
e_psidot_prev_4 = 0
e_psidot_prev_8 = 0
e_psidot_prev_15 = 0
yaw_rate_desired_4 = []
yaw_rate_desired_8 = []
yaw_rate_desired_15 = []

kp = 2
kd = 400

for i in range(len(time)-1):

  acc_4 = acceleration_constraints(x_ss_lr_fr_TV_4[i,3],  0.3
    ↪ * g, p.longitudinal)
  acc_8 = acceleration_constraints(x_ss_lr_fr_TV_8[i,3],  0.3
    ↪ * g, p.longitudinal)
  acc_15 = acceleration_constraints(x_ss_lr_fr_TV_15[i,3],
    ↪ 0.3 * g, p.longitudinal)

  T_E_4 =  p.m * p.R_w * acc_4
  T_E_8 =  p.m * p.R_w * acc_8
  T_E_15 =  p.m * p.R_w * acc_15

  # Calculate errors between desired and actual yaw rate
  e_psidot_4 = desired_values(x_ss_lr_fr_TV_4[i], u_ss) -
    ↪ x_ss_lr_fr_TV_4[i,5]
  e_psidot_8 = desired_values(x_ss_lr_fr_TV_8[i], u_ss) -
    ↪ x_ss_lr_fr_TV_8[i,5]
  e_psidot_15 = desired_values(x_ss_lr_fr_TV_15[i], u_ss) -
    ↪ x_ss_lr_fr_TV_15[i,5]

  # Calculate control input feedbacks from yaw rate error
    ↪ using a PD controller
  ∆T_psidot_4 = kp * e_psidot_4 + kd * (e_psidot_4 -
```

```
    ↪ e_psidot_prev_4)/dt
35  ΔT_psidot_8 = kp * e_psidot_8 + kd * (e_psidot_8 -
    ↪ e_psidot_prev_8)/dt
36  ΔT_psidot_15 = kp * e_psidot_15 + kd * (e_psidot_15 -
    ↪ e_psidot_prev_15)/dt
37
38  # Apply saturation to ensure the total torque adjustment is
    ↪ within the desired range (-T_E/4 <= ΔT <= T_E/4)
39  ΔT_4 = max(- T_E_4/4 , min(ΔT_psidot_4, T_E_4/4 ))
40  ΔT_8 = max(- T_E_8/4 , min(ΔT_psidot_8, T_E_8/4 ))
41  ΔT_15 = max(- T_E_15/4 , min(ΔT_psidot_15, T_E_15/4 ))
42
43  eng_tsplit_4 = tune_T_se(e_psidot_4)
44  eng_tsplit_8 = tune_T_se(e_psidot_8)
45  eng_tsplit_15 = tune_T_se(e_psidot_15)
46
47  # Update the previous errors for the next iteration
48  e_psidot_prev_4 = e_psidot_4
49  e_psidot_prev_8 = e_psidot_8
50  e_psidot_prev_15 = e_psidot_15
51
52  # Apply torque vectoring by adding the corrective torque to
    ↪ each wheel
53  u_controlled_4 = (v_delta, 0.3 * g, ΔT_4, eng_tsplit_4)
54  u_controlled_8 = (v_delta, 0.3 * g, ΔT_8, eng_tsplit_8)
55  u_controlled_15 = (v_delta, 0.3 * g, ΔT_15, eng_tsplit_15)
56
57  # Simulate the multi-body kinematic model with the updated
    ↪ control input
58  x_ss_lr_fr_TV_4[i + 1] = odeint(func_MB, x_ss_lr_fr_TV_4[i],
    ↪  [0, dt], args=(u_controlled_4, p))[1]
59  x_ss_lr_fr_TV_8[i + 1] = odeint(func_MB, x_ss_lr_fr_TV_8[i],
    ↪  [0, dt], args=(u_controlled_8, p))[1]
60  x_ss_lr_fr_TV_15[i + 1] = odeint(func_MB, x_ss_lr_fr_TV_15[i
    ↪ ], [0, dt], args=(u_controlled_15, p))[1]
61
62  yaw_rate_desired_4.append(desired_values(x_ss_lr_fr_TV_4[i],
    ↪  u_ss))
63  yaw_rate_desired_8.append(desired_values(x_ss_lr_fr_TV_8[i],
    ↪  u_ss))
64  yaw_rate_desired_15.append(desired_values(x_ss_lr_fr_TV_15[i
    ↪ ], u_ss))
```

⌞ Πατήστε εδώ για να επιστρέψετε στο κυρίως κείμενο.

Απόσπασμα Κώδικα Α.21: Πειραματισμός με το Single Track Model του Commonroad προς εξαγωγή επιθυμητής τροχιάς για τον ελιγμό αποφυγής εμποδίου

```
from scipy.integrate import odeint
import numpy as np
import math
import matplotlib.pyplot as plt

from vehiclemodels.init_std import init_std
from vehiclemodels.parameters_vehicle2 import
    ↪ parameters_vehicle2
from vehiclemodels.vehicle_dynamics_std import
    ↪ vehicle_dynamics_std   # Importing the Single Track Drift
    ↪  (STD) vehicle kinematics model


def func_STD(x, t, u, p):
    # Obtaining vehicle dynamics
    f = vehicle_dynamics_std(x, u, p)
    return f

# Set initial state parameters for simulation
x0 = 0                      #x-position in a global coordinate
    ↪ system
y0 = 0                      #y-position in a global coordinate
    ↪ system
delta0 = 0              #steering angle of front wheels
v_0 = 4                     #velocity at vehicle center
yaw_ang0 = 0               #yaw angle
yaw_rate0 = 0            #yaw rate
slip_ang0 = 0               #slip angle at vehicle center

# Load vehicle parameters
p = parameters_vehicle2()

# Set simulation time parameters
g = 9.81  # [m/s^2]
t_start = 0
t_final = 5
dt = 0.01
time = np.arange(t_start, t_final, dt)

# Set initial state
initial_state_std = [x0, y0, delta0, v_0, yaw_ang0, yaw_rate0,
    ↪  slip_ang0]
x0_STD = init_std(initial_state_std, p)
x_STD = np.full((len(time),9), x0_STD)
```

```
39
40 for i in range(len(time)-1):
41
42   # Set vehicle input parameters for each time step
43   u = (-0.5 * np.cos((2*np.pi/4)*time[i] + np.pi/35), 0.2 * g)
       ↪  # Steering angle velocity & long. acceleration values
44
45   # Simulate the single-track kinematic model for each time
       ↪ step
46   x_STD[i + 1] = odeint(func_STD, x_STD[i], [0, dt], args=(u,
       ↪ p))[-1]
```

⌞ Πατήστε εδώ για να επιστρέψετε στο κυρίως κείμενο.

Απόσπασμα Κώδικα Α.22: Ελιγμός αποφυγής εμποδίου με επιτάχυνση $a_{\text{long}} = 0.2g$

```
from scipy.integrate import odeint
import numpy as np
import matplotlib.pyplot as plt
import math

from vehiclemodels.parameters_vehicle2 import
    ↪ parameters_vehicle2
from vehiclemodels.init_mb import init_mb
from vehiclemodels.vehicle_dynamics_mb import
    ↪ vehicle_dynamics_mb

def func_MB(x, t, u, p):
    f = vehicle_dynamics_mb(x, u, p)
    return f

# load parameters
p = parameters_vehicle2()
g = 9.81  # [m/s^2]

# set options
    ↪ ------------------------------------------------------------
    ↪
tStart = 0  # start time
tFinal = 7.85  # end time
dt = 0.01  # time steps
time = np.arange(0, tFinal, dt)

delta0 = 0
vel0 = 4
Psi0 = 0
dotPsi0 = 0
beta0 = 0
sy0 = 0
initialState_4 = [0, sy0, delta0, vel0, Psi0, dotPsi0, beta0]
    ↪  # initial state for simulation
x0_MB_4 = init_mb(initialState_4, p)  # initial state for
    ↪ multi-body model
x_oa_4 = np.full((len(time), 29), x0_MB_4)
vel0 = 8
initialState_8 = [0, sy0, delta0, vel0, Psi0, dotPsi0, beta0]
    ↪  # initial state for simulation
x0_MB_8 = init_mb(initialState_8, p)  # initial state for
    ↪ multi-body model
x_oa_8 = np.full((len(time), 29), x0_MB_8)
vel0 = 15
```

```
initialState_15 = [0, sy0, delta0, vel0, Psi0, dotPsi0, beta0]   # initial state for simulation
x0_MB_15 = init_mb(initialState_15, p)  # initial state for multi-body model
x_oa_15 = np.full((len(time), 29), x0_MB_15)

for i in range(len(time)-1):

  # Set vehicle input parameters for each time step
  v_delta_4 = -0.5 * np.cos((2*np.pi/4)*time[i] + np.pi/35)
  v_delta_8 = -0.5 * np.cos((2*np.pi/4)*time[i] + np.pi/10.1)
  v_delta_15 = -0.5 * np.cos((2*np.pi/4)*time[i] + np.pi/6.13)

  u_oa_4 = [v_delta_4, 0.2*g, 0, 0.5]
  u_oa_8 = [v_delta_8, 0.2*g, 0, 0.5]
  u_oa_15 = [v_delta_15, 0.2*g, 0, 0.5]

  # Simulate the multi-body kinematic model for each time step
  x_oa_4[i+1] = odeint(func_MB, x_oa_4[i], [0, dt], args=(u_oa_4, p))[-1]
  x_oa_8[i+1] = odeint(func_MB, x_oa_8[i], [0, dt], args=(u_oa_8, p))[-1]
  x_oa_15[i+1] = odeint(func_MB, x_oa_15[i], [0, dt], args=(u_oa_15, p))[-1]

# position
plt.title('Θέση')
plt.plot([tmp[0] for tmp in x_oa_4[:455]], [tmp[1] for tmp in x_oa_4[:455]])
plt.plot([tmp[0] for tmp in x_oa_8[:455]], [tmp[1] for tmp in x_oa_8[:455]])
plt.plot([tmp[0] for tmp in x_oa_15[:455]], [tmp[1] for tmp in  x_oa_15[:455]])
plt.legend(['u0 = 4 m/s', 'u0 = 8 m/s', 'u0 = 15 m/s'])
plt.xlabel('x-θέση (m)')
plt.ylabel('y-θέση (m)')
plt.grid(True)
plt.show()

# orientation
plt.title('Γωνία εκτροπής')
plt.plot(time[:455], [tmp[4] for tmp in x_oa_4[:455]])
plt.plot(time[:455], [tmp[4] for tmp in x_oa_8[:455]])
plt.plot(time[:455], [tmp[4] for tmp in x_oa_15[:455]])
plt.legend(['u0 = 4 m/s', 'u0 = 8 m/s', 'u0 = 15 m/s'])
plt.xlabel('Χρόνος (s)')
plt.ylabel('Γωνία εκτροπής (rad)')
plt.grid(True)
```

```
78 plt.show()
79
80 # yaw rate
81 plt.title('Ρυθμός εκτροπής')
82 plt.plot(time[:455], [tmp[5] for tmp in x_oa_4[:455]])
83 plt.plot(time[:455], [tmp[5] for tmp in x_oa_8[:455]])
84 plt.plot(time[:455], [tmp[5] for tmp in x_oa_15[:455]])
85 plt.legend(['u0 = 4 m/s', 'u0 = 8 m/s', 'u0 = 15 m/s'])
86 plt.xlabel('Χρόνος (s)')
87 plt.ylabel('Ρυθμός εκτροπής (rad/s)')
88 plt.grid(True)
89 plt.show()
```

⌞ Πατήστε εδώ για να επιστρέψετε στο κυρίως κείμενο.

Απόσπασμα Κώδικα Α.23: Ελιγμός αποφυγής εμποδίου με επιτάχυνση $a_{\text{long}} = 0.2g$ και αριστερό-δεξί TV

```
from vehiclemodels.utils.steering_constraints import
    ↪ steering_constraints
from vehiclemodels.utils.acceleration_constraints import
    ↪ acceleration_constraints

# Initialize arrays, variables & lists for the control loop
x_oa_lr_TV_4 = x_oa_4.copy()
x_oa_lr_TV_8 = x_oa_8.copy()
x_oa_lr_TV_15 = x_oa_15.copy()
e_psidot_prev_4 = 0
e_psidot_prev_8 = 0
e_psidot_prev_15 = 0
yaw_rate_desired_4 = []
yaw_rate_desired_8 = []
yaw_rate_desired_15 = []

kp = 2
kd = 400

for i in range(len(time)-1):

  acc_4 = acceleration_constraints(x_oa_lr_TV_4[i,3],  0.2 * g
     ↪ , p.longitudinal)
  acc_8 = acceleration_constraints(x_oa_lr_TV_8[i,3],  0.2 * g
     ↪ , p.longitudinal)
  acc_15 = acceleration_constraints(x_oa_lr_TV_15[i,3],  0.2 *
     ↪  g, p.longitudinal)

  T_E_4 =  p.m * p.R_w * acc_4
  T_E_8 =  p.m * p.R_w * acc_8
  T_E_15 =  p.m * p.R_w * acc_15

  # Calculate errors between desired and actual yaw rate
  e_psidot_4 = desired_values(x_oa_lr_TV_4[i], u_oa_4) -
     ↪ x_oa_lr_TV_4[i,5]
  e_psidot_8 = desired_values(x_oa_lr_TV_8[i], u_oa_8) -
     ↪ x_oa_lr_TV_8[i,5]
  e_psidot_15 = desired_values(x_oa_lr_TV_15[i], u_oa_15) -
     ↪ x_oa_lr_TV_15[i,5]

  # Calculate control input feedbacks from yaw rate error
     ↪ using a PD controller
  ΔT_psidot_4 = kp * e_psidot_4 + kd * (e_psidot_4 -
     ↪ e_psidot_prev_4)/dt
```

```
    ΔT_psidot_8 = kp * e_psidot_8 + kd * (e_psidot_8 -
        ↪ e_psidot_prev_8)/dt
    ΔT_psidot_15 = kp * e_psidot_15 + kd * (e_psidot_15 -
        ↪ e_psidot_prev_15)/dt

    # Apply saturation to ensure the total torque adjustment is
        ↪ within the desired range (-T_E/4 <= ΔT <= T_E/4)
    ΔT_4 = max(- T_E_4/4 , min(ΔT_psidot_4, T_E_4/4 ))
    ΔT_8 = max(- T_E_8/4 , min(ΔT_psidot_8, T_E_8/4 ))
    ΔT_15 = max(- T_E_15/4 , min(ΔT_psidot_15, T_E_15/4 ))

    # Update the previous errors for the next iteration
    e_psidot_prev_4 = e_psidot_4
    e_psidot_prev_8 = e_psidot_8
    e_psidot_prev_15 = e_psidot_15

    # Set vehicle input parameters for each time step
    v_delta_4 = -0.5 * np.cos((2*np.pi/4)*time[i] + np.pi/35)
    v_delta_8 = -0.5 * np.cos((2*np.pi/4)*time[i] + np.pi/10.1)
    v_delta_15 = -0.5 * np.cos((2*np.pi/4)*time[i] + np.pi/6.13)

    # Apply torque vectoring by adding the corrective torque to
        ↪ each wheel
    u_oa_4 = [v_delta_4, 0.2*g, 2*ΔT_4, 0.5]
    u_oa_8 = [v_delta_8, 0.2*g, 2*ΔT_8, 0.5]
    u_oa_15 = [v_delta_15, 0.2*g, 2*ΔT_15, 0.5]

    # Simulate the multi-body kinematic model with the updated
        ↪ control input
    x_oa_lr_TV_4[i + 1] = odeint(func_MB, x_oa_lr_TV_4[i], [0,
        ↪ dt], args=(u_oa_4, p))[1]
    x_oa_lr_TV_8[i + 1] = odeint(func_MB, x_oa_lr_TV_8[i], [0,
        ↪ dt], args=(u_oa_8, p))[1]
    x_oa_lr_TV_15[i + 1] = odeint(func_MB, x_oa_lr_TV_15[i], [0,
        ↪  dt], args=(u_oa_15, p))[1]

    yaw_rate_desired_4.append(desired_values(x_oa_lr_TV_4[i],
        ↪ u_oa_4))
    yaw_rate_desired_8.append(desired_values(x_oa_lr_TV_8[i],
        ↪ u_oa_8))
    yaw_rate_desired_15.append(desired_values(x_oa_lr_TV_15[i],
        ↪ u_oa_15))
```

⌞ Πατήστε εδώ για να επιστρέψετε στο κυρίως κείμενο.

Απόσπασμα Κώδικα Α.24: Ελιγμός αποφυγής εμποδίου με επιτάχυνση $a_{\text{long}} = 0.2g$ και εμπρόσθιο-οπίσθιο TV

```
from scipy.interpolate import interp1d
import pandas as pd

def tune_T_se(error):

  x = [-2.58, 2.58]
  y = [0.1,0.9]
  t = interp1d(x, y, bounds_error=False, fill_value=(0.1,0.9))
  t_eng = t(error)

  return t_eng.item()

  from vehiclemodels.utils.steering_constraints import
    ↪ steering_constraints
from vehiclemodels.utils.acceleration_constraints import
   ↪ acceleration_constraints

# Initialize arrays, variables & lists for the control loop
x_oa_fr_TV_4 = x_oa_4.copy()
x_oa_fr_TV_8 = x_oa_8.copy()
x_oa_fr_TV_15 = x_oa_15.copy()
e_psidot_prev_4 = 0
e_psidot_prev_8 = 0
e_psidot_prev_15 = 0
yaw_rate_desired_4 = []
yaw_rate_desired_8 = []
yaw_rate_desired_15 = []

kp = 2
kd = 400

for i in range(len(time)-1):

  acc_4 = acceleration_constraints(x_oa_fr_TV_4[i,3],  0.2 * g
    ↪ , p.longitudinal)
  acc_8 = acceleration_constraints(x_oa_fr_TV_8[i,3],  0.2 * g
    ↪ , p.longitudinal)
  acc_15 = acceleration_constraints(x_oa_fr_TV_15[i,3],  0.2 *
    ↪  g, p.longitudinal)

  T_E_4 =  p.m * p.R_w * acc_4
  T_E_8 =  p.m * p.R_w * acc_8
  T_E_15 =  p.m * p.R_w * acc_15

```

```
    # Calculate errors between desired and actual yaw rate
    e_psidot_4 = desired_values(x_oa_fr_TV_4[i], u_oa_4) -
       ↪ x_oa_fr_TV_4[i,5]
    e_psidot_8 = desired_values(x_oa_fr_TV_8[i], u_oa_8) -
       ↪ x_oa_fr_TV_8[i,5]
    e_psidot_15 = desired_values(x_oa_fr_TV_15[i], u_oa_15) -
       ↪ x_oa_fr_TV_15[i,5]

    # Calculate control input feedbacks from yaw rate error
       ↪ using a PD controller
    ΔT_psidot_4 = kp * e_psidot_4 + kd * (e_psidot_4 -
       ↪ e_psidot_prev_4)/dt
    ΔT_psidot_8 = kp * e_psidot_8 + kd * (e_psidot_8 -
       ↪ e_psidot_prev_8)/dt
    ΔT_psidot_15 = kp * e_psidot_15 + kd * (e_psidot_15 -
       ↪ e_psidot_prev_15)/dt

    # Apply saturation to ensure the total torque adjustment is
       ↪ within the desired range (-T_E/4 <= ΔT <= T_E/4)
    ΔT_4 = max(- T_E_4/4 , min(ΔT_psidot_4, T_E_4/4 ))
    ΔT_8 = max(- T_E_8/4 , min(ΔT_psidot_8, T_E_8/4 ))
    ΔT_15 = max(- T_E_15/4 , min(ΔT_psidot_15, T_E_15/4 ))

    eng_tsplit_4 = tune_T_se(e_psidot_4)
    eng_tsplit_8 = tune_T_se(e_psidot_8)
    eng_tsplit_15 = tune_T_se(e_psidot_15)

    # Update the previous errors for the next iteration
    e_psidot_prev_4 = e_psidot_4
    e_psidot_prev_8 = e_psidot_8
    e_psidot_prev_15 = e_psidot_15

    # Set vehicle input parameters for each time step
    v_delta_4 = -0.5 * np.cos((2*np.pi/4)*time[i] + np.pi/35)
    v_delta_8 = -0.5 * np.cos((2*np.pi/4)*time[i] + np.pi/10.1)
    v_delta_15 = -0.5 * np.cos((2*np.pi/4)*time[i] + np.pi/6.13)

    # Apply torque vectoring by adding the corrective torque to
       ↪ each wheel
    u_oa_4 = [v_delta_4, 0.2*g, 0, eng_tsplit_4]
    u_oa_8 = [v_delta_8, 0.2*g, 0, eng_tsplit_8]
    u_oa_15 = [v_delta_15, 0.2*g, 0, eng_tsplit_15]

    # Simulate the multi-body kinematic model with the updated
       ↪ control input
    x_oa_fr_TV_4[i + 1] = odeint(func_MB, x_oa_fr_TV_4[i], [0,
       ↪ dt], args=(u_oa_4, p))[1]
    x_oa_fr_TV_8[i + 1] = odeint(func_MB, x_oa_fr_TV_8[i], [0,
```

```
    ↪ dt], args=(u_oa_8, p))[1]
77  x_oa_fr_TV_15[i + 1] = odeint(func_MB, x_oa_fr_TV_15[i], [0,
    ↪  dt], args=(u_oa_15, p))[1]
78
79  yaw_rate_desired_4.append(desired_values(x_oa_fr_TV_4[i],
    ↪ u_oa_4))
80  yaw_rate_desired_8.append(desired_values(x_oa_fr_TV_8[i],
    ↪ u_oa_8))
81  yaw_rate_desired_15.append(desired_values(x_oa_fr_TV_15[i],
    ↪ u_oa_15))
```

⌞ Πατήστε εδώ για να επιστρέψετε στο κυρίως κείμενο.

Απόσπασμα Κώδικα Α.25: Ελιγμός αποφυγής εμποδίου με επιτάχυνση $a_{\text{long}} = 0.2g$ και αριστερό-δεξί & εμπρόσθιο-οπίσθιο TV

```
# u0=4,8 m/s
from vehiclemodels.utils.steering_constraints import
    ↪ steering_constraints
from vehiclemodels.utils.acceleration_constraints import
    ↪ acceleration_constraints

# Initialize arrays, variables & lists for the control loop
x_oa_lr_fr_TV_4 = x_oa_4.copy()
x_oa_lr_fr_TV_8 = x_oa_8.copy()
e_psidot_prev_4 = 0
e_psidot_prev_8 = 0
yaw_rate_desired_4 = []
yaw_rate_desired_8 = []

kp = 2
kd = 400

for i in range(len(time)-1):

  acc_4 = acceleration_constraints(x_oa_lr_fr_TV_4[i,3],  0.2
     ↪ * g, p.longitudinal)
  acc_8 = acceleration_constraints(x_oa_lr_fr_TV_8[i,3],  0.2
     ↪ * g, p.longitudinal)

  T_E_4 =  p.m * p.R_w * acc_4
  T_E_8 =  p.m * p.R_w * acc_8

  # Calculate errors between desired and actual yaw rate
  e_psidot_4 = desired_values(x_oa_lr_fr_TV_4[i], u_oa_4) -
     ↪ x_oa_lr_fr_TV_4[i,5]
  e_psidot_8 = desired_values(x_oa_lr_fr_TV_8[i], u_oa_8) -
     ↪ x_oa_lr_fr_TV_8[i,5]

  # Calculate control input feedbacks from yaw rate error
     ↪ using a PD controller
  ΔT_psidot_4 = kp * e_psidot_4 + kd * (e_psidot_4 -
     ↪ e_psidot_prev_4)/dt
  ΔT_psidot_8 = kp * e_psidot_8 + kd * (e_psidot_8 -
     ↪ e_psidot_prev_8)/dt

  # Apply saturation to ensure the total torque adjustment is
     ↪ within the desired range (-T_E/4 <= ΔT <= T_E/4)
  ΔT_4 = max(- T_E_4/4 , min(ΔT_psidot_4, T_E_4/4 ))
  ΔT_8 = max(- T_E_8/4 , min(ΔT_psidot_8, T_E_8/4 ))
```

```
35
36    eng_tsplit_4 = tune_T_se(e_psidot_4)
37    eng_tsplit_8 = tune_T_se(e_psidot_8)
38
39    # Set vehicle input parameters for each time step
40    v_delta_4 = -0.5 * np.cos((2*np.pi/4)*time[i] + np.pi/35)
41    v_delta_8 = -0.5 * np.cos((2*np.pi/4)*time[i] + np.pi/10.1)
42
43    # Apply torque vectoring by adding the corrective torque to
        ↪ each wheel
44    u_oa_4 = (v_delta_4, 0.2 * g, ▵T_4, eng_tsplit_4)
45    u_oa_8 = (v_delta_8, 0.2 * g, ▵T_8, eng_tsplit_8)
46
47    # Update the previous errors for the next iteration
48    e_psidot_prev_4 = e_psidot_4
49    e_psidot_prev_8 = e_psidot_8
50
51    # Simulate the multi-body kinematic model with the updated
        ↪ control input
52    x_oa_lr_fr_TV_4[i + 1] = odeint(func_MB, x_oa_lr_fr_TV_4[i],
        ↪  [0, dt], args=(u_oa_4, p))[1]
53    x_oa_lr_fr_TV_8[i + 1] = odeint(func_MB, x_oa_lr_fr_TV_8[i],
        ↪  [0, dt], args=(u_oa_8, p))[1]
54
55    yaw_rate_desired_4.append(desired_values(x_oa_lr_fr_TV_4[i],
        ↪  u_oa_4))
56    yaw_rate_desired_8.append(desired_values(x_oa_lr_fr_TV_8[i],
        ↪  u_oa_8))
57
58    #u0=15 m/s
59  from vehiclemodels.utils.steering_constraints import
      ↪ steering_constraints
60  from vehiclemodels.utils.acceleration_constraints import
      ↪ acceleration_constraints
61
62  # Initialize arrays, variables & lists for the control loop
63  x_oa_lr_fr_TV_15 = x_oa_15.copy()
64  e_psidot_prev = 0
65  t_split = []
66  yaw_rate_desired_15 = []
67
68  kp = 2
69  kd = 400
70
71  for i in range(len(time)-1):
72
73    acc = acceleration_constraints(x_oa_lr_fr_TV_15[i,3],  0.2 *
        ↪  g, p.longitudinal)
```

```
74
75  T_E =  p.m * p.R_w * acc
76
77  # Calculate errors between desired and actual yaw rate
78  e_psidot = desired_values(x_oa_lr_fr_TV_15[i], u_oa_15) -
       ↪ x_oa_lr_fr_TV_15[i,5]
79
80  # Calculate control input feedbacks from yaw rate error
       ↪ using a PD controller
81  ΔT_psidot = kp * e_psidot + kd * (e_psidot - e_psidot_prev)/
       ↪ dt
82
83  # Apply saturation to ensure the total torque adjustment is
       ↪ within the desired range (-T_E/4 <= ΔT <= T_E/4)
84  ΔT = max(- T_E / 4, min(ΔT_psidot, T_E / 4))
85
86  # Apply torque vectoring by adding the corrective torque to
       ↪ each wheel
87  v_delta = -0.5 * np.cos((2*np.pi/4)*time[i] + np.pi/6.13)
88
89  eng_tsplit = tune_T_se(e_psidot)
90  t_split.append(eng_tsplit)
91
92
93  u_oa_15 = (v_delta, 0.2 * g, 2*ΔT, eng_tsplit)
94
95  # Update the previous errors for the next iteration
96  e_psidot_prev = e_psidot
97
98  # Simulate the multi-body kinematic model with the updated
       ↪ control input
99  x_oa_lr_fr_TV_15[i + 1] = odeint(func_MB, x_oa_lr_fr_TV_15[i
       ↪ ], [0, dt], args=(u_oa_15, p))[1]
100
101 yaw_rate_desired_15.append(desired_values(x_oa_lr_fr_TV_15[i
       ↪ ], u_oa_15))
```

⌞ Πατήστε εδώ για να επιστρέψετε στο κυρίως κείμενο.

Βιβλιογραφία

[1] Matthias Althoff, Markus Koschi, and Stefanie Manzinger. "CommonRoad: Composable benchmarks for motion planning on roads". In: *Proc. of the IEEE Intelligent Vehicles Symposium*. 2017.

[2] R. W. Allen et al. *Vehicle dynamic stability and rollover*. Final Report Systems Technology, Inc., Hawthorne, CA. June 1992.

[3] MSC Software. "Adams/Tire help". In: (Apr. 2011).

[4] HB Pacejka. *Tyre and vehicle dynamics*. Butterworth Heinemann, 2002. ISBN: 0-7506-5141-5.

[5] Matthias Althoff and Silvia Magdici. "Set-Based Prediction of Traffic Participants on Arbitrary Road Networks". In: *IEEE Transactions on Intelligent Vehicles* PP (Oct. 2016), pp. 1–1. DOI: 10.1109/TIV.2016.2622920.

[6] Jonathan C. Wheals. "Torque Vectoring Driveline: SUV-based Demonstrator and Practical Actuation Technologies". In: *SAE Transactions* 114 (2005), pp. 631–648. ISSN: 0096736X, 25771531. URL: http://www.jstor.org/stable/44725096.

[7] Rizwan Latif et al. "Vehicle Modeling and Performance Evaluation Using Active Torque Distribution". In: vol. 1. Oct. 2013, p. 8. DOI: 10.4271/2014-01-0103.

[8] Jonathan C. Wheals et al. "Torque Vectoring AWD Driveline: Design, Simulation, Capabilities and Control". In: *SAE 2004 World Congress and Exhibition*. SAE International, Mar. 2004. DOI: https://doi.org/10.4271/2004-01-0863. URL: https://doi.org/10.4271/2004-01-0863.

[9] R. Rajamani. *Vehicle Dynamics and Control*. Mechanical Engineering Series. Springer US, 2011. ISBN: 9781461414339. URL: https://books.google.gr/books?id=cZJFDox4KuUC.