Technical University of Crete

School of Production Engineering and Management

Application of numerical approximation methods in control systems which describe movement of autonomous vehicles in lane free-roads

Εφαρμογή αριθμητικών μεθόδων σε συστήματα ελέγχου που περιγράφουν την κίνηση αυτόνομων οχημάτων σε δρόμους χωρίς λωρίδες

Created by: Tzitzikopoulos Nikolaos Marinos

Supervisor: Dr. Papamichail Ioannis

Committee:

Dr. Chalkiadakis Georgios

Dr. Doitsidis Lefteris

Chania, 2022

Acknowledgements

Words cannot express my gratitude towards my supervisor and chair of my committee, Dr. Papamichail Ioannis, as also my co-supervisor Dr. Theodosis Dionysis for their invaluable patience and feedback. I would also like to thank the rest of my committee, Dr. Chalkiadakis Georgios and Dr. Doitsidis Lefteris for attending my presentation and their comments towards the improvement of my thesis.

Lastly, I am also grateful to my friends and family for their moral support that kept me motivated through the whole process of my studies.

Abstract

Automobiles have changed people's everyday life and have become essential for the personal transportation of millions of people. Advancements in technology are growing and enhancing the driver's experience, from automatic headlights to automatic emergency braking and autonomous driving, nowadays. Autonomous driving on lane-free roads is a complex system where the vehicles have to be “connected” with each other and “cooperate” to perform their movement with safety. Usually, this kind of problems consists of non-linear or stiff differential equations which cannot be solved analytically, thus in this thesis we utilize numerical approximation methods to investigate a proposed system of cooperative autonomous vehicles in lane-free roads [6] to observe the simulation's results along with the system's control functions. However, such complex systems describing autonomous vehicles driving on lane-free roads tend to be a challenge for numerical approximation methods, where high order Runge-Kutta methods may not be applicable while low order Runge-Kutta methods may present numerical instability if the initial step size is not sufficiently small. To achieve our goals, we analyze the system's characteristics and utilize a variety of numerical approximation methods to observe the vehicle's behavior on lane-free roads, as also their results, and make comparisons between them and their errors. Furthermore, we utilize an adaptive step size control in order to maintain the numerical solutions inside a defined open set, as also have the advantage of increasing and decreasing the step size, depending on the behavior of the system at any given moment. Following, we will use a Lyapunov function of the system, which represents the energy developed between the vehicles as the step size evaluator at an adaptive numerical method, in contrast to the regular adaptive methods which utilize the vehicle's characteristics, their lateral and longitudinal positions, their velocities and their wheel orientations. Lastly, we will try to investigate certain repulsive potential functions of the system, bound to keep the integrity of the vehicles hoping for smoother and more desired trajectories developed.

Περίληψη

Τα αυτοκίνητα έχουν δημιουργήσει πολλές αλλαγές στην καθημερινή ζωή των ανθρώπων και έχουν γίνει απαραίτητα για την προσωπική μεταφορά εκατομμυρίων. Οι εξελίξεις στην τεχνολογία αυξάνονται και βελτιώνουν την οδηγική εμπειρία, από τους αυτόματους προβολείς, στο αυτόματο φρενάρισμα έκτακτης ανάγκης, πλέον μέχρι και την αυτόνομη οδήγηση. Η αυτόνομη οδήγηση σε δρόμους χωρίς λωρίδες είναι ένα σύνθετο σύστημα όπου τα οχήματα πρέπει να είναι «συνδεδεμένα» μεταξύ τους και να «συνεργάζονται» έτσι ώστε να εκτελούν την κίνησή τους με ασφάλεια. Συνήθως, τέτοιου είδους προβλήματα αποτελούνται από μη γραμμικές ή δύσκαμπτες διαφορικές εξισώσεις οι οποίες δεν μπορούν να λυθούν αναλυτικά, επομένως σε αυτή τη διπλωματική χρησιμοποιούμε μεθόδους αριθμητικής ανάλυσης για τη διερεύνηση ενός προτεινόμενου συστήματος συνεργαζόμενων αυτόνομων οχημάτων σε δρόμους χωρίς λωρίδες [6], με στόχο να παρατηρήσουμε τα

αποτελέσματα των προσομοιώσεων μαζί με τις λειτουργίες ελέγχου του συστήματος. Ωστόσο, τέτοια συστήματα που περιγράφουν αυτόνομα οχήματα που κινούνται σε δρόμους χωρίς λωρίδες τείνουν να αποτελούν πρόκληση για μεθόδους αριθμητικής ανάλυσης, όπου για παράδειγμα Runge-Kutta μέθοδοι υψηλής τάξης ενδέχεται να μην είναι εφαρμόσιμοι, ενώ Runge-Kutta μέθοδοι χαμηλής τάξης ενδέχεται να παρουσιάζουν αριθμητική αστάθεια εάν το αρχικό μέγεθος του βήματος δεν είναι αρκετά μικρό. Για να επιτύχουμε τους στόχους μας, αναλύουμε τα χαρακτηριστικά του συστήματος και χρησιμοποιούμε ένα εύρος μεθόδων αριθμητικής ανάλυσης για να παρατηρήσουμε τη συμπεριφορά των οχημάτων σε δρόμους χωρίς λωρίδες, καθώς και τα αποτελέσματά τους και να κάνουμε συγκρίσεις μεταξύ των αποτελεσμάτων και των σφαλμάτων των μεθόδων. Επιπλέον, χρησιμοποιούμε μεθόδους μεταβαλλόμενου μεγέθους του βήματος για να διατηρήσουμε τις αριθμητικές λύσεις μέσα σε ένα καθορισμένο ανοιχτό σύνολο τιμών, καθώς με αυτή την πρακτική έχουμε επίσης το πλεονέκτημα της αύξησης και της μείωσης του μεγέθους βήματος, ανάλογα με τη συμπεριφορά του συστήματος σε κάθε δεδομένη στιγμή. Στη συνέχεια, θα χρησιμοποιήσουμε μια Lyapunov συνάρτηση του συστήματος, η οποία αντιπροσωπεύει την ενέργεια που αναπτύσσεται μεταξύ των οχημάτων ως τον καθοριστικό παράγοντα του μεγέθους του βήματος σε μια αριθμητική μέθοδο μεταβαλλόμενου βήματος, σε αντίθεση με τις κανονικές «προσαρμοστικές» μεθόδους που χρησιμοποιούν τα χαρακτηριστικά του οχήματος, τις θέσεις των οχηματων στο δρόμο, τις ταχύτητες και τον προσανατολισμό των τροχών τους. Τέλος, θα προσπαθήσουμε να διερευνήσουμε ορισμένες απωστικές συναρτήσεις του συστήματος, η λειτουργία των οποίων είναι να διατηρήσουν την ακεραιότητα των οχημάτων, με στόχο ομαλότερες και πιο επιθυμητές τροχιές.

Table of Contents

1. Introduction

Transportation has been a problem for humans since ancient times when humans lived as nomads and had to move around searching for better places to live. As the years passed, humanity tried to domesticate animals for labor and transportation. A milestone for human transportation is the invention of the wheel, dating back to around 4000 BC, creating the first wheeled vehicles. Since then many ways for transportation have been created, but nowadays the most popular one is the automobile since it provides the ability to move flexibly from place to place and far-reaching destinations [1, p. 47]. However, traffic collisions are the largest cause of injury-related deaths worldwide [1, p. 20]. Human factor plays a great part in vehicular movement and eliminating it could improve the performance of transportation systems, their safety, reduce congestion and traffic accidents, and improve traffic flow on highways.

Autonomous vehicles date back to the beginning of the 20^{th} century [2] when scientists started to dream of vehicles driving autonomously along highways, in order to increase the safety, efficiency, and convenience of the transportation system. However, communicative and cooperative systems have not been introduced to vehicles yet. Autonomous vehicles right now rely on radar and vision systems, while lidar, sonar, and camera systems have also been used. Another great advancement in technology is non-line of sight propagation [18]. Typical vision systems image objects that are in line of sight, however advanced measurement systems, such as femtosecond time-resolved detectors, acoustic systems, etc., are able to detect and reconstruct objects hidden behind obstacles [3]. This technology could immensely help autonomous vehicles prevent accidents, along with making safer decisions for their movement.

As automobiles came into widespread use, head-on collisions became more common and parallel lanes were introduced on roads, to separate traffic going in different directions and increase safety. By the increasing use of vehicles multiple lanes were introduced. That made driving a simpler task, considering drivers have only to acknowledge the width and speed of their vehicle, as also monitor the distance and speed of the front vehicle. However, lanes added an extra risky operation which is that of lane changing. When a driver wishes to properly change lane, he has not only to monitor the front vehicle but also seek an available gap, by observing the vehicle movement at the next lane and estimating its speed and acceleration. This task is risky, considering you have to check your mirrors and also check your blind spot by looking over your shoulder, a small amount of time at which your sight is not onto the front vehicle. In fact, for 10% of all accidents, lane changes were responsible [4]. Taking into consideration that autonomous vehicles will not be commanded by a driver, hence shorter inter-vehicle distances could be utilized. Thus a lane-free concept would highly impact road-traffic and congestion, through the increase of road capacity. In addition,

cooperation and communication between vehicles will provide them with the necessary knowledge about their distance from the boundaries of the road and the relative distance from adjacent vehicles. This is an innovative concept at which as vehicles will utilize the whole width of the road, they will not collide with other each other and neither with the boundary of the road. Taking into account, cooperation and communication between vehicles, a "nudging" effect can take place [5]. By the term "nudging", we refer to a virtual force that vehicles apply to the vehicles in front of them without jeopardizing speed limits or traffic safety. With the proposed nudge effect, an increased flow and road capacity can be achieved see [27].

The movement of autonomous vehicles in lane-free roads forms a demanding and complex problem. All vehicles have to be connected and communicate with each other through their sensors, in order to complete their movements with safety. Such a system is usually described by a great number of non-linear differential equations, which are impossible to be solved analytically. Thus, numerical methods are essential for investigating such systems and gaining an approximation of the real solution. Still, numerical methods find challenging to solve such ordinary differential equations systems, because systems describing the movement of vehicles in lane-free roads tend to be:

i. Non-linear
ii. Stiff
iii. The state space is defined by an open set

Hence, numerical methods for ordinary differential equations, such as the 4^{th} order Runge-Kutta may not be immediately applicable, while lower-order methods, such as Euler and Heun may present numerical instability if the initial step size is not sufficiently small. Moreover, the numerical solution may attain values outside the open set defining the system due to numerical errors. For such cases, an adaptive step size technique is preferred. The main advantage of this methodology is that of adjusting the step size, depending on the behavior of the numerical solution gained from each iteration. This way, the numerical solution approximation speed can be enhanced, along with its precision. The methods mentioned above will be used to approximate the numerical solution of the problem of the movement of autonomous vehicles. Subsequently, simulations will be made and the following will be reviewed:

i. The movement behavior of vehicles in lane-free roads, and
ii. The trends of the numerical solutions and their errors, obtained from the various numerical approximation methods used.

In this thesis, we are numerically approximating the solution of a vehicular integrated infrastructure, in which vehicles communicate and cooperate with each other, increasing the capacity of the road through maintaining shorter inter-vehicle distances and adapting a lane-free concept [6]. We are going to utilize both stable and adaptive numerical approximation methods, in order to evaluate and compare their solutions with each other. Furthermore, a

new direction of adaptive technique is introduced. In contrast of the regular adaptive methods, we intend to utilize a Lyapunov function that represents the energy of the system, thus we do not depend on the vehicles parameters for the evaluation of the step taken, rather than from the whole energy produced by the vehicles, their inter – vehicle distances, the distances from the boundary of the road, their velocities and their wheels orientation. The thesis is outlined as follows, in chapter 2 is stated the necessary introduction to Ordinary Differential Equations and Numerical Analysis, as in chapter 3, a detailed review of the Numerical Approximation Methods utilized for the solution of a set of ODEs is given. Without these two chapters, it would be impossible to analyze the Cooperative Adaptive Cruise Control (CACC) given and numerically investigate the solution of its set of ODEs. Thus, chapters 4 and 5 are created regarding the 2D CACC and its numerical approximation. Finally, chapters 6 and 7 focus on comparing the results gained by various numerical methods and investigating certain functions of the system.

2. Preliminaries for Ordinary Differential Equations and Numerical Analysis

Differential equations are equations that involve one or more derivatives of a function. This kind of equation could involve an independent variable, a dependent variable, and one or more derivatives of the dependent variable

$$a_0(t)y(t) + a_1(t)y'(t) + a_2(t)y''(t) + \ldots + a_n(t)y^{(n)}(t) + b(t) = 0$$

Ordinary differential equations involve only one independent variable, whilst, equations with two or more independent variables are called partial differential equations [7]. Linear differential equations play an important role since they regularly appear in physical phenomena and can be solved analytically. However, the behavior of complex systems is usually described by nonlinear differential equations which are hard or even impossible to solve explicitly [8]. Linear equations have a constant slope, hence forming a line. In contrast, nonlinear equations have the opposite characteristics of the linear ones; their slope may vary between points, resulting to a shape different than a line.

For this reason, we utilize numerical methods gaining a quick, but also acceptable approximation as the solution for the system. The numerical solutions gained, involve two types of errors, round-off errors and truncation (or discretization) errors. Round-off errors are the result of the inability of computers to present all real ($\mathbb{R}$) numbers and their precision. Considering that most numerical methods when solving a system of ODEs calculate approximate solutions step by step, hence with the approximation comes an error for each step taken. This error, which also depends on the different equations utilized from each numerical approximation method, is called truncation (or discretization) error [9].

The error added with every step is called local error, though the propagated error is the added error due to the previous approximation. Hence the overall difference of the exact and the approximated solution is the sum of both errors and is called global truncation error [10, p. 56],[9, p. 98][22]. The main difference between numerical approximation methods is the procedure from which the slope is estimated. All of the methods belong to the Runge-Kutta family. You can calculate the error exactly, by comparing the approximation with the analytical solution [21],[23]. However, in many cases, as with our system, this cannot be achieved due to the absence of the analytical solution. Hence, we can only obtain an estimation of the errors.

Another difficulty regarding numerical methods is that of stiff equations. Although there is no precise definition of stiffness, a stiff equation is considered to be numerically unstable for relatively high step sizes and prone to even small changes at its initial conditions. Stiff problems pose difficulties to solving by standard explicit methods, whereas some implicit methods seem to perform better. However, implicit methods take more time to approximate

the solution, since they require the solution of a non-linear algebraic equation system. Such problems may consist of both rapidly and slowly changing parts, its step size decreases by stability requirements due to having some eigenvalues λ_i negative and large in magnitude or for complex eigenvalues with negative real parts [8],[15, p. $254 - 256$].

Stability is another important characteristic that should be taken into account, concerning numerical approximation methods. A numerical solution is considered to be unstable if the error grows exponentially for a problem with a bounded solution [10, p. 559]. Stability depends on three factors: the given differential equations, the step size and the numerical method utilized. Furthermore, a numerical method converges if its error tend to zero when the step size tend to zero. By decreasing the step size the iterations on the other hand increase, hence the computational costs increase. For a method to be considered convergent, it is required to converge on all problems for all reasonable initial conditions [11, p. 137].

3. Numerical methods for solving ordinary differential equations

Two main types of numerical methods exist, the explicit and the implicit methods. The explicit methods calculate the numerical solution of the ODEs, subject to the exactly previous approximation of the numerical solution and the values of the equation. It could have the form of [12].

$$y_i = F(t_{i-1}, t_i, y_{i-1})$$

The implicit methods demand either the solution of a non-linear system of algebraic equations or the solution of the algebraic equation with a root-finding method such as Newton Raphson. It could have the form of:

$$G(t_{i-1}, t_i, y_{i-1}, y_i) = 0$$

The simplest of all numerical methods is the forward or explicit Euler method (1768), which is produced from the first two terms of the Taylors sequence [9, p. 708],

$$y_{i+1} = y_i + (t_i - t_{i-1}) f(t_i, y_i)$$

The term $(t_i - t_{i-1})$ will be called time step and be denoted by

$$h \equiv t_i - t_{i-1}$$

It is a first-order Runge-Kutta method used to solve initial value problems. This method uses a constant step size to compute step by step, approximations for $y_1, y_2,$ etc from an initial value $y(0) = y_0$, basically constructing the tangent of the slope. The local error (error per step) of Euler's method is proportional to the square of the step size h^2 and the global error (error given at any time) is proportional to the step size h [9, p. 710-712]. We could use a table to display the factors of every Runge-Kutta method. This table is called the Butcher tableau and for the explicit Euler method is as shown below [12, p. 135],[25]:

0	
	1

Another simple method, often known as improved Euler method, is Heun method [9, p. 720],

$$k_1 = f(t_i, y_i)$$

$$k_2 = f(t_i + h, y_i + k_1 h)$$

$$y_{(i+1)} = y_i + h/2\,(k_1 + k_2)$$

It is a second order Runge-Kutta method which derives from transforming the trapezoidal method,

$$y_{i+1} = y_i + h/2\left(f(t_i, y_i) + f(t_{i+1}, y_{i+1})\right)$$

Although this method also uses a constant step size, it can perform better than Euler since it uses one more approximation to calculate the numerical solution. Heun method has the following Butcher tableau [12, p. 135],[25]:

$$\begin{array}{c|cc} 0 & & \\ 1 & 1 & \\ \hline & 1/2 & 1/2 \end{array}$$

Example 1: Consider the following ordinary differential equation,

$$\dot{y} = -y + \cos(t) - t\sin(t)$$

whose analytical solution with initial condition $y(0) = 2.0$ is given analytically from the equation, $y = 2e^{-t} + \frac{1}{2}t\cos(t) + \frac{1}{2}\sin(t) - \frac{1}{2}t\sin(t)$. The table below shows the approximation of $y(3)$, for both Euler and Heun method, with step size $h = 0.1$.

Time step	Real Value	Euler Method	Heun Method
0	*2.0*	*2.0*	*2*
0.1	*1.9044*	*1.9*	*1.9043*
0.2	*1.8149*	*1.8085*	*1.8147*
...	...	...	...
1	*1.0059*	*1.0202*	*1.0043*
1.1	*0.8707*	*0.8881*	*0.8690*
...	...	...	...
2	*-0.6001*	*-0.5881*	*-0.6001*
2.1	*-0.7599*	*-0.7528*	*-0.7595*
...	...	...	...
2.9	*-1.5251*	*-1.5748*	*-1.5214*
3	*-1.5265*	*-1.5838*	*-1.5224*

Notice first that Heun provides a better approximation than the Euler method. Moreover, we can observe all the values gained at Figure 1,

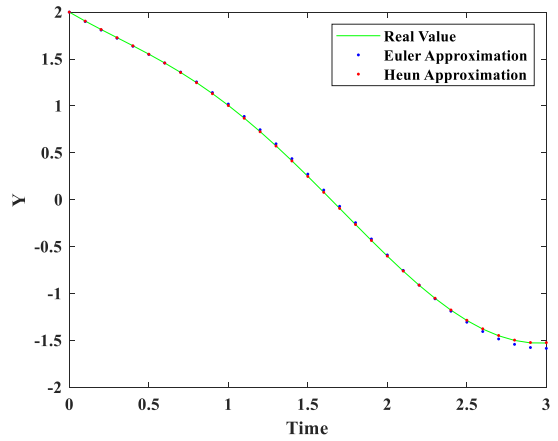


Figure 1: Comparisons between the Real Value and the Approximations

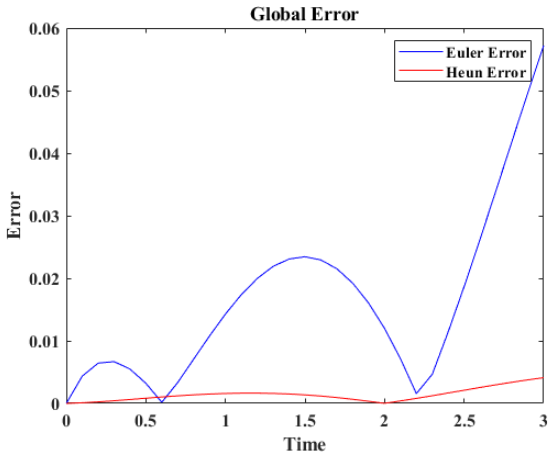


Figure 2: Error produced by each approximation

The example above shows that for simple problems, a small step size can produce rather good approximations. But when the problem is more complex and/or the time step needs to be greater, these kind of methods could fail producing a fine approximation.

Even though Euler and Heun methods seem to produce a fine approximation of the analytical solution both underperform when a stiff problem is introduced. To gain a good approximation, the step size has to be decreased, increasing significantly execution time of the algorithm. Considering the nature of our problem, execution time plays an essential role in driving us to methods using less iteration but also maintaining a good approximation of the solution [12, p. 164]. Such methods are called automatic or adaptive step size methods. The idea of adaptive techniques is such to evaluate an approximation made comparing it with a tolerance to decide the following step size [12],[23], aiming to enhance the precision of the method. There exist many methods to approximate as also to evaluate the approximation gained.

Example 2: Consider the following ordinary differential equation,

$$\dot{y}(t) = -30y(t), y(0) = 1$$

Here, both Euler and Heun fail to produce a fine approximation of the real solution for a step size of $h = 0.1$ and even for $h = 0.05$ over the period of 1 second.

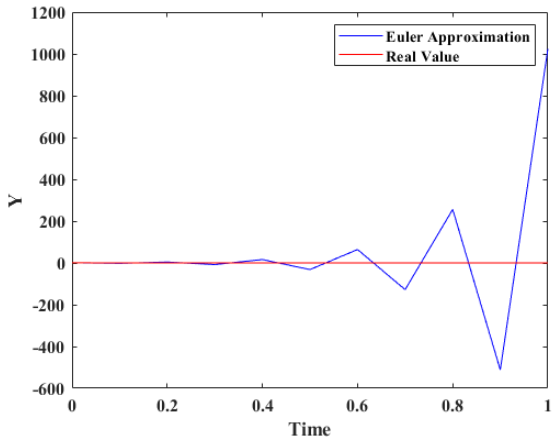


Figure 3: Approximation of Euler Method with step size of $h = 0.1$

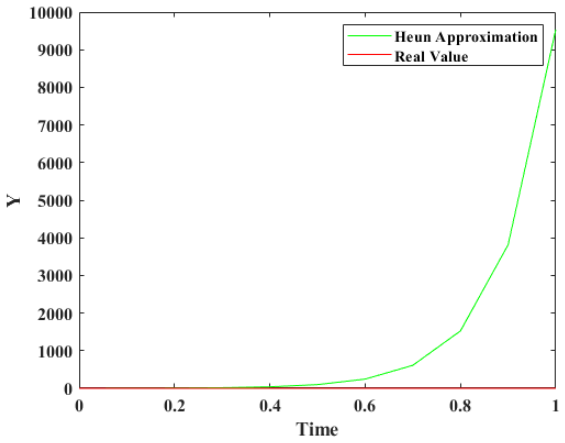


Figure 4: Approximation of Heun Method with step size of $h = 0.1$

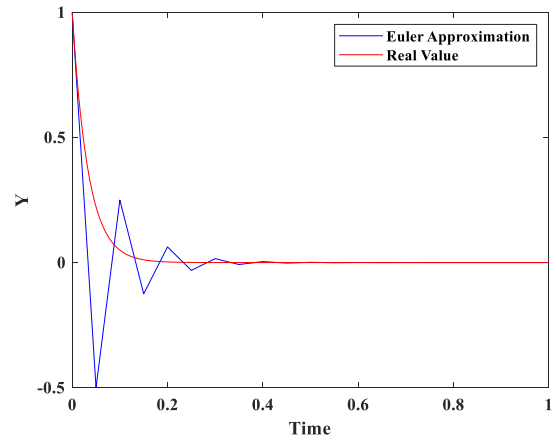


Figure 5: Approximation of Euler Method with step size of $h = 0.05$

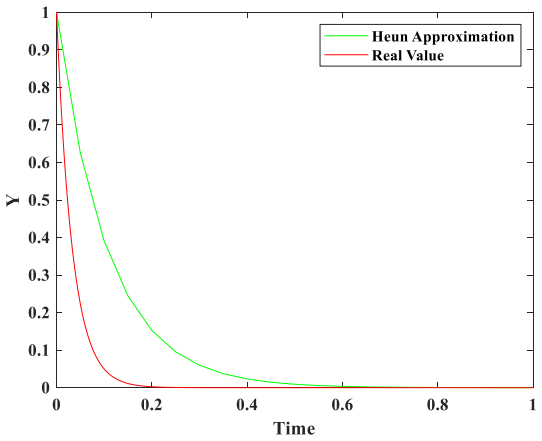


Figure 6: Approximation of Heun Method with step size of $h = 0.05$

On the other hand, an Adaptive technique with an initial step size $h = 0.1$, is able to adapt its step size during each iteration and produce a far more accurate approximation.

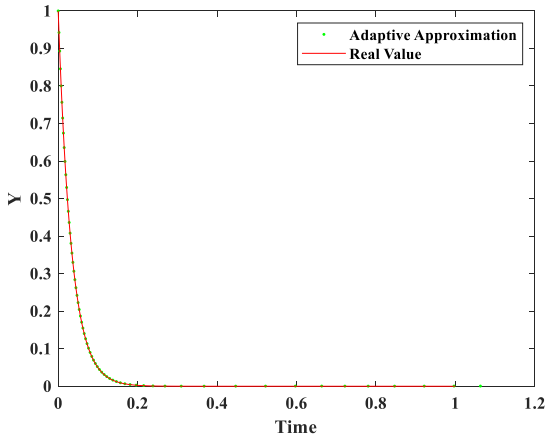


Figure 7: Approximation of Adaptive Method

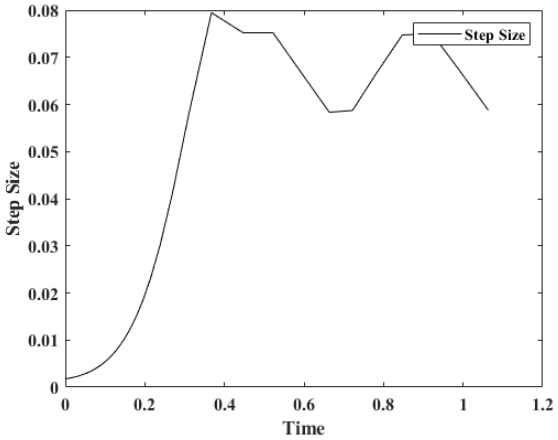
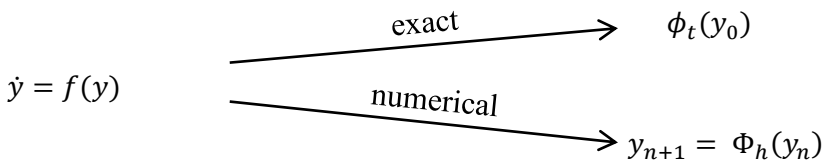


Figure 8: Step Size of Adaptive Method

As already mentioned, when using an approximation method instead of the analytical procedure, an error in the results is introduced, called truncation error [10, p. 89]. Also, by using a computer for the approximations round-off errors have to be taken into account. These errors arise due to the inability of the computers to represent some real numbers. Generally, in approximation methods we distinguish the error in the *local* error and the *global* error [22]. Local truncation error is difference between the approximation and the real value at every single step. The propagated truncation error is the result of performing iterations with values that have been approximated. Lastly, global truncation error is the sum of the local and the propagated truncation error and it can be found at the last iteration performed [13, p. 337].



Local Error: $y_i - \phi_i(y_0)$

Global Error: $y_n - \phi_n(y_0)$

In most applications though, the exact solution is not available, which is also the reason we use numerical approximation methods, hence we have to depend on estimations of the error. The error estimation also derives from the Taylor series and consists from the remainder parts that are left out of the method used. Euler method which uses the first two parts for the approximation has an error of the remainder parts as follows [10, p. $103 - 106$]:

$$y_{i+1} = y_i + f(t_i, y_i)h + \frac{f'(t_i, y_i)}{2!}h^2 + \cdots + \frac{f^{(n-1)}(t_i, y_i)}{n!}h^{(n)} + O(h^{n+1})$$

where $O(h^{n+1})$ specifies that the local truncation error is proportional to the step size raise to the power $(n+1)$, where n represents the order of the method used. By leaving parts of the Taylor series off of our numerical approximation, it simulates as ignoring parts of the exact solution, hence the approximation error. Thus for Euler method we have the following error follows:

$$E_t = \frac{f'(t_i, y_i)}{2!}h^2 + \cdots + \frac{f^{(n-1)}(t_i, y_i)}{n!}h^{(n)} + O(h^{n+1})$$

,and for a relatively small step size h, all terms except the first can be ignored:

$$E_t = \frac{f'(t_i, y_i)}{2!}h^2 = O(h^2)$$

Therefore, Euler method has an approximate local truncation error of $O(h^2)$. This confirms the general local truncation error equation $O(h^{n+1})$, since Euler method is a first order Runge-Kutta method [10, p. $557 - 559$].

Since Heun method is a second order Runge-Kutta method, we have to predict that the approximate local truncation error will be $O(h^3)$. Considering that Heun method is made from transforming the Trapezoidal rule, they have the same approximate local truncation error. The Trapezoidal rule has a local truncation error of [10, p. $562 - 566$]:

$$E_t = -\frac{f''(\xi)}{12}h^3 = O(h^3)$$

where ξ is between t_i and t_{i+1}.

Richardson Extrapolation

This is a simple error based adaptive technique, which utilizes a Runge-Kutta method to compute two numerical approximations y_1 and y_2. The first approximation is found with a step size h, while for the second one the step size is halved, $h/2$. That way an estimate of the error is computed as the difference between those two approximations. Considering for the second approximation where the step size is half, a slightly better approximation is expected. Since the exact solution most of the times is not available, the error is estimated by the difference between the two approximations provided by the Richardson extrapolation. The error is computed by the following equation [12, p. 164 – 165]:

$$err = \sqrt{\sum_{i=1}^{n} \left(y_{1,i} - y_{2,i}\right)^2}$$

Following, if the error is below a designated tolerance we set the new step size as:

$$h_{new} = 2h_{old}$$

or if it is over the designated tolerance:

$$h_{new} = \frac{h_{old}}{2}$$

Embedded Runge-Kutta Methods

In the embedded Runge-Kutta scheme, rather than using one method, two Runge-Kutta methods are utilized, one of which with order p and the other with order $p-1$. The simplest of all is Heun-Euler method, which utilizes the second order Heun method, defined as y_1 and the first order Euler method, defined as $\hat{y}_1$. By using two consecutive order methods, with the same step size, less computational costs derive, since Euler method is included in Heun's method. Another advantage of this technique is that through the difference between the two approximations obtained, an estimate of the local error is produced. Hence, as a measure of error we take:

$$err = \sqrt{\frac{1}{n}\sum_{i=1}^{n} \left(\frac{y_{1,i} - \hat{y}_{1,i}}{sc_i}\right)^2}$$

where $\mathrm{sc_i} = \mathrm{Atol_i} + \max\left(|\mathrm{y_{0i}}|, |\mathrm{y_{1i}}|\right) \cdot \mathrm{Rtol_i}$ and satisfy $|\mathrm{y_{1i}} - \hat{\mathrm{y}}_{1\mathrm{i}}| \leq \mathrm{sc_i}$. Relative errors are considered for $\mathrm{Atol_i}$, absolute errors for $\mathrm{Rtol_i}$ and both are prescribed as the desired tolerances by the user. Here n is the number of ordinary differential equations that the system approached contains [12, p. 165 – 167].

All techniques adapt the step size in an effort to keep the local error within an appointed tolerance. This error computational method is used in order to scale the error in such way that if it is less than 1, the step made is considered *accepted*, and if greater than 1, the step made is considered *rejected*. If the step is accepted the new step size will increase, otherwise it will decrease.
In the problem described in the following chapter, the new step size is considered *accepted* if $\mathrm{err} \leq 1$ and the code advances to the next step using the new step size $\mathrm{h_{new}}$. On the other hand, the new step size is considered *rejected* not only if $\mathrm{err} > 1$, but also if the parameters gain values outside of the open Ω set (the set is described at the following chapter). If $\mathrm{err} > 1$ and the values gained respect the Ω set, we get the new step size as stated below, else we set the step size as half of the previous and we perform the iteration again.

Computing the new Step-Size

Method 1: Through the estimated error the new step size can be determined. Generally a safe way to calculate the new step size is stated below [12, p. 167 – 168]:

$$\mathrm{h_{new}} = \mathrm{h}\,\mathrm{min}\left(\mathrm{facmax}, \mathrm{max}\left(\mathrm{fac}\sqrt{\frac{1}{\mathrm{err}}}, facmin\right)\right)$$

This function has a minimum factor *facmin*, which prevents from fast decreases of the step size as also a maximum factor *facmax*, which prevents from fast increases. These factors are much needed, considering that the bigger step size is prone to errors and our step size is also adaptive, hence our propagated error may grow rapidly.

Method 2: Another way of gaining the new step size is the following,

$$\mathrm{h_{new}} = \mathrm{h}\,\mathrm{min}\left(\mathrm{P}, \mathrm{fac}\sqrt{\frac{1}{\mathrm{err}}}\right)$$

which does not allow fast increase or decrease of the step size. Big step increases are prevented by P, the maximum step size multiplier, while fac is a safety factor, ensuring that the following error will be acceptable.

Method 3: A rather simple technique is that of doubling and halving the step size in order to obtain the new one. By using this technique the change of the step size is drastic, fast increases and decreases at each step, hence fluctuations will be observed. This way, the computational costs for gaining the approximation are increased. The new step size is doubled when it is considered *accepted*,

$$\mathrm{h_{new}} = 2\mathrm{h}$$

and halved when it is considered *rejected*,

$$\mathrm{h}_{\mathrm{new}} = \frac{1}{2}h$$

In Chapter 5 we will study and compare the above methods and we will provide a new adaptive method designed suitable for the problem at hand.

4. 2D Cooperative Cruise Control

New technologies are being introduced in our lives every single day. Vehicle automation and communication between vehicles are one with great impact since it will improve the performance of transportation systems, their safety, reduce congestion and traffic accidents, and improve traffic flow on highways.

Adaptive Cruise Control (ACC) [6], [30] systems are an evolution of Cruise Control systems, which maintained the speed of the vehicle at a certain desired value. ACC is able to automatically adjust the vehicle's speed to maintain a certain distance from a front vehicle or to maintain a desired speed. These certain technologies require a lot of sensors and information to be evaluated for a decision to be taken. Cooperative Adaptive Cruise Control (CACC) [6], [29] systems are wirelessly connecting vehicles, enabling them to exchange valuable information, therefore the decisions will be taken faster with less complicated calculations with fewer time needed. Thus shorter inter-vehicle distances will be maintained, the capacity of the roads and the safety will increase.

In this chapter we consider the two-dimensional movement of autonomous vehicles in lane free roads [5], [6], [28], [31]. We consider n identical vehicles in a lane free road, whose movement is described by the following ODEs:

$$\begin{aligned} \dot{x}_i &= v_i \cos\theta_i \\ \dot{y}_i &= v_i \sin\theta_i \\ \dot{\theta}_i &= \sigma^{-1} v_i \tan\delta_i \\ \dot{v}_i &= F_i \end{aligned} \tag{4.1}$$

for $i = 1, \dots, n$. Here, (x_i, y_i) are the longitudinal and lateral position of the i-th vehicle respectively, with $x_i \in \mathbb{R}$ and $y_i \in (-a, a)$, while we place its reference point at the midpoint of the rear axle of the vehicle; v_i is the speed of the i-th vehicle at the point (x_i, y_i); $\theta_i \in \left(-\frac{\pi}{2}, \frac{\pi}{2}\right)$ is the heading angle of the i-th vehicle and δ_i is the steering angle of the front wheels. Last but not least, F_i is the acceleration of the i-th vehicle. This model is known as the bicycle kinematic model [6], [32].

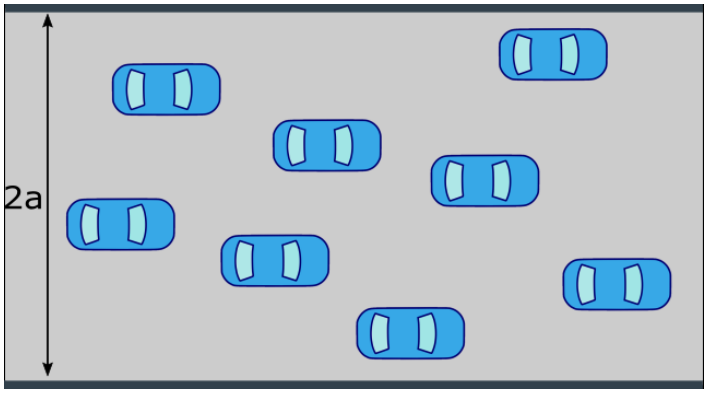


Figure 9: Lane-free road of width 2a > 0

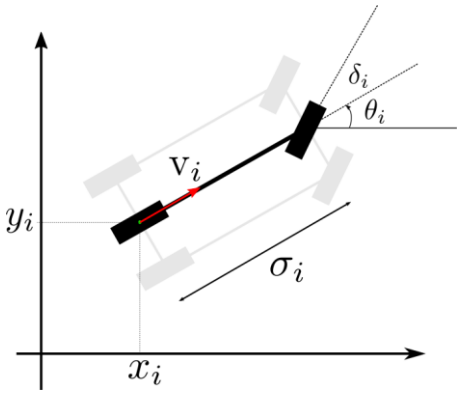


Figure 10: Each vehicle is modeled by the bicycle kinematic model

In order to make the analysis less complex, we define

$$u_i = \sigma^{-1} v_i \tan(\delta_i), i = 1, \dots, n \tag{4.2}$$

Hence, the model can be written like the following simpler form

$$\begin{aligned} \dot{x}_i &= v_i \cos(\theta_i) \\ \dot{y}_i &= v_i \sin(\theta_i) \\ \dot{\theta}_i &= u_i \\ \dot{v}_i &= F_i \end{aligned} \tag{4.3}$$

for $i = 1, 2, \dots, n$, where F and u are the control inputs. Considering that communication between vehicles may change over time, the control laws adapted from this methodology are decentralized and only depend on local sensing capabilities. The controllers are decentralized in such way that relies on the full state of the vehicle i and also the state of the vehicles that are within its sensing radius. The distance between vehicles is defined by:

$$d = \sqrt{(x_i - x_j)^2 + p(y_i - y_j)^2} \tag{4.4}$$

For $p = 1$, we obtain the Euclidean distance while for $p > 1$ we have an "elliptical" metric which will allow to approximate more accurately the dimensions of a vehicle. This elliptical metric allows more vehicles to be placed along the width of the road while maintaining a certain safety distance between them.

Furthermore we define the set

$$S := \{\mathbb{R}^n \times (-a, a)^n \times (-\varphi, \varphi)^n \times (0, v_{max})^n\} \tag{4.5}$$

, and also follow the notation

$$w = (x_1, \dots, x_n, y_1, \dots, y_n, \theta_1, \dots, \theta_n, v_1, \dots, v_n)' \in \mathbb{R}^{4n} \tag{4.6}$$

The set S represents all the possible values the parameters of all n vehicles of the system can obtain. First of all, each vehicle has to stay within the road boundaries $(x_i, y_i) \in \mathbb{R} \times (-a, a)$ for $i = 1, \dots, n$. All vehicles operate on a lane-free road with speed limit $v_{max} > 0$, as also they are not allowed to move backwards at any given moment. For the given constant $\varphi \in \left(0, \pi/2\right)$, the vehicles should not be able to turn perpendicular to the road, hence it should hold that $\theta_i \in (-\varphi, \varphi)$. This constant performs as an orientation safety constraint, considering vehicles can achieve high speed, φ should restrict the steering angle values close to zero. Probably the most important property which is not constrained by the set S, is the collision avoidance between vehicles. Hence, a safety distance factor $L > 0$ is defined, where all distances between references points concerning any pair of vehicles should respect.

Thus the state-space of the n vehicles that operate on a lane-free road are described by an open set $\Omega \subset \mathbb{R}^{4n}$:

$$\Omega := \{w \in S : \ d_{i,j} > L, i, j = 1, \dots, n, j \neq i\} \tag{4.7}$$

To sum up, below are stated the objectives the decentralized feedback laws should follow**:**

i. all vehicles operating should not collide with each other nor with the boundary of the road,

ii. their velocities always be positive and remain below the given speed limit, as also converge to a certain speed set-point,

iii. the orientation of all vehicles always remain bounded by the given value $\varphi \in \left(0, \frac{\pi}{2}\right)$ and converge to 0,

iv. their accelerations, angular and lateral speeds tend to zero.

Considering all of the above, for every initial condition $w(0) \in \Omega$, we obtain a unique solution $w(t) \in \Omega$ for all $t \geq 0$ under the effect of all feedback law u_i and F_i for $i = 1, \dots, n$, from the closed loop system. Furthermore, all of the objectives stated above should be satisfied, for every initial condition $w(0) \in \Omega$, as well for their solutions $w(t) \in \Omega$.

Following, the decentralized control system has to be determined, as well as the constant parameters utilized by the model [6]. As already stated, the decentralized system shall

prevent collision between vehicles and the boundary of the road. Thus, repulsive potential functions are utilized, in such a manner that the repulsion force between vehicles grows when the individual distance is decreasing, and the repulsion tends to zero when the vehicles are distant. Considering that, functions $V: (L, +\infty) \to \mathbb{R}_+$ and $U = (-a, a) \to \mathbb{R}_+$ are $\mathbb{C}^2$ functions:

$$V(d) = \begin{cases} q\dfrac{(\lambda - d)^3}{d - L}, & L < d \leq \lambda \\ 0, & d > \lambda \end{cases} \tag{4.8}$$

$$U(y) = \begin{cases} \left(\dfrac{1}{a^2 - y^2} - \dfrac{c}{a^2}\right)^4, & -a < y < -\dfrac{a\sqrt{c-1}}{\sqrt{c}} \; and \; \dfrac{a\sqrt{c-1}}{\sqrt{c}} < y < a \\ 0, & -\dfrac{a\sqrt{c-1}}{\sqrt{c}} \leq y \leq \dfrac{a\sqrt{c-1}}{\sqrt{c}} \end{cases} \tag{4.9}$$

and satisfy,

$\lim_{d \to L^+}\big(V(d)\big) = +\infty$ and $V(d) = 0$, for all $d \geq \lambda$. $\lim_{y \to (-a)^+}\big(U(y)\big) = +\infty$, $\lim_{y \to a^-}\big(U(y)\big) = +\infty$, and $U(0) = 0$. The potential functions V and U are designed in a way to prevents inter vehicle collisions and collisions with the boundary of the road respectively.

Here $\lambda > L$ are constants which correspond to a large and a small ellipse around each vehicle, respectively. This ellipse is defined as following:

$$\begin{aligned} x^2 + py^2 &= L^2 \\ x^2 + py^2 &= \lambda^2 \end{aligned} \tag{4.10}$$

By appropriately selecting $p \geq 1$, the above functions (4.10) create ellipses around every vehicle as desired, and we can also determine their eccentricity $e = \sqrt{1 - \frac{1}{p}}$. The two concentric ellipses are considered to have semi-major axes of L and λ and semi-minor axes of $\frac{L}{\sqrt{p}}$ and $\frac{\lambda}{\sqrt{p}}$, respectively. The ellipses are determined in such way to maximize the road capacity as also prevent the vehicles to come close together and with the boundary of the road. The number of vehicles that can sit side-by-side depends on the size of the road $2a > 0$, the safety distance L, as well as the weight p. The formula which calculates this number N is the following:

$$N = \frac{2a\sqrt{p}}{L}$$

If there are no vehicles inside the larger ellipse and nor is the vehicle referenced near the boundary of the road, there is no repulsive force. By contrast, when the vehicle gets closer to another vehicle, the repulsive force grows, tending to infinity while the inner vehicle distance tends to L. The selection of both constants is equally important, since L is the safety distance and no vehicles are allowed to come within this "safety" ellipsoid, and λ creates the ellipsoid from which the vehicles gain their needed information. Thus, if the λ is selected large, the measurement area around the vehicles is increased, as also the inter vehicle distances may be affected.

Considering the repulsive potential functions, we have to appropriately select the rest of their constants. The constant q is responsible for the magnitude of the acceleration F_i, as also the repulsive force taking action between the vehicles V. For instance, by choosing small values of q, the values of V and F_i will be smaller away from the safety distance L, but will increase rapidly when d comes close to L. The constant $c \geq 1$, is responsible for the final configuration of the vehicles alongside the road. If we choose $c = 1$ and $y = 0$, then $U(y) = 0$, hence the vehicles will converge to the middle of the road forming a platoon. On the other hand, if $c > 1$ we have that $U(y) = 0$ for an area around $y = 0$, and thus the vehicles converge between $-\frac{a\sqrt{c-1}}{\sqrt{c}} \leq y \leq \frac{a\sqrt{c-1}}{\sqrt{c}}$.

In order to satisfy the objectives followed by the decentralized feedback laws, a control Lyapunov function [6], [30], which also possesses characteristics of barrier functions is applied. Thus, a function H, the total energy of a set of n vehicles derives:

$$\begin{aligned} H(w) := & \frac{1}{2}\sum_{i=1}^{n}\left(v_i\cos(\theta_i) - v^*\right)^2 + \frac{1}{2}\sum_{i=1}^{n} v_i^2 \sin^2(\theta_i) \\ & + \sum_{i=1}^{n} U(y_i) + \frac{1}{2}\sum_{i=1}^{n}\sum_{j\neq i} V(d_{i,j}) + A\sum_{i=1}^{n}\left(\frac{1}{\cos(\theta_i) - \cos(\varphi)} - \frac{1}{1-\cos(\varphi)}\right) \end{aligned} \tag{4.11}$$

This Lyapunov function consists of three parts, the kinetic energy, the potential energy and a penalty term. The kinetic energy is represented by the first two terms, the potential energy of the system is based on the third and fourth term and last is a penalty term which blows up when $\theta_i \to \pm\varphi$. Also, $A > 0$ is a parameter of the controller and the Lyapunov function and $v^* \epsilon (0, v_{max})$ is the desired longitudinal velocity, and $\varphi\epsilon\left(0, \frac{\pi}{2}\right)$ is a constant that should always satisfy the following inequality:

$$\cos(\varphi) \geq \frac{v^*}{v_{max}}$$

By using the last term, the feedback laws for each vehicle can be designed as shown below:

$$u_i = -\left(v^* + \frac{A}{v_i\left(\cos(\theta_i) - \cos(\varphi)\right)^2} \right)^{-1} \left(\mu_1 v_i \sin(\theta_i) + U'(y_i) + p\sum_{j\neq i} V'(d_{i,j}) \frac{(y_i - y_j)}{d_{i,j}} + \sin(\theta_i) F_i \right) \quad (4.12)$$

$$F_i = -\frac{k_i(w)}{\cos(\theta_i)}\left(v_i \cos(\theta_i) - v^*\right) - \frac{1}{\cos(\theta_i)} \sum_{j\neq i} V'(d_{i,j}) \frac{(x_i - x_j)}{d_{i,j}} \quad (4.13)$$

$$k_i(w) = \mu_2 + \frac{1}{v^*} \sum_{j\neq i} V'(d_{i,j}) \frac{(x_i - x_j)}{d_{i,j}} + \frac{v_{\max}\cos(\theta_i)}{v^*\left(v_{\max}\cos(\theta_i) - v^*\right)} f\left(-\sum_{j\neq i} V'(d_{i,j}) \frac{(x_i - x_j)}{d_{i,j}} \right) \quad (4.14)$$

where μ_1, μ_2 are constants controlling the rotation and acceleration rate respectively, and $f \in C^1(\Re)$ is any function that satisfies $\max(x,0) \le f(x)$ for all $x \in \Re$. We utilize the following f function [6]:

$$f(x) = \frac{1}{2\varepsilon} \begin{cases} 0 & if\ x \le -\varepsilon \\ \left(x + \varepsilon\right)^2 & if\ -\varepsilon < x < 0 \\ \varepsilon^2 + 2\varepsilon x & if\ x \ge 0 \end{cases} \quad (4.15)$$

for every $\varepsilon > 0$, which allows the longitudinal acceleration to be regulated as desired.
The term $k_i(w)$ seen in the acceleration function $F_i(t)$ is a controller responsible for maintaining the vehicles speeds positive and lower than the speed limit. Concerning the second term of the acceleration function (4.13), is the summation of the repelling forces acting between all vehicles. As already mentioned in the introduction, there is a nudging effect, a "pushing" force taking action between vehicles. If V is decreasing then $-V'(d_{i,j}) \frac{(x_i - x_j)}{d_{i,j}} > 0$, and if vehicle j is behind vehicle i, then nudging [5], [6], [27] is introduced between those two vehicles, meaning j is "pushing" i in order to increase its velocity. We should also take into account that nudging will not jeopardize traffic safety, such as vehicles collisions, but also will not force vehicles gain parameters outside the Ω set. Lastly, we have to make clear than only information from vehicles with distance less than $\lambda > 0$ are needed by the feedback laws. Furthermore, the only information needed is the distance from the adjacent vehicles, whilst no other information is required such as their velocities or their wheel orientations.

5. Numerical investigation of solutions

First of all, the computer we used for all of the simulation is a personal computer with 16 gigabytes of RAM and an AMD Ryzen 7 1700 Eight-Core Processor with 3.00 GHz. As for the software we use Matlab R2018a and C language run in CodeBlocks with MinGW 64 bit 8.1.0 compiler. From now on we will always pronounce the software used for gaining the results presented.

In order to analyze better the simulations, a set of initial parameters is randomly picked and the solutions gained from each method are compared with each other. For this numerical investigation we assume that all vehicles have the same length $\sigma = 5m$ and operate in a lane free road with an ideal velocity of $v = 30\,{}^{m}/_{s}$ and width $a = 7.2m$. The vehicles must not exceed the maximum velocity of $v_{max} = 35\,{}^{m}/_{s}$ and set $\varphi = 0.25$, thus we obtain the optimal eccentricity and safety distance $p = 5.11$ and $L = 5.59m$, respectively. Furthermore, we select $\varepsilon = 0.2$, $\lambda = 25m$ and the design parameters $c = 1.5$, $q = 3 * 10^{-3}$. The simulations were performed for a time period of 500 seconds with an initial step size of $h = 0.01$.

The results shown below are all gained from random set of initial parameters, where all of them were gained with respect to the Ω set. We gained initial parameters for $10, 20, 50, 100, 150$ and 200 vehicles, but for most of the presented results we tend to utilize a random set of 100 vehicles. The randomly chosen set of initial parameters is set number 2 and is presented at Appendix B.

Euler Method

First of all, we investigate the numerical solution of the Euler Method. In order to successfully approximate the real solution, the execution time is undermined by using a considerably small step size of $h = 0.01$, thus needing 50000 iterations for a 500 seconds simulation. Obviously, by using a smaller step size we would gain a slightly better approximation in the expense of memory and execution time. Through Figures 11 to 14, the trajectories, velocities and accelerations of the vehicles are presented, all gained from an algorithm in C language. In Figure 11 are presented all the trajectories of all the vehicles in order to show that all vehicles remain within the boundaries of the road. Following, in Figure 12 we present 5 random vehicles trajectories to observe how vehicles change direction to avoid collisions with others vehicles or the boundaries of the road.

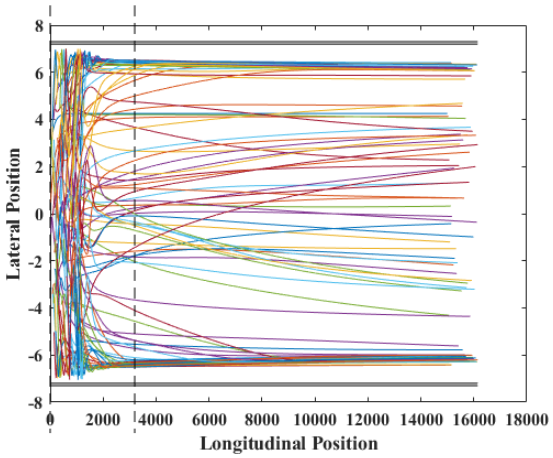


Figure 11: Vehicles Trajectories for Euler Method

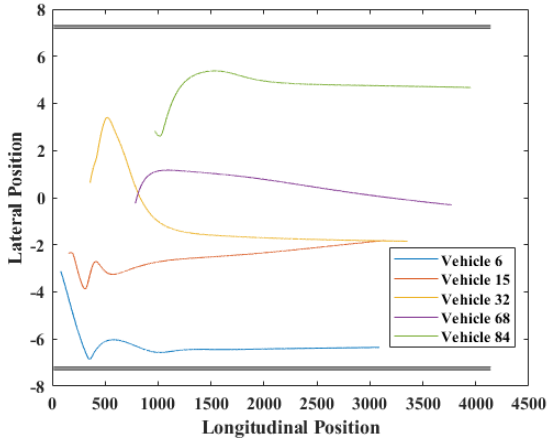


Figure 12: 5 Random Vehicles Trajectories for Euler Method

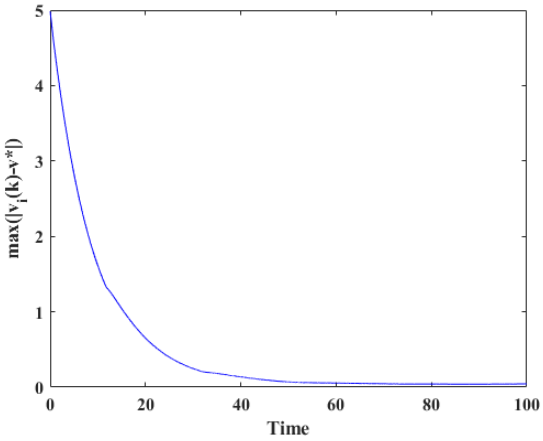


Figure 13: Vehicles Velocities Convergence for Euler Method

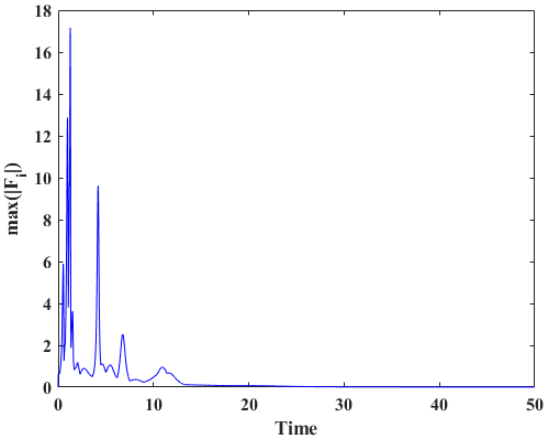


Figure 14: Vehicles Accelerations Convergence for Euler Method

By using a bigger step size $h = 0.1$, in order to gain a faster approximation, Euler Method is unable to approximate the numerical solution and in such way the vehicles crash. This can be observed below at Figure 15. Note however that the step size $h = 0.01$, that produced the "correct" previous approximation, does not imply that an approximation of the solution can always be obtained, and for a different set of initial conditions, the Euler method may fail again. For such cases an even smaller step size should be selected.

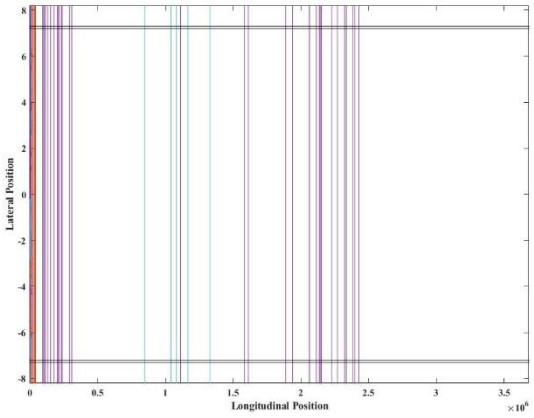


Figure 15: Failure to approximate for Euler Method

Heun Method

Following, the Heun Method can also approximate the numerical solution for a rather small step size of $h = 0.01$. This method needs even more execution time due to the increased calculations needed. However, Heun is able to produce a better approximation due to the increased calculations which also affect the ability to approximate a solution with a slightly higher step size. The same number of iterations is performed as with Euler Method and the algorithm is written also in C language. The results are presented at Figures 16 to 19.

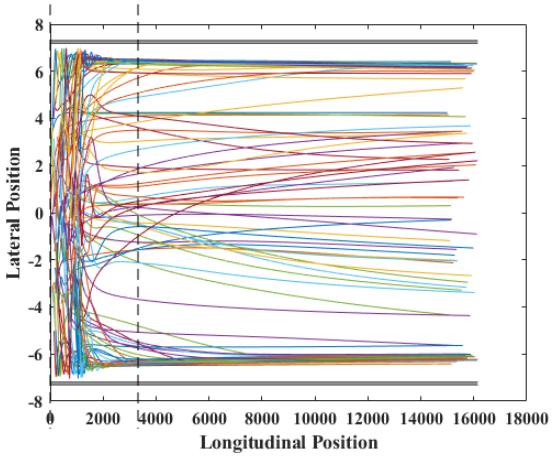


Figure 16: Vehicles Trajectories for Heun Method

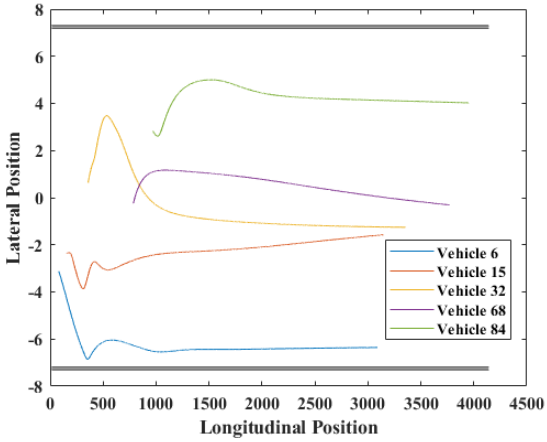


Figure 17: 5 Random Vehicles Trajectories for Heun Method

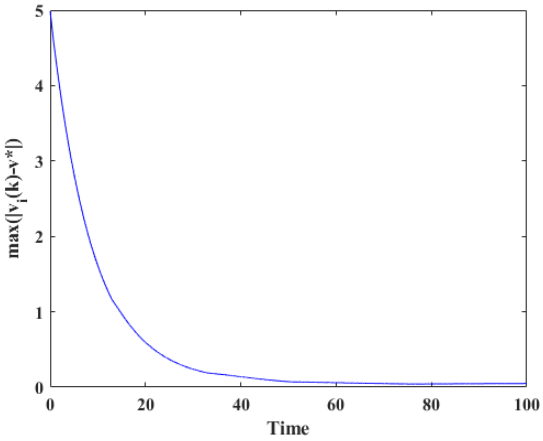


Figure 18: Vehicles Velocities Convergence for Heun Method

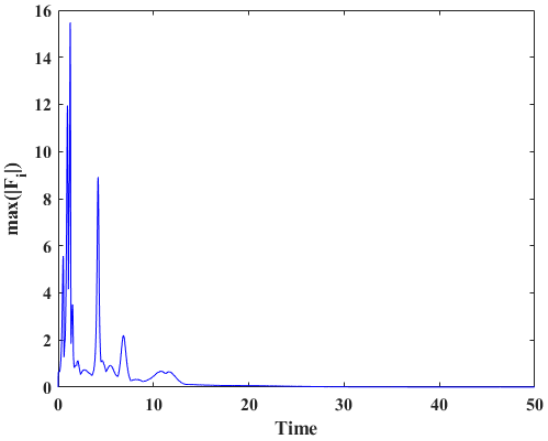


Figure 19: Vehicles Accelerations Convergence for Heun Method

Even though Heun Method is providing better approximations than Euler Method, it still cannot approximate the numerical solution for a bigger step size such as $h = 0.1$, and it can be observed at the following Figure.

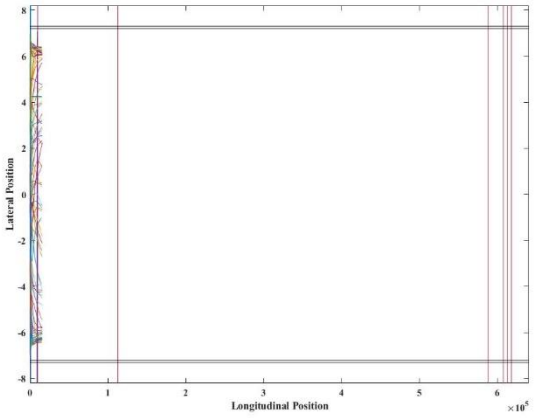


Figure 20: Failure to approximate for Heun Method

Adaptive Method

In contrast, the adaptive technique manages to overcome the large initial step size, $h = 0.1$, producing a good solution in less iterations, hence is less time. Note however, that the step size may become smaller than $h = 0.01$ at certain times, to retain the numerical stability of the system. Figures 21 to 24, present the trajectories, velocities and accelerations of the vehicles. This algorithm was also written in C language.

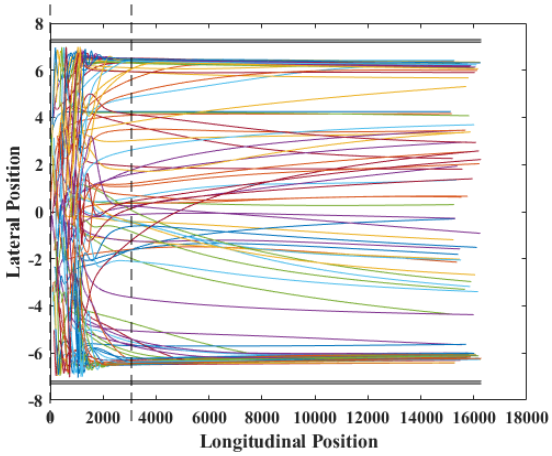


Figure 21: Vehicles Trajectories for Adaptive Method

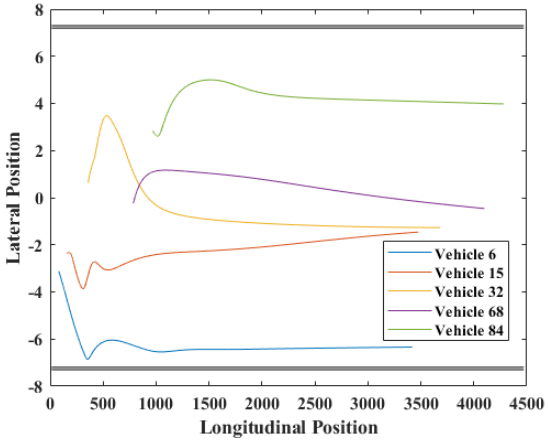


Figure 22: 5 Random Vehicles Trajectories for Adaptive Method

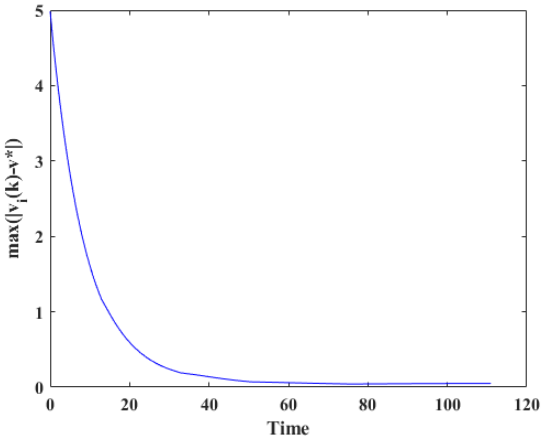


Figure 23: Vehicles Velocities Convergence for Adaptive Method

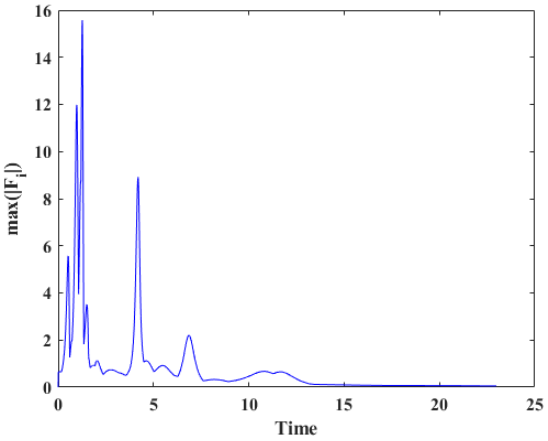
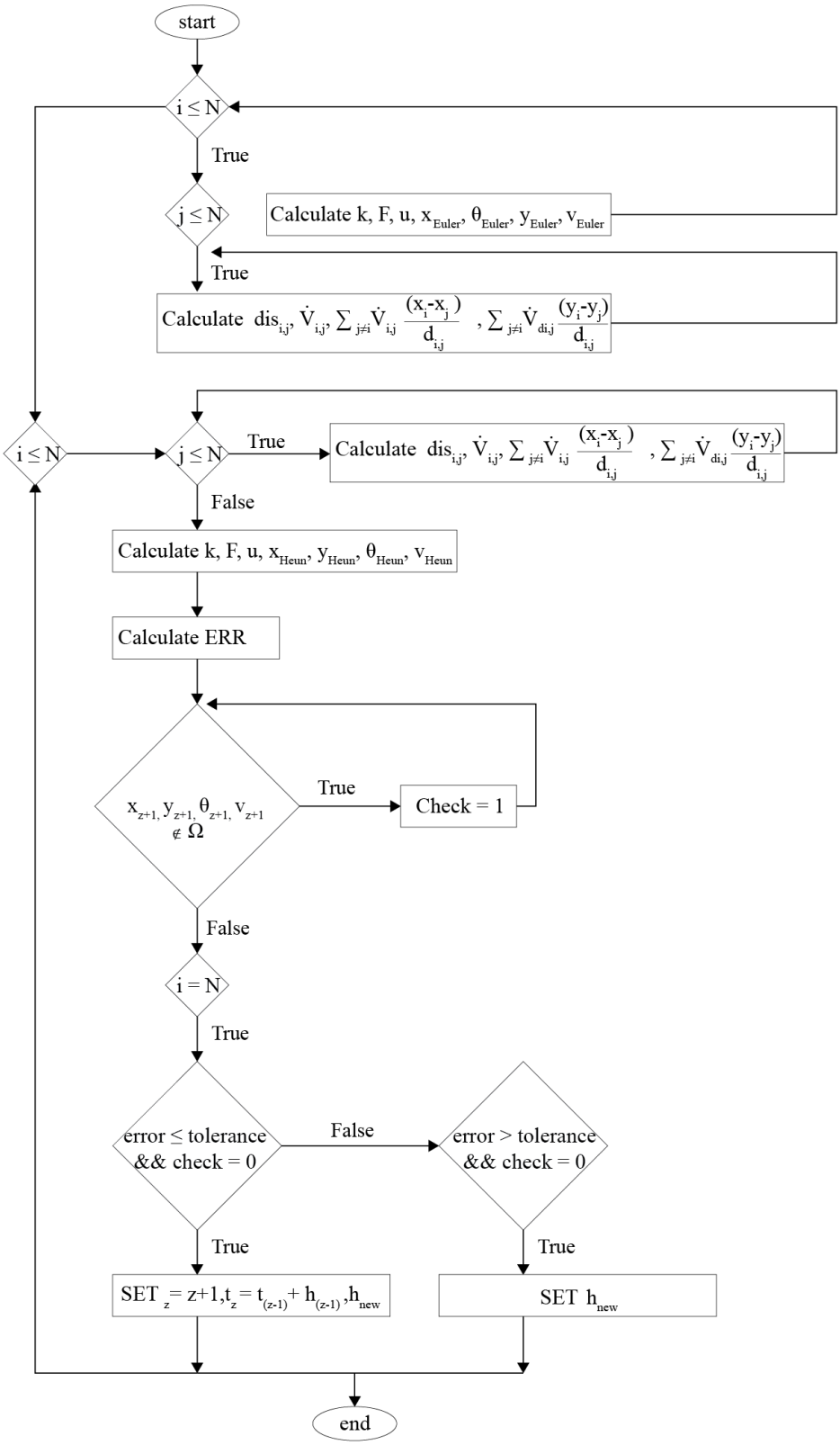


Figure 24: Vehicles Accelerations Convergence for Adaptive Method

Considering that the step size adapts in such way to keep the local error within an appointed tolerance, we can design our code in such way where for any initial parameters given with respect to the open Ω set, our system will not fail approximating a solution. This has to do with the fact that if the local error is over the appointed tolerance, our new step size is smaller than our old step size, thus we can decrease our step size till avoiding our systems failure.

Flowchart


start
i ≤ N
True
j ≤ N
Calculate k, F, u, x_{Euler}, θ_{Euler}, y_{Euler}, v_{Euler}
True
Calculate $dis_{i,j}$, $\dot{V}_{i,j}$, $\sum_{j\neq i}\dot{V}_{i,j}\frac{(x_i-x_j)}{d_{i,j}}$, $\sum_{j\neq i}\dot{V}_{di,j}\frac{(y_i-y_j)}{d_{i,j}}$
i ≤ N
j ≤ N
True
Calculate $dis_{i,j}$, $\dot{V}_{i,j}$, $\sum_{j\neq i}\dot{V}_{i,j}\frac{(x_i-x_j)}{d_{i,j}}$, $\sum_{j\neq i}\dot{V}_{di,j}\frac{(y_i-y_j)}{d_{i,j}}$
False
Calculate k, F, u, x_{Heun}, y_{Heun}, θ_{Heun}, v_{Heun}
Calculate ERR
$x_{z+1}, y_{z+1}, \theta_{z+1}, v_{z+1} \notin \Omega$
True
Check = 1
False
i = N
True
error ≤ tolerance && check = 0
False
error > tolerance && check = 0
True
True
SET $_z$ = z+1,$t_z = t_{(z-1)} + h_{(z-1)}$,$h_{new}$
SET h_{new}
end

Above is presented a flowchart of the algorithm we used for the Adaptive Method. For a more detailed algorithm you may look at the pseudo code, at Appendix A.

Adaptive step size through Systems Energy

This technique is similar to the Adaptive Method, but differs in the way it adapts the step size. For this process the error depends on the systems energy, instead of the parameters of each vehicle. Hence, the systems energy has to be computed at every step and compared with another approximation of the systems energy. The second approximation can either be gained by the Euler Method or the previous systems energy can be used. If the energy of the system is greater than zero, less than the energy of the previous step and of course all the parameters are within the open Ω set, the new step size is increased. Otherwise, the step size is decreased and the iteration is performed again. If the second approximation is gained by the Euler Method the error is computed as:

$$err = \sqrt{\left(\frac{H_{EULER} - H_{HEUN}}{sc_H}\right)^2}$$

On the other hand, if the previous approximation of the systems energy is used, the error may be computed as:

$$err = \min\left(facmax, \max\left(facmin, var\left|\frac{H_{z+1} - H_z}{H_z}\right|\right)\right)$$

After trials with both errors, we decided to use the one arising between the two different Runge – Kutta methods. This method allows us to easily normalize the error between 0 and 1, in order to be able to compare it with the given tolerance of the local error, regarding the adaptive step size control. Following the embedded Runge – Kutta adaptive method, this technique as mentioned adapts the step size via the energy of the system. With this method we can also achieve bigger step sizes, hence produce solutions in less iterations. Using the systems energy as a decision making attribute, we confirm that the energy decreases step by step, rather than evaluating whether the difference between the approximations from two methods is small. This method is more focused on a quick development of stability over our system. Below, from Figure 25 to 28, are presented the results for the Adaptive Method through the Systems Energy performed by a code written in Matlab.

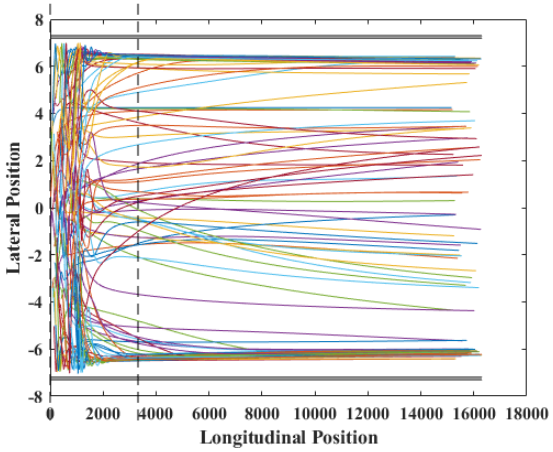


Figure 25: Vehicles Trajectories for Adaptive Method through Systems Energy

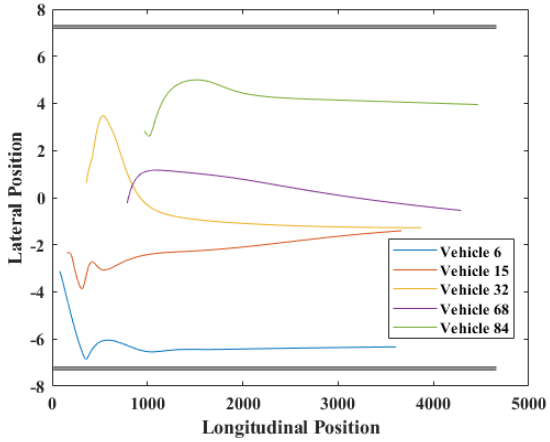


Figure 26: 5 Random Vehicles Trajectories for Adaptive Method through Systems Energy

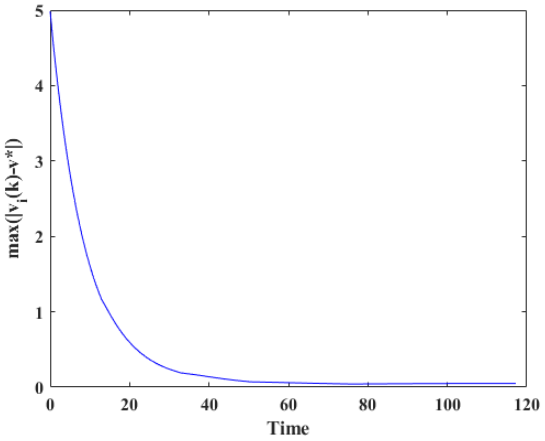


Figure 27: Vehicles Velocities Convergence for Adaptive Method through Systems Energy

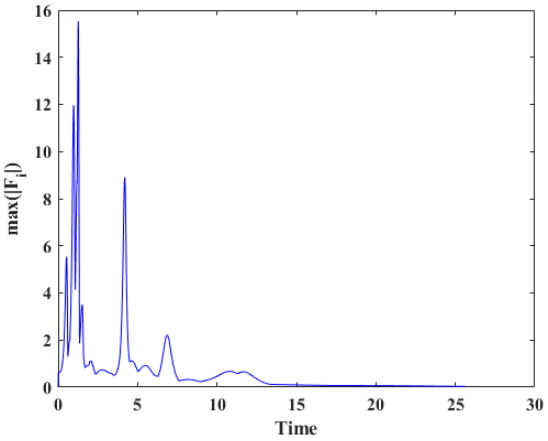


Figure 28: Vehicles Accelerations Convergence for Adaptive Method through Systems Energy

6. Comparisons between Numerical Approximation Methods

In order to observe the differences and make comparisons between the trajectories obtained from two different numerical approximation methods, we may plot the trajectories from both methods. That way, we can visually understand whether there are noticeable differences created due to bigger step sizes or different methods. On the other hand, if we want to better understand and evaluate a methods accuracy, we have to research the global error of each method. The difficulty we face is that there is no way to find the real solution of our system or it is too slow and demands a lot of computational costs. However, we may depend on an approximation of the global error. For the approximated global error we may use a really good approximation with a stable step size, thus we will utilize Heun method with a constant step size of 10^{-4} and the results will be named as the best approximation we can gain.

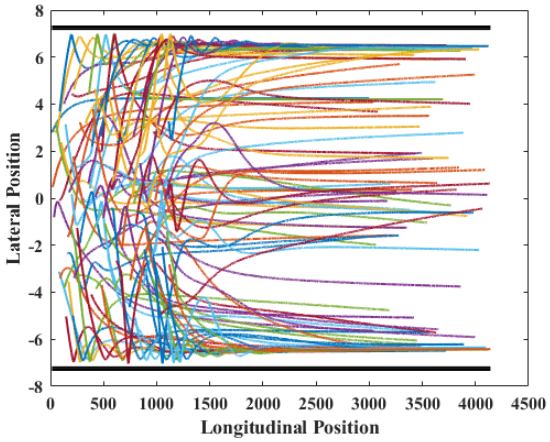


Figure 29: Best Approximated Vehicles Trajectories using Heun Method

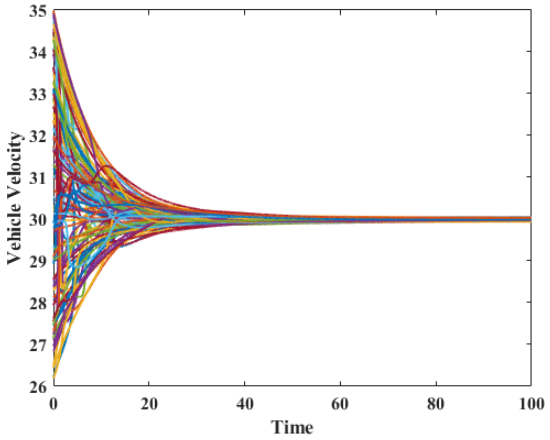


Figure 30: Best Approximated Vehicles Velocities using Heun Method

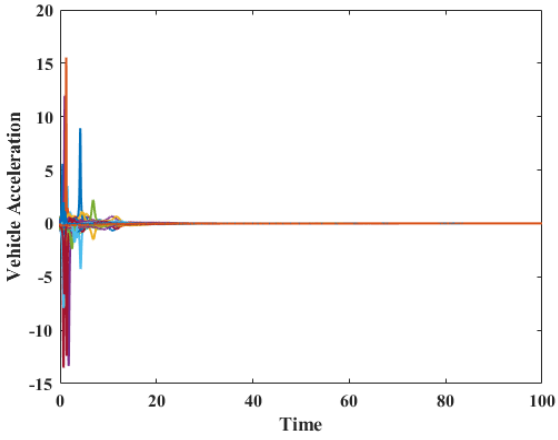


Figure 31: Best Approximated Vehicles Accelerations using Heun Method

Above are presented the Vehicles Trajectories, Velocities and Accelerations concerning Heun method with a constant step size of $h = 10^{-4}$ for 100 seconds simulation. The code used is written in C language and needed 933 $seconds$ for the simulation. The initial parameters are shown in Appendix B. Even though we use a really small step size we can still observe high accelerations, reaching $15\,{}^{m}/_{s}$ which is around $1.5g's$. All the vehicles remain within the boundaries of the road and do not collide with each other, as also their velocities tend to $30\,{}^{m}/_{s}$ which is the set ideal velocity.

Differences between Heun with step size $\mathbf{h = 10^{-2}}$ and $\mathbf{h = 10^{-4}}$

This comparison is made in order to understand how much of a difference will a smaller but acceptable step size makes. By observing the parameters obtained by those two different step sizes, we cannot really tell how big the difference is. For example, at Figure 32 we present the trajectory of vehicle number 85 which presents the highest approximated error.

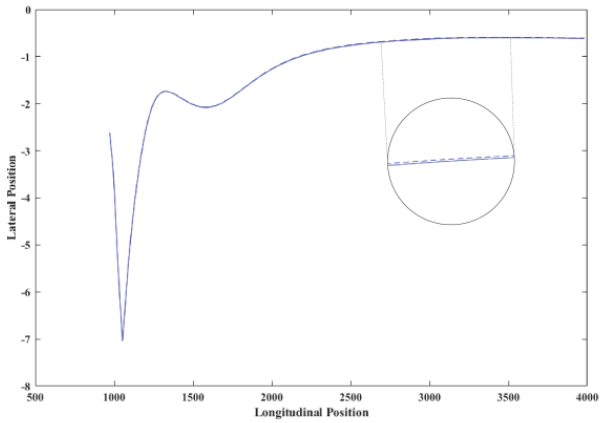


Figure 32: Vehicle 85 Trajectory difference between Best Approximation and Heun Method

Figures 33 and 34, present the differences between the velocities and wheel orientation that vehicle number 85 gains from Heun Method by the two different step sizes.

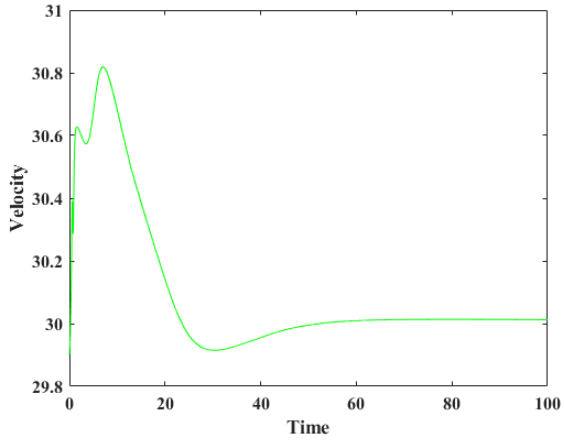


Figure 33: Vehicle 85 Velocity difference between Best Approximation and Heun Method

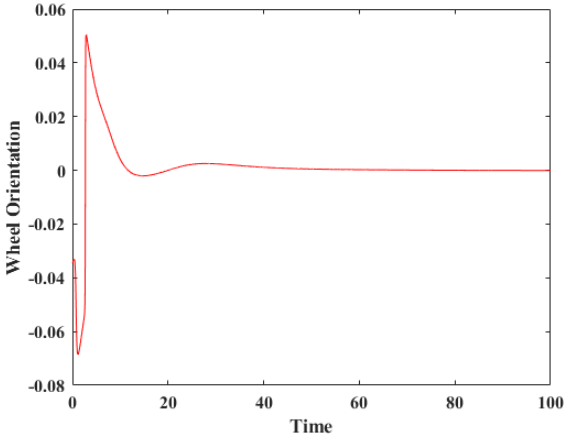


Figure 34: Vehicle 85 Wheel Orientation difference between Best Approximation and Heun Method

By comparing the results obtained using a data interpolation technique, we can find an approximation of the global error. Thus, we find the differences by interpolating data of the bigger step size to the smaller step size results and gain an approximation of the global error to be equal with 7.6047. Below, at Figures 35 and 36, we present the Absolute Local Error for vehicle 85 and the Absolute summation of the Local Error for all the vehicles.

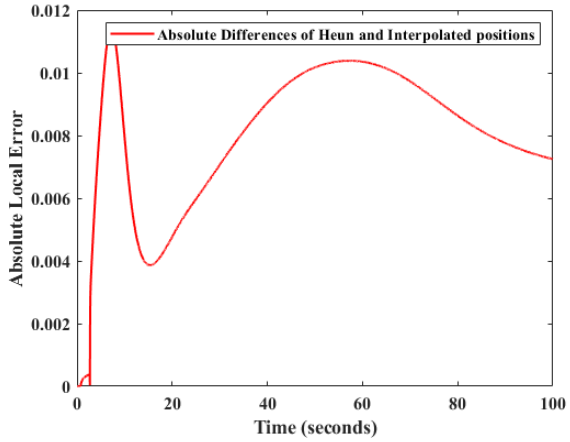


Figure 35: Absolute Local Error Vehicle 85 Heun Method

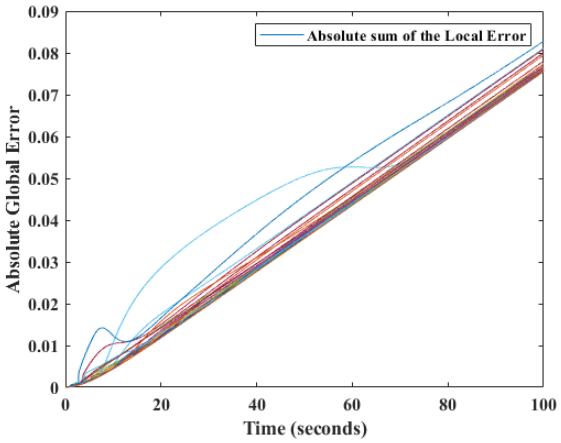


Figure 36: Absolute Global Error Heun Method

By adding the final local errors of all the vehicles, hence the global error of each vehicle, we obtain the Global Error for the whole set of vehicles. Those two Figures show that the differences by cutting the iterations in half are not that great and hence Heun method produces a fine approximation for a step size of $h = 10^{-2}$.

Differences between Euler, Heun and the Adaptive Method

First of all, the adaptive technique is way faster than using a constant step size. Although Heun's Method may seem the better choice when it comes to accuracy, we can find the global error of the Adaptive Method is rather small. We will use 100 vehicles for 100 seconds simulation and have a constant step size of 10^{-4} for Heun Method, to gain the best approximation we can.

Previously, we investigated two approximations gained from Heun Method and we were not able to detect any great differences. On the other hand, Euler Method seems to have some differences from our best approximation. This is understandable since Euler Method is simpler and also is a first order Runge–Kutta Method, whereas Heun Method is a second order Runge-Kutta Method. Below the trajectories of 9 vehicles are presented where 6 of them have different trajectories gained from Euler Method, but we have to state that the overall differences are not that noticeable.

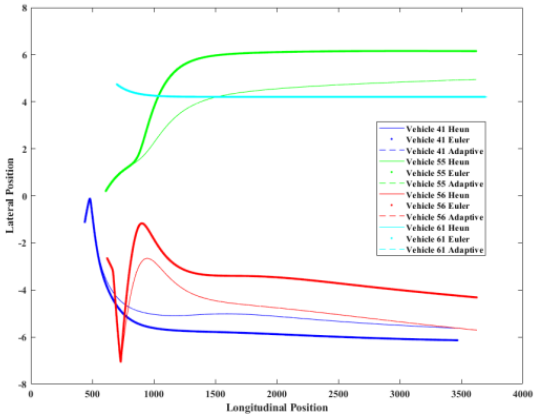


Figure 37: Euler, Heun and Adaptive Method Vehicles trajectories differences (1)

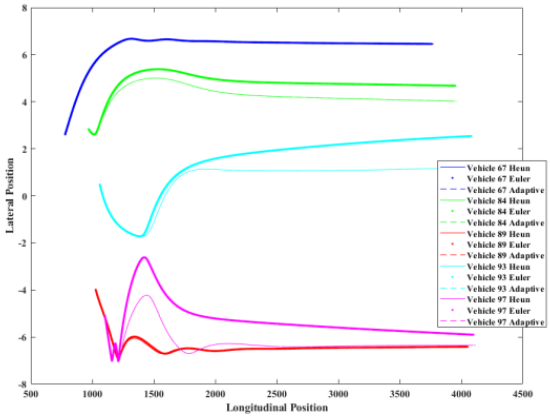


Figure 38: Euler, Heun and Adaptive Method Vehicles trajectories differences (2)

However, we still cannot define how great these differences are. Again we utilize a data interpolation technique to obtain an approximation of the global error. The Adaptive Method has a total Global Error equal with 8.4923 and Euler Method has 1813.309.This time vehicle 56 for Euler Method displays the worse approximation with global error equal to 19.3812 and for the Adaptive Method vehicle 31 has a total global error of 0.0914, and can all be observes at the Figures below.

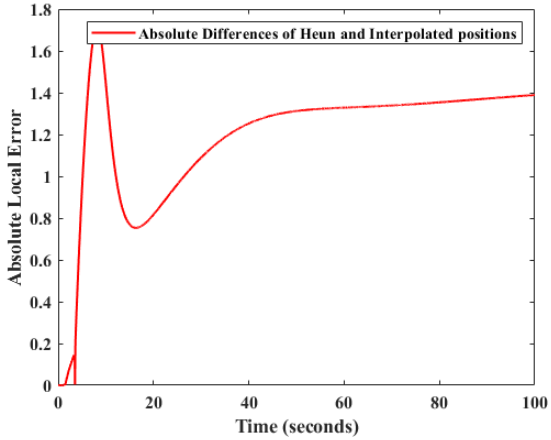


Figure 39: Absolute Local Error Vehicle 56 Euler Method

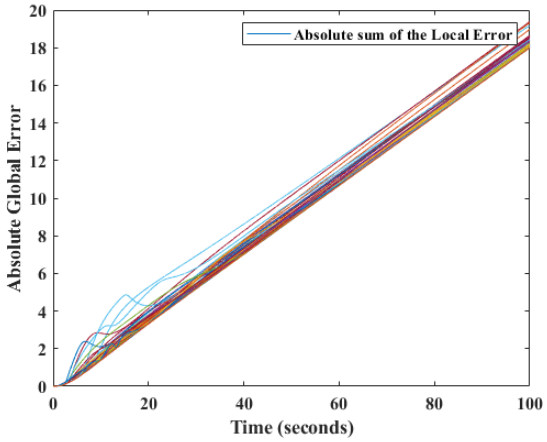


Figure 40: Absolute Global Error Euler Method

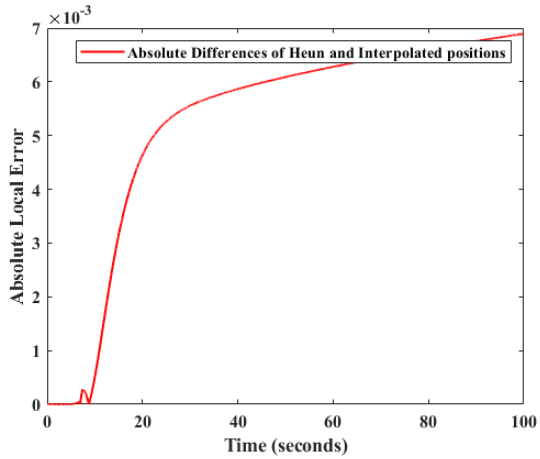


Figure 41: Absolute Local Error Vehicle 31 Adaptive Method

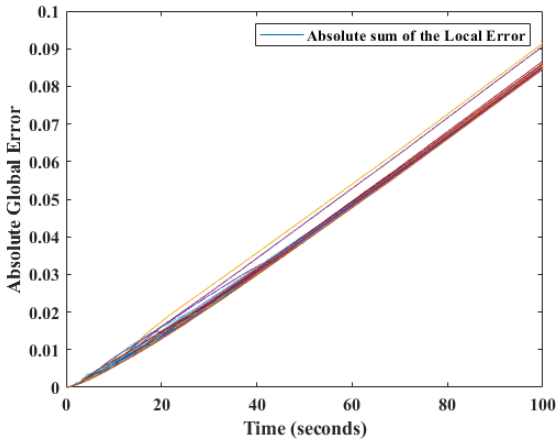


Figure 42: Absolute Global Error Adaptive Method

From the Figures above it is easily noticeable that the Adaptive Method produces a better approximation than Euler Method, even though it requires 1240 iterations and is extremely faster that the constant step size of $h = 10^{-2}$ of Euler Method. The simulation time for the Adaptive Method was 1.429 $seconds$ and for Euler Method 22.637 $seconds$, both written in C language.

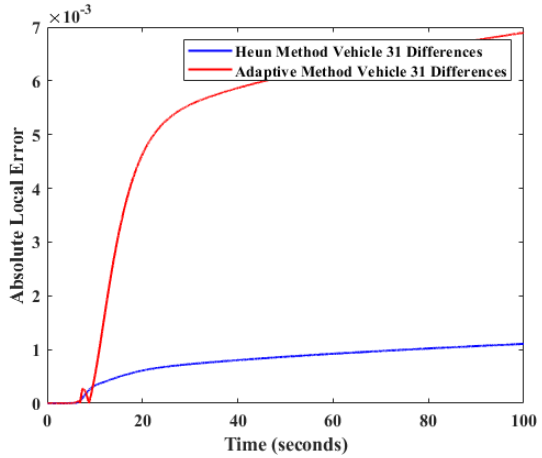


Figure 43: Heun and Adaptive Method Absolute Local Error for Vehicle 31

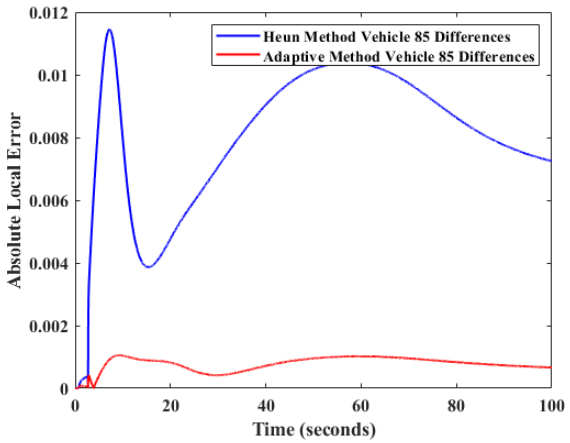


Figure 44: Heun and Adaptive Method Absolute Local Error for Vehicle 85

In Figures 43 and 44, we compare the two worse approximations for Heun Method with step size of $h = 10^{-2}$ and Adaptive Method. The simulation for Heun Method needed 49.647 $seconds$. Thus, we come to the conclusion that the Adaptive Method is generally better than using Heun Method with a constant step size. At Figure 45, we can see how the step size of the Adaptive Method, adapts through the simulation.

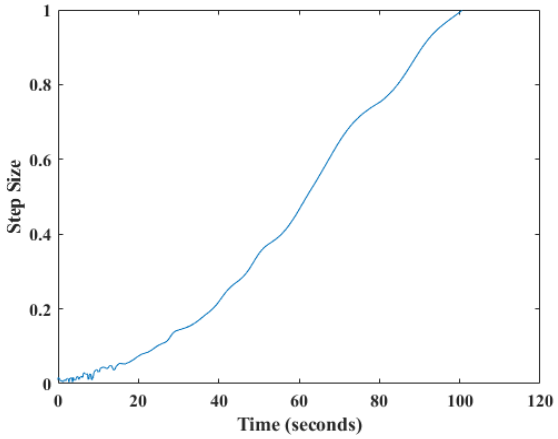


Figure 45: Adaptive Step Size

Differences between Adaptive Method, Richardson's Extrapolation and Adaptive using the Systems Energy

Comparisons should also be made between the Adaptive Methods, to acknowledge any advantages and disadvantages they may have, concerning the system given. Therefore, presented at the Figures below are 9 vehicles trajectories, 5 of which are not the same and 4 of them have almost the same trajectories. For this visual comparison we use as the reference trajectory, the one produced by the Adaptive Method, since we already know it is a good approximation. The algorithms for Richardson's Extrapolation and the Adaptive Method using the Systems Energy were written in Matlab.

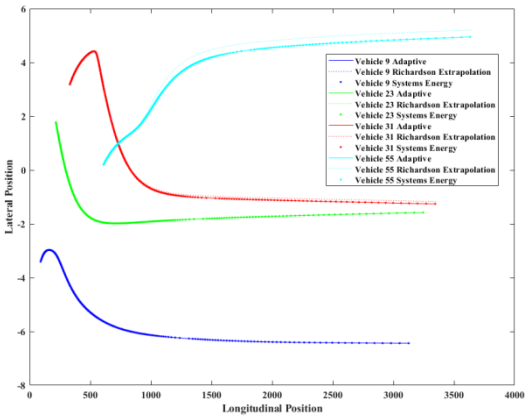


Figure 46: Adaptive Method, Richardson's Extrapolation and Adaptive Method through Systems Energy Vehicles trajectories differences (1)

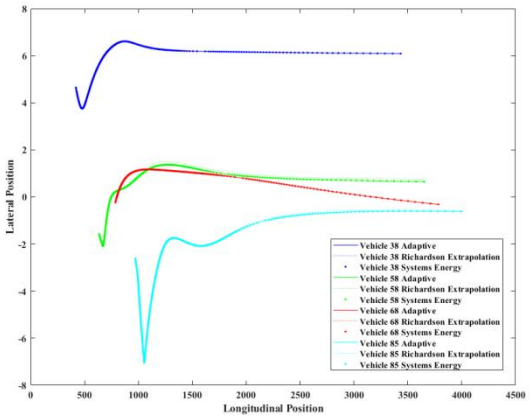


Figure 47: Adaptive Method, Richardson's Extrapolation and Adaptive Method through Systems Energy Vehicles trajectories differences (2)

For a thorough investigation we will again utilize the data interpolation technique, and analyze the approximations gained with Richardson's Extrapolation and the Adaptive Method through the Systems Energy against the Best approximation we have.

Richardson's Extrapolation produces a good approximation with Global Error equal with 24.5896, with the vehicle with the worst approximation being vehicle number 56 and its global error equal to 0.4861. The simulation time was 251.901 seconds.

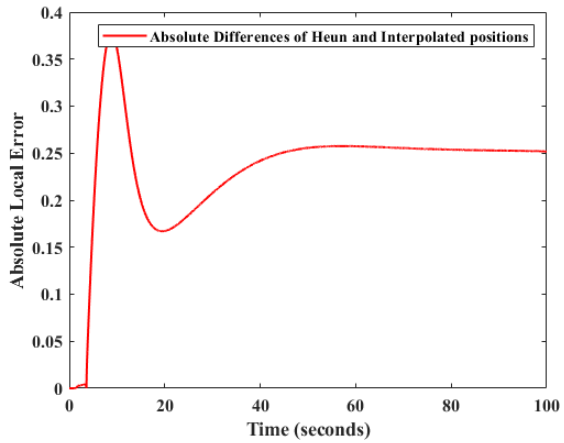


Figure 48: Absolute Local Error Vehicle 56 Richardson's Extrapolation Method

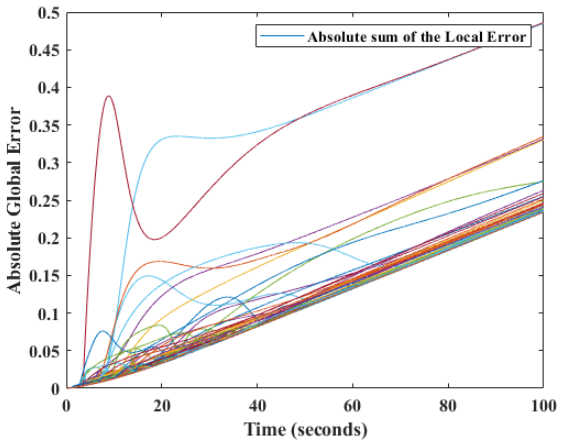


Figure 49: Absolute Global Error Richardson's Extrapolation Method

Adaptive Method through Systems Energy has a great simulation time but suffers a little bit concerning the accuracy. It has a total Global Error of 54.2835 with vehicle number 31 having the worst global error of 0.5520.

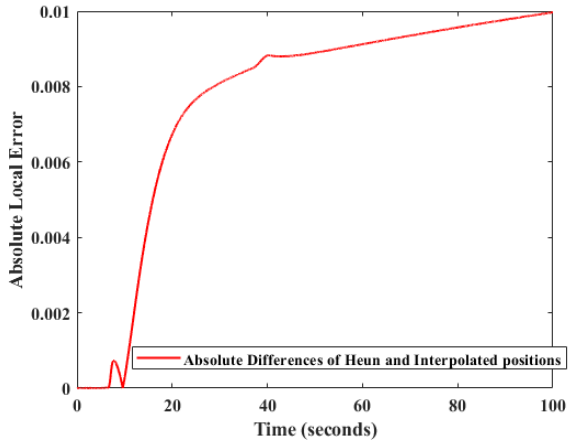


Figure 50: Absolute Local Error Vehicle 31 Adaptive Method through Systems Energy

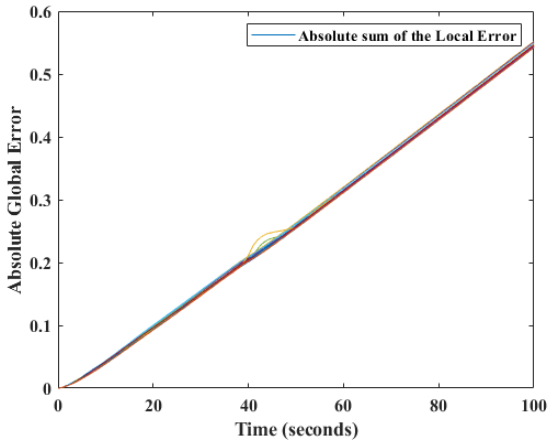


Figure 51: Absolute Global Error Adaptive Method through Systems Energy

At the following Figures we can observe the step sizes from Richardson's Extrapolation and the Adaptive Method through Systems Energy, respectively.

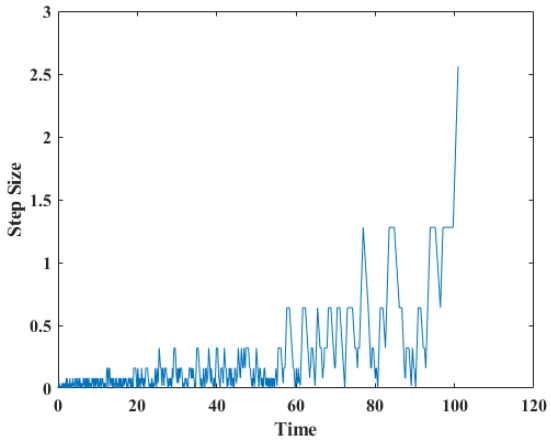


Figure 52: Richardson's Extrapolation Step Size

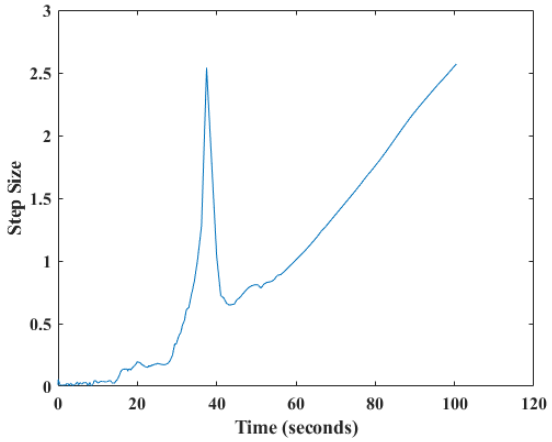


Figure 53: Adaptive Method through Systems Energy Step Size

Adaptive Methods Step Size Upper Bound Differences

Below at Figure 55, are presented the average time needed by the Adaptive Method to simulate for different step size upper bounds. For the simulations we used 20 sets of 100 vehicles for 500 seconds. The algorithm we used to perform these simulations was written in Matlab.

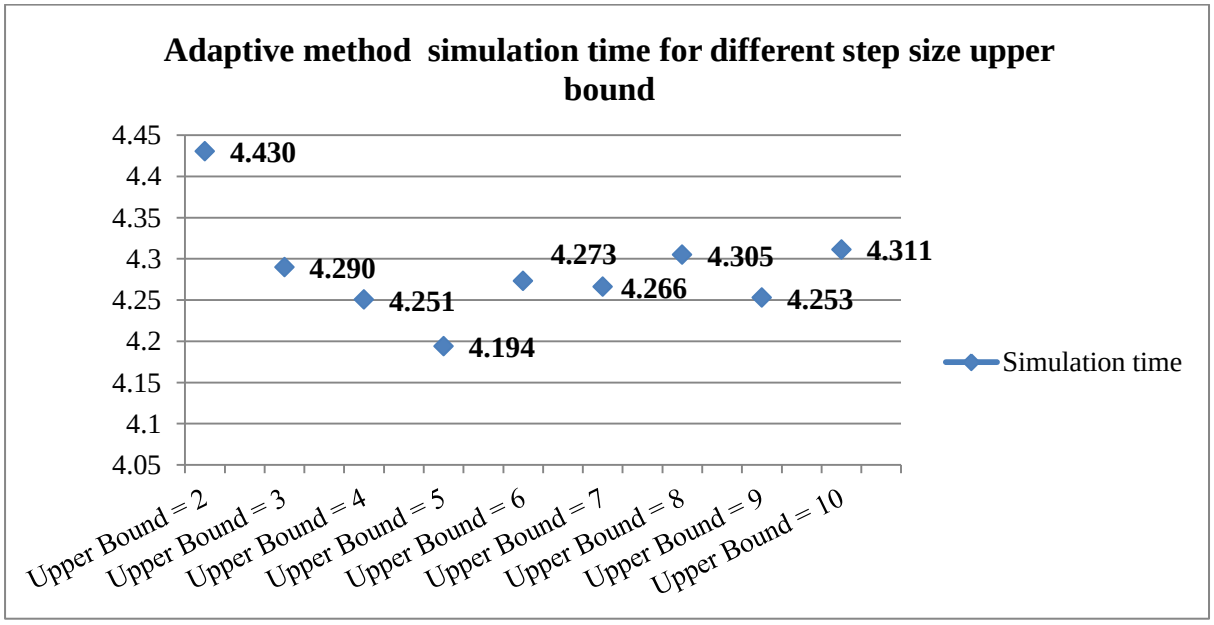


Figure 54: Adaptive Method's upper bound simulations

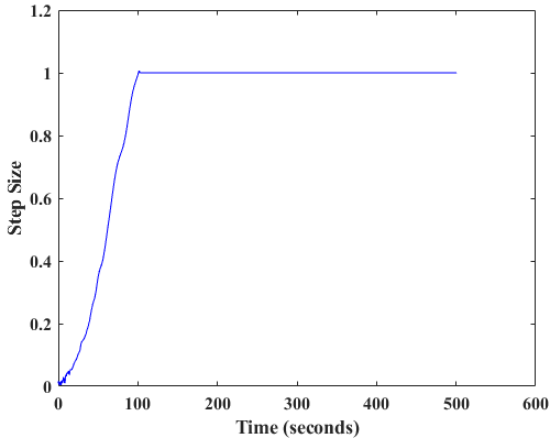


Figure 55: Adaptive Method's Step Size for $UB = 1$

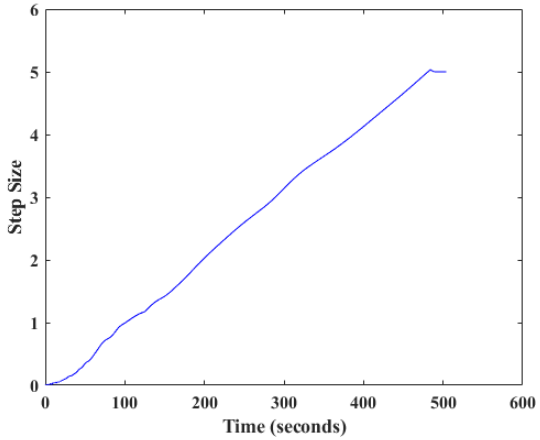


Figure 56: Adaptive Method's Step Size for $UB = 1$

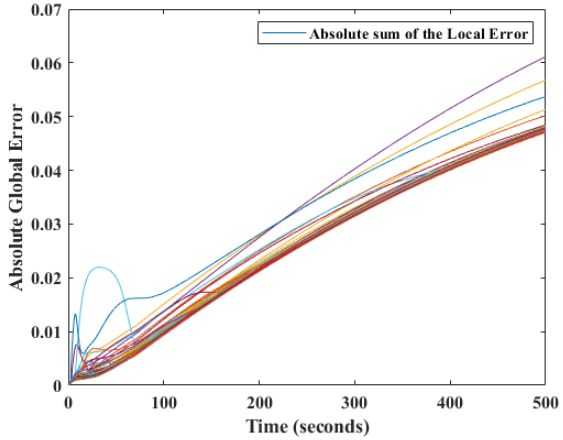


Figure 57: Absolute Global Error Adaptive Method for $UB = 1$

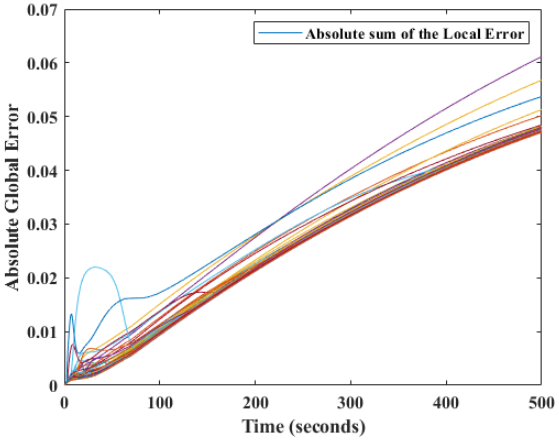


Figure 58: Absolute Global Error Adaptive Method for UB = 5

The differences between the Global Errors for Step Size with Upper Bound equal to 1 and 5 are almost equal, as we can observe from the Figures above. We can confirm the above allegation by using the results from a data interpolation technique, resulting to a Global Error equal to 4.76405 for $UB = 1$ and 4.76406 for $UB = 5$. With $UB = 1$ the Adaptive Method needed 1640 iterations with an average time of 4.430 seconds, whilst with $UB = 5$ it only needed 1401 iterations with average time of 4.194 seconds. Comparing those results, which we gained for a simulation of 500 seconds, with the ones we got for the 100 seconds simulations, we observe that we only needed 200 more iterations for 400 more seconds.

At Figure 60 we present the average times needed by the Adaptive Method through Systems Energy to simulate for different step size upper bounds. This algorithm was also written in Matlab.

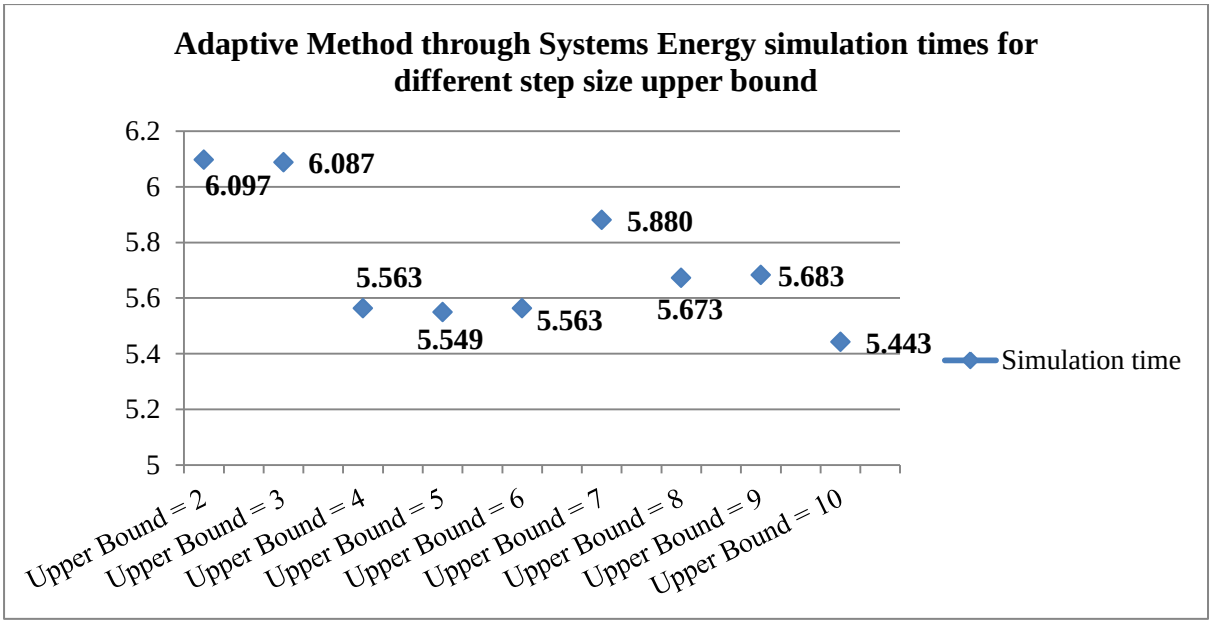


Figure 59: Adaptive Method through Systems Energy upper bound simulations

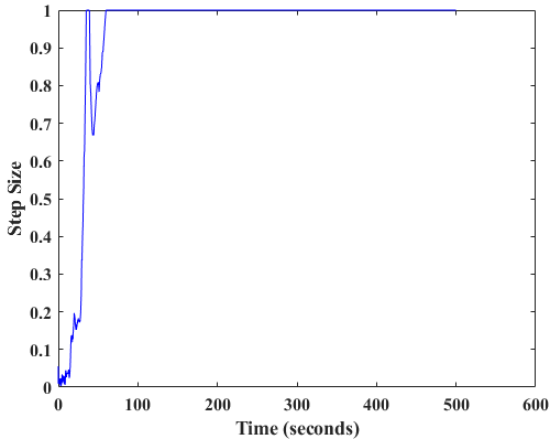


Figure 60: Adaptive Method through Systems Energy Step Size for $UB = 1$

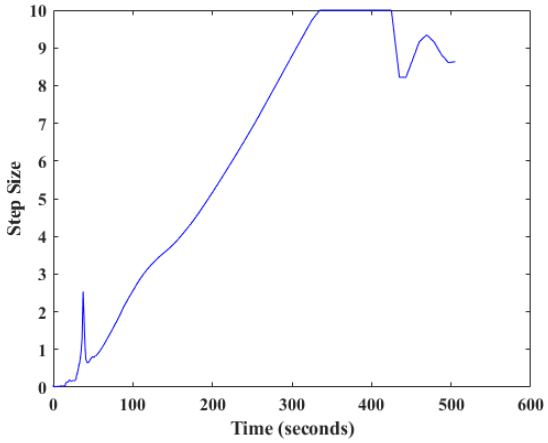


Figure 61: Adaptive Method through Systems Energy Step Size for $UB = 10$

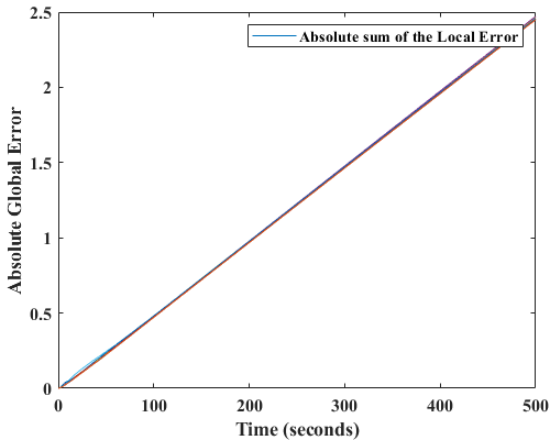


Figure 62: Absolute Global Error Adaptive Method through Systems Energy for $UB = 1$

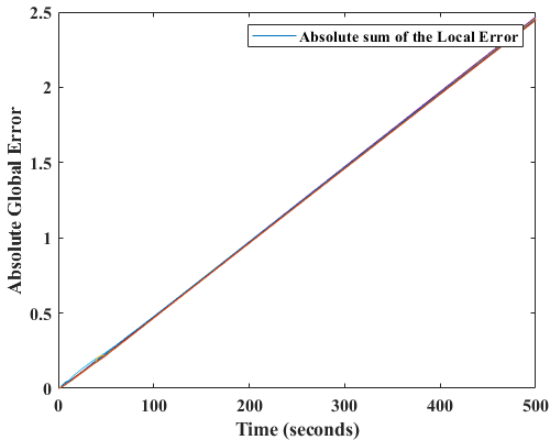


Figure 63: Absolute Global Error Adaptive Method through Systems Energy for $UB = 10$

Again we observe little differences for the Global Errors for the different Upper Bounds. For $UB = 1$ with an average time of 6.097, we gain a Global Error equal to 245.069 and for UB $= 10$ and an average time of 5.443 we have a Global Error equal to 244.641. Also we need 1401 iterations for approximating with an $UB = 1$, almost 400 more than with $UB = 10$, which needs 1048 iterations.

Adaptive Method Absolute and Relative Error Tolerances Differences

Concerning the Absolute and Relative Error Tolerances, we utilized 20 sets of 100 vehicles for 500 $seconds$. It is reasonable that whichever of those two gain a smaller value, the simulation time increases.

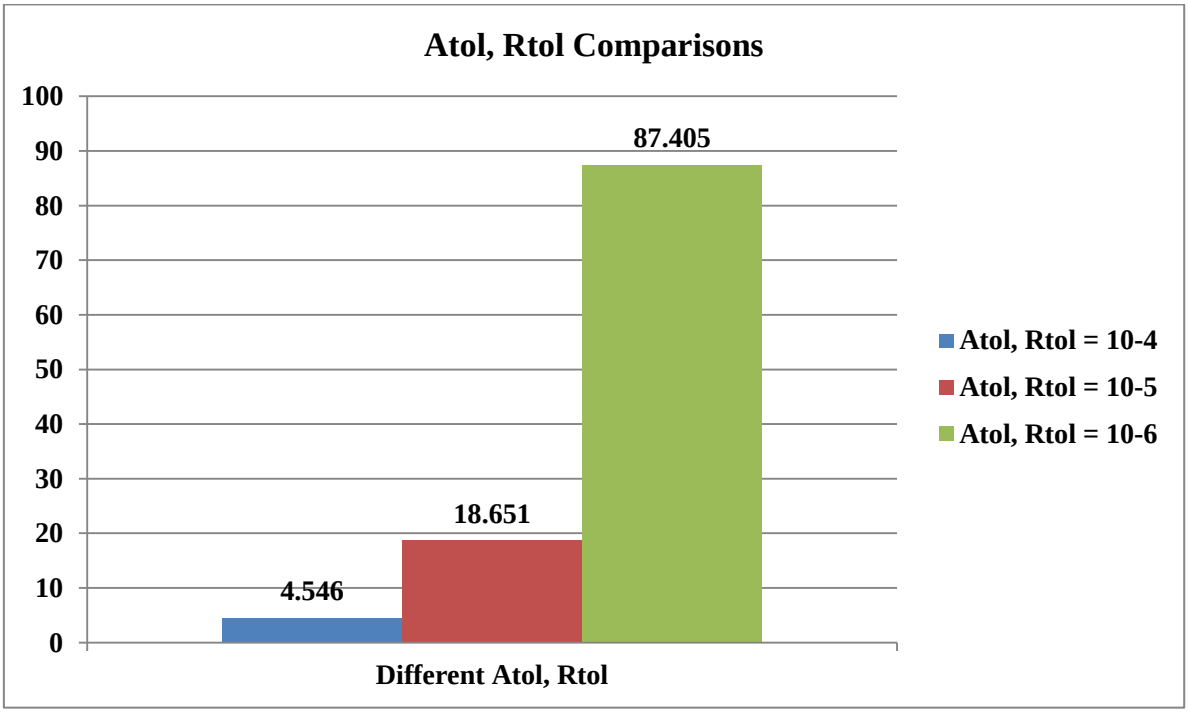


Figure 64: Simulation times for 3 different Atol and Rtol values

For gaining a better insight of the simulation results, we will use only the first 100 $seconds$ of each simulation, for the set number 2, and compare those results with the Best results we have using a data interpolation technique. For $Atol = 10^{-4}, Rtol = 10^{-4}$ the simulation time took 4.109 $seconds$, for $Atol = 10^{-5}, Rtol = 10^{-5}$ the simulation time took 15.38 $seconds$ and for $Atol = 10^{-6}, Rtol = 10^{-6}$ the simulation time took 72.941 $seconds$. Obviously for $Atol = 10^{-4}, Rtol = 10^{-4}$ we will gain the results from differences between Adaptive Method and the Best approximation, hence we will not evaluate the approximation again. For these simulations we utilized a Matlab algorithm.

For $Atol = 10^{-5}, Rtol = 10^{-5}$ the worst trajectory we obtain is for vehicle number 41 with Global Error equal to 0.0129. We also have a total Global Error of 1.235.

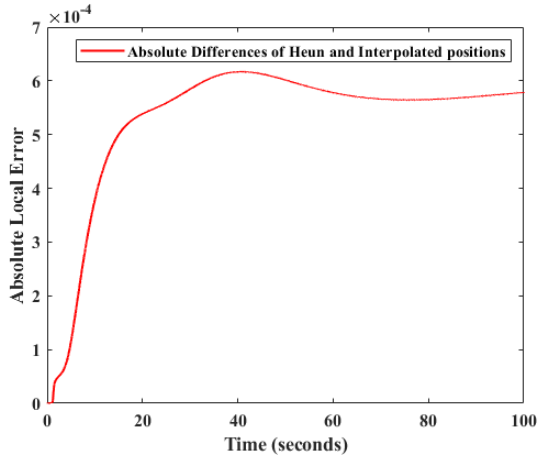


Figure 65: Absolute Local Error Vehicle 41 Adaptive Method Atol = 10^{-5}, Rtol = 10^{-5}

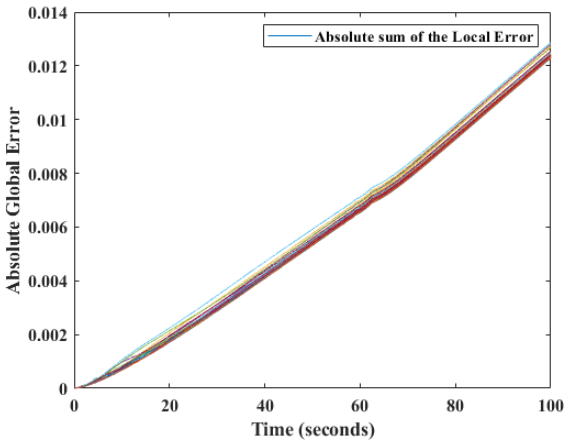


Figure 66: Absolute Global Error Adaptive Method Atol = 10^{-5}, Rtol = 10^{-5}

Concerning $h = 10^{-6}, Rtol = 10^{-6}$, we also gain the worse trajectory approximation for vehicle 41 with Global Error equal to 0.00386. The total Global Error was 0.349.

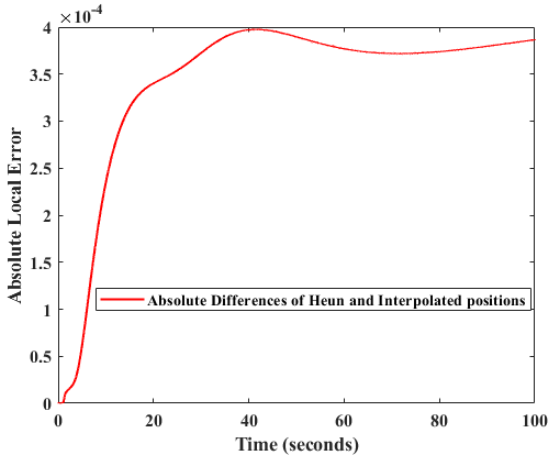


Figure 67: Absolute Local Error Vehicle 41 Adaptive Method Atol = 10^{-6}, Rtol = 10^{-6}

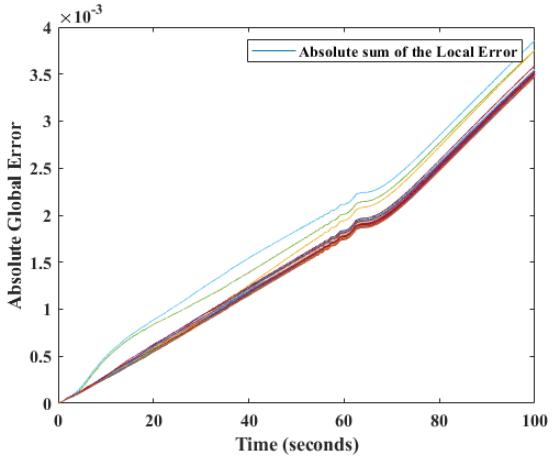


Figure 68: Absolute Global Error Adaptive Method Atol = 10^{-6}, Rtol = 10^{-6}

Time Differences between Matlab and C codes

At the Figure below, we present the time taken for executing the simulations in C language. For those simulations we used 20 different sets of initial parameters for $10, 20, 50, 100, 150$ and 200 vehicles and gained the average simulation time. As we can observe from the Figure, the quickest simulations are those using the Adaptive Method, next come the Euler Method and last are the Heun Method. This is understandable since Adaptive Method produces the approximation with way less iterations and Heun Method has more computations in order to approximate the solution.

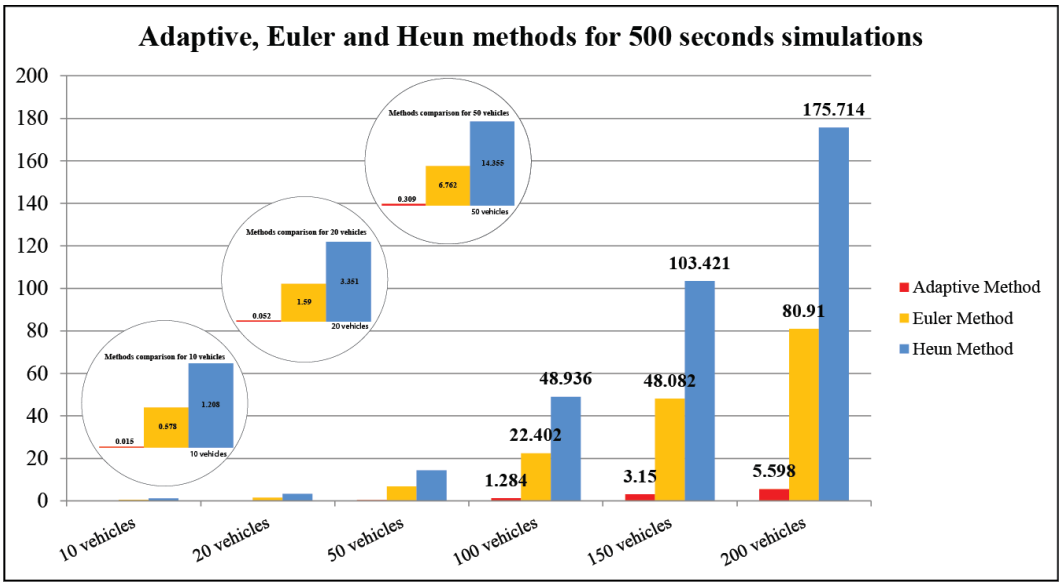


Figure 69: C language time simulations

We observe the same pattern with C language simulations for the simulations that were performed in Matlab. The main difference between Figure 46 and 47 is that for Matlab the simulation execution times are way bigger. For example, for 200 vehicles Heun Method needs an average of 175.714 seconds when run in C language, while it needs approximately 1706.946 seconds when run in Matlab which is about 10 times more.

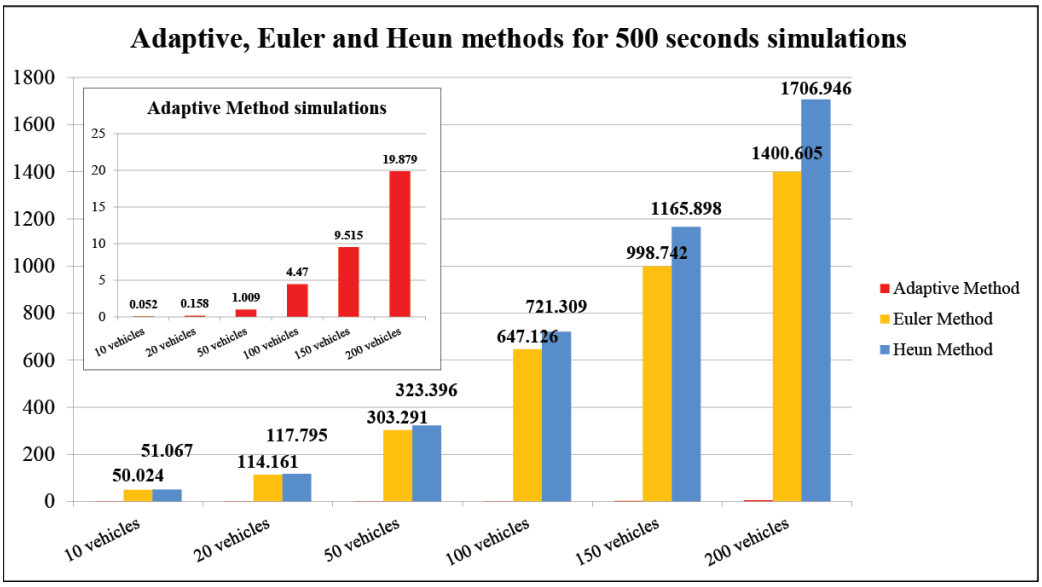


Figure 70: Matlab Simulations

7. Investigation for the repulsive potential function U_i and V_i

The repulsive potential functions are created and used to reassure the vehicles integrity. The potential functions V and U prevent inter vehicle collisions and collisions with the boundary of the road, respectively. Both of them are designed in such way that the repulsive force comes in smoothly when the vehicle's distance from a barrier or another vehicle is further from the "danger" zone and rapidly when the vehicle's distance between the boundaries of the road or another vehicle tends to zero. This rapid repulsive force could be translated into a rapid change of direction, or a great acceleration in the lateral direction or even a great deceleration. This problem is easily observed at the beginning of our simulations where we have gained random initial parameters from the Ω set, thus vehicles projected trajectories may cross each other or even the vehicles may be really close to each other. But still, when the distances tend to L the repulsive force must be great. Below at Figures 70 and 71 we observe the two ellipses surrounding the vehicles which are "responsible" for enabling the repulsive force between them.

Figure 71: Vehicles not exchanging information

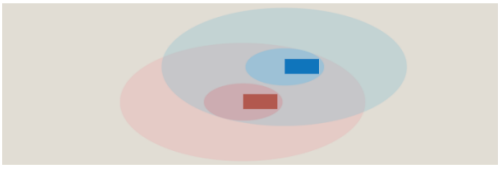


Figure 72: Vehicles exchanging information

In order to smoothen the repulsive effect when a vehicle or the boundaries of the road enter the faint colored ellipse, we have to gradually increase the repulsive force instead of rapidly increasing it when they come close to the dark colored ellipse. At all the previous simulations we have used the functions mentioned at paragraph 4, and more specifically the following ones:

$$V(d) = \begin{cases} q\dfrac{(\lambda - d)^3}{d - L}, & L < d \leq \lambda \\ 0, & d > \lambda \end{cases}$$

$$U(y) = \begin{cases} \left(\dfrac{1}{a^2 - y^2} - \dfrac{c}{a^2}\right)^4, & -a < y < -\dfrac{a\sqrt{c-1}}{\sqrt{c}} \text{ and } \dfrac{a\sqrt{c-1}}{\sqrt{c}} < y < a \\ 0, & -\dfrac{a\sqrt{c-1}}{\sqrt{c}} \leq y \leq \dfrac{a\sqrt{c-1}}{\sqrt{c}} \end{cases}$$

We are investigating those two functions and we wish to design them in such way, that they will repulse the vehicles in a more progressive way, while decreasing the produced accelerations for collision avoidance. Thus, we utilize "bell-shaped" functions that will gradually increase the repulsive force, before a rapid increase is needed. For these reasons we

will equip our function with an extra term, the Gaussian function [14]. It is the archetypal bell shaped function and can be encountered in many problems.

$$f(x) = a * e^{\left(-\frac{(x-b)^2}{2c^2}\right)}$$

where below we use $var_1 = a, var_2 = b, var_3 = c$. We utilize this extra term in order to create a local minimum between L and l, so shorter inter-vehicle distances may be maintained. Following, we present the currently used V function and the one we will investigate:

$$V(d) = \begin{cases} q\dfrac{(\lambda - d)^3}{d - L}, L < d \leq \lambda \\ 0 \quad , d > \lambda \end{cases}$$

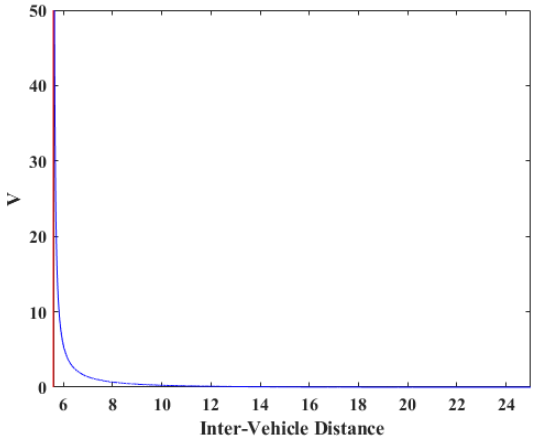


Figure 73: V Function

$$V'(d) = \begin{cases} \left(q\dfrac{-3(\lambda - d)^2(d - L) - (\lambda - d)^3}{(d - L)^2}\right), L < d \leq \lambda \\ 0 \quad , d > \lambda \end{cases}$$

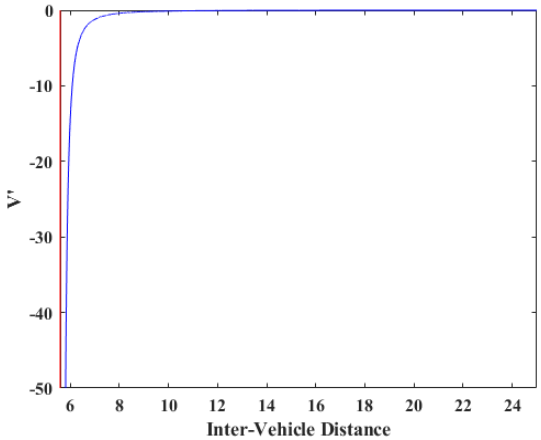


Figure 74: Derivative of V Function

$$V_{new}(d) = \begin{cases} q\dfrac{(\lambda - d)^3}{d - L} + var_1 e^{-(d-var_2)^2/(2var_3^2)}, L < d \leq \lambda \\ 0 \qquad\qquad\qquad\qquad\qquad\qquad\quad , d > \lambda \end{cases}$$

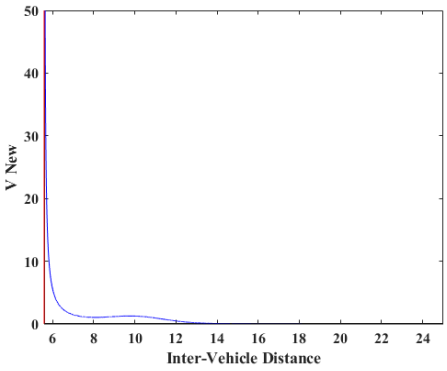


Figure 75: New V Function

$$V'_{new}(d) = \begin{cases} \left(q\dfrac{-3(\lambda - d)^2(d - L) - (\lambda - d)^3}{(d - L)^2}\right) - \dfrac{var_1(d - var_2)e^{-\frac{(d-var_2)^2}{(2var_3^2)}}}{(var_3^2)}, L < d \leq \lambda \\ 0 \qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\quad , d > \lambda \end{cases}$$

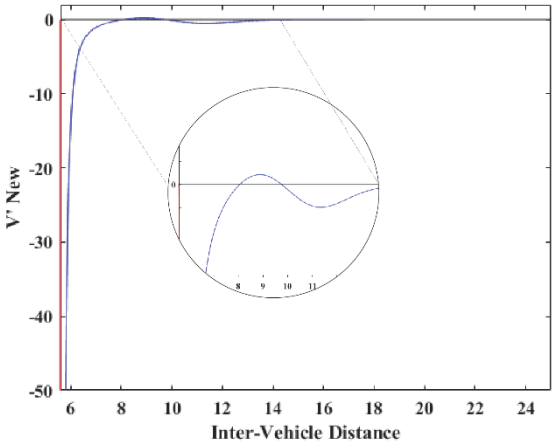


Figure 76: Derivative of the new V Function

The use of the derivatives is due to the fact that in our control functions, we utilize the derivatives of the U and the V functions, see paragraph 4 equations (4.12), (4.13), (4.14).

Yet, we still have to evaluate the new V function. This potential function, as we have already mentioned, is responsible for the vehicles not crashing with each other. Thus, by inserting this bell shaped function, we expect shorter inter vehicular distances. We performed two simulations for the Adaptive method, one for 100 seconds and a second for 500 seconds.

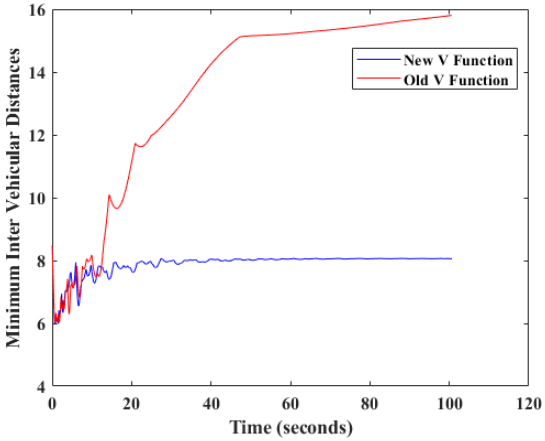


Figure 77: Inter Vehicular Distances 100 Seconds Simulation

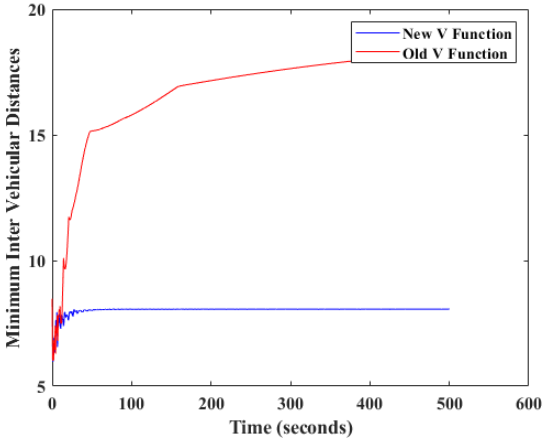


Figure 78: Inter Vehicular Distances 500 Seconds Simulation

From the Figures 77 and 78, we already observe that the New V Function is able to create shorter minimum inter vehicular distances, for the Adaptive Method. Following, at figures 79 and 80 we present the results for the New V Function for the Adaptive Method through Systems Energy, where we yet again observe shorter minimum inter vehicular distances.

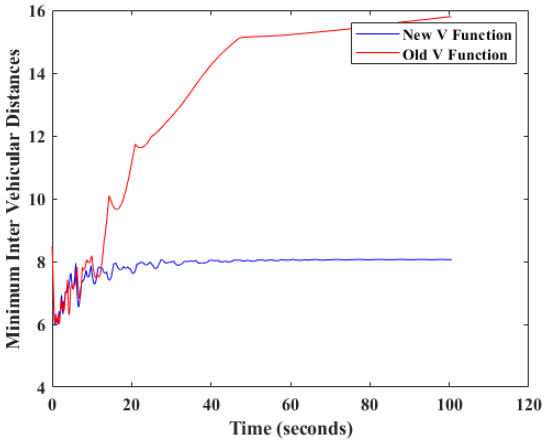


Figure 79: Inter Vehicular Distances 100 Seconds Simulation

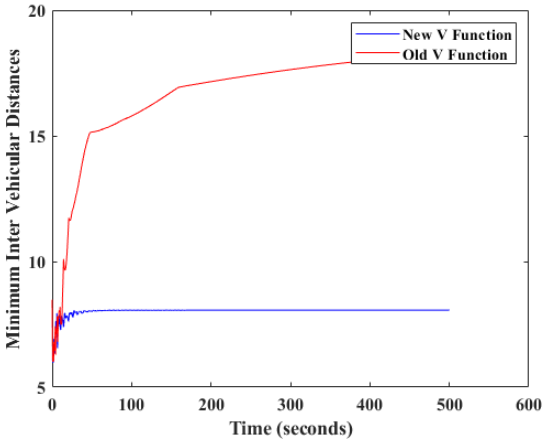


Figure 80: Inter Vehicular Distances 500 Seconds Simulation

Below are presented the currently used U function and the one we investigate:

$$U(y) = \begin{cases} \left(\frac{1}{a^2 - y^2} - \frac{c}{a^2}\right)^4, & -a < y < -\frac{a\sqrt{c-1}}{\sqrt{c}} \text{ and } \frac{a\sqrt{c-1}}{\sqrt{c}} < y < a \\ 0, & -\frac{a\sqrt{c-1}}{\sqrt{c}} \leq y \leq \frac{a\sqrt{c-1}}{\sqrt{c}} \end{cases}$$

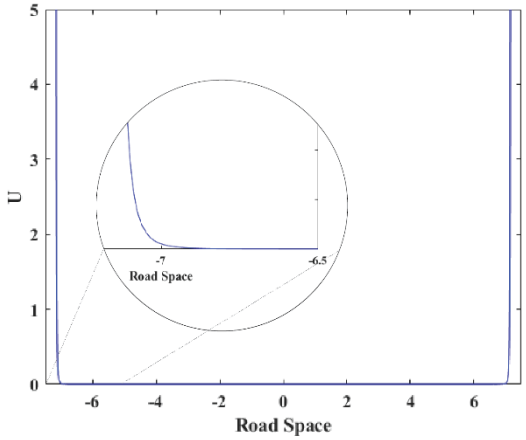


Figure 81: U Function

$$U'(y) = \begin{cases} 4\left(\frac{2y}{(a^2-y^2)^2}\right)\left(\frac{1}{a^2-y^2}-\frac{c}{a^2}\right)^3, & -a < y < -\frac{a\sqrt{c-1}}{\sqrt{c}} \; and \; \frac{a\sqrt{c-1}}{\sqrt{c}} < y < a \\ 0, & -\frac{a\sqrt{c-1}}{\sqrt{c}} \le y \le \frac{a\sqrt{c-1}}{\sqrt{c}} \end{cases}$$

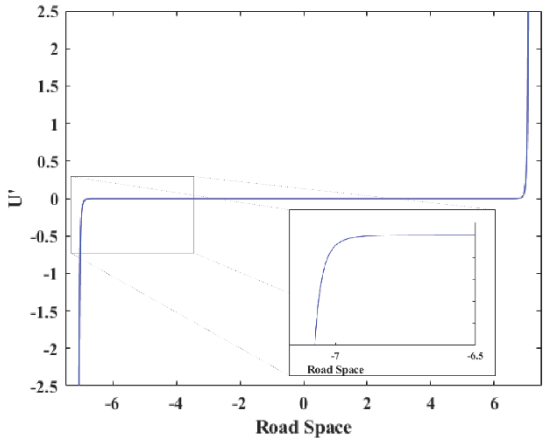


Figure 82: Derivative of U Function

For the U function we found from various trials that it will work better if we simplify the existing U by defining $c = 1$ and utilizing the Gaussian function twice, once for the negative parts and once for the positive parts. We use the following parameters for the Gaussian functions: $var_1 = 0.5$, $var_2 = -3$, $var_3 = 0.15$, $var_4 = var_1$, $var_5 = -var_2$, $var_6 = var_3$. Thus, the function we will investigate becomes as:

$$U_{new}(y) = \left(\frac{1}{a^2-y^2}-\frac{1}{a^2}\right)^4 + var_1 e^{-(y-var_2)^2/(2var_3^2)} + var_4 e^{-(y-var_5)^2/(2var_6^2)}, -a < y < a$$

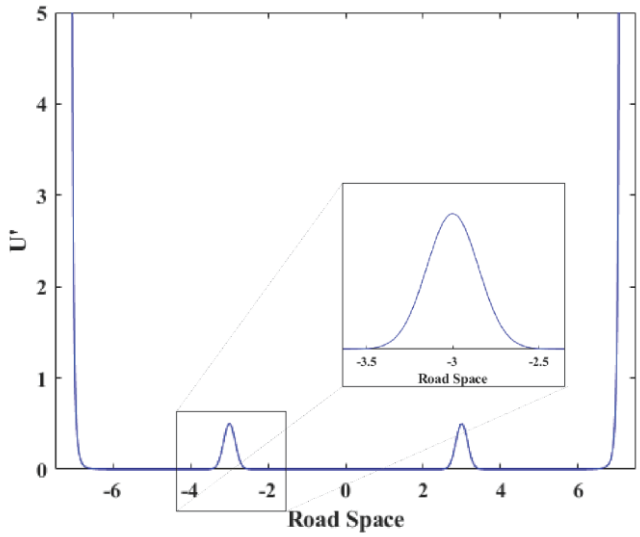


Figure 83: New U Function

$$U'_{new}(y) = 4\left(\frac{2y}{(a^2-y^2)^2}\right)\left(\frac{1}{a^2-y^2}-\frac{1}{a^2}\right)^3 - \frac{var_1(y-var_2)e^{-\frac{(y-var_2)^2}{(2var_3^2)}}}{(var_3^2)} - \frac{var_4(y-var_5)e^{-\frac{(y-var_5)^2}{(2var_6^2)}}}{(var_6^2)}, -a < y < a$$

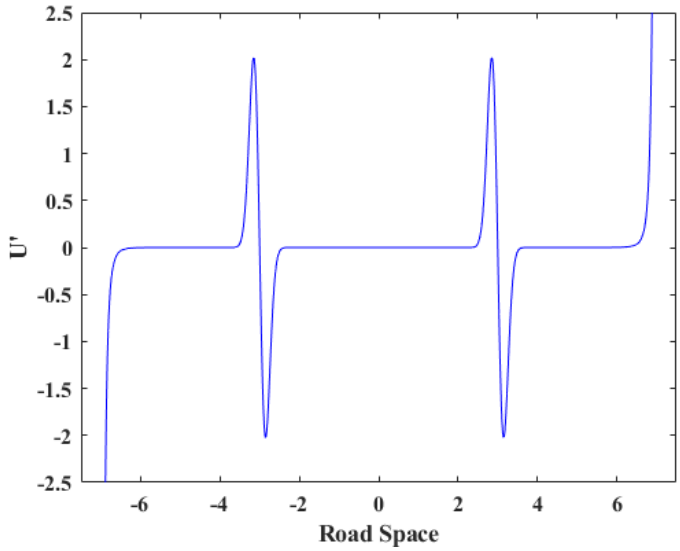


Figure 84: Derivative of New U function

As we can see from Figures 81 and 82, the new U function creates two new local minimums. This change leads to the creation of three lanes alongside the road and for the simulations we utilize the second set of parameters for 100 vehicles, for 100 and 500 seconds simulations. The results are presented below:

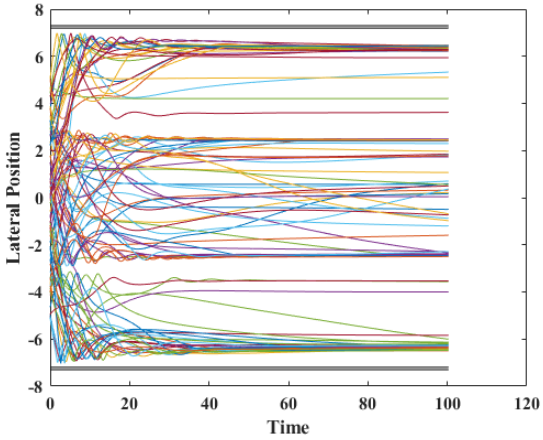


Figure 85: Trajectories of 100 seconds simulation

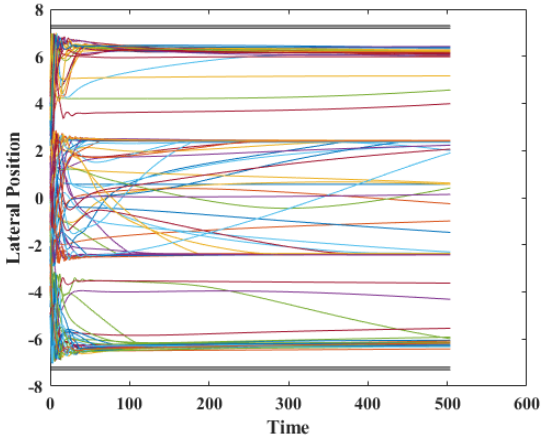


Figure 86: Trajectories of 500 seconds simulation

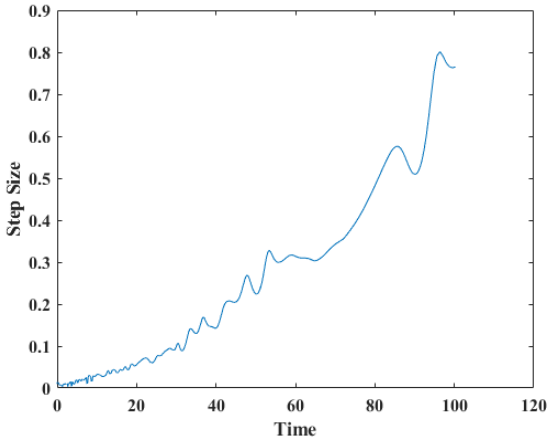


Figure 87: Step Size of 100 seconds simulation

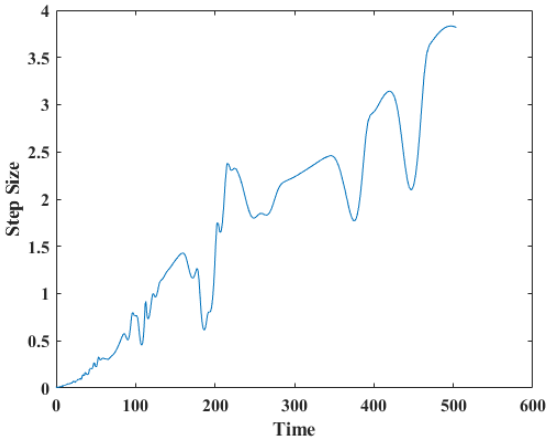


Figure 88: Step Size of 500 seconds simulation

The simulations needed 6.685 and 7.761 seconds respectively. Following, we present the results for the Adaptive Method through Systems Energy for 100 and 500 seconds simulations:

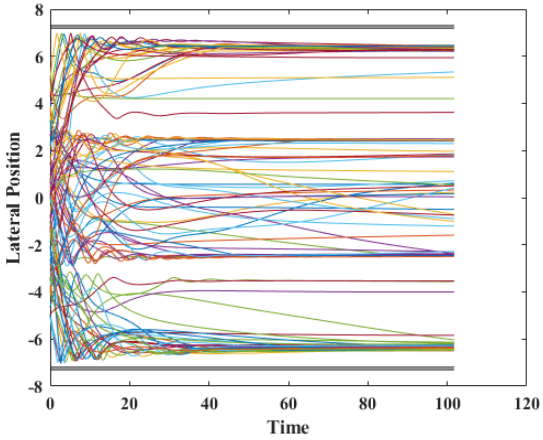


Figure 89: Trajectories of 100 seconds simulation

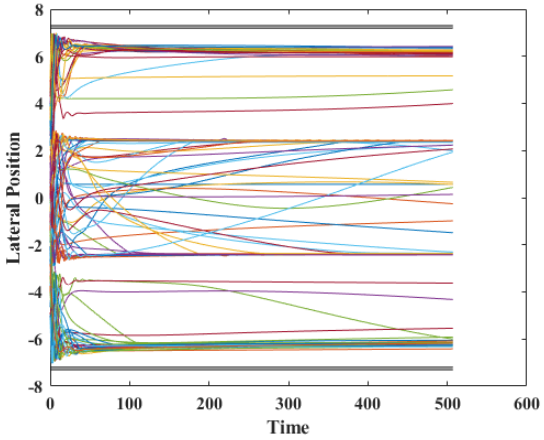


Figure 90: Trajectories of 500 seconds simulation

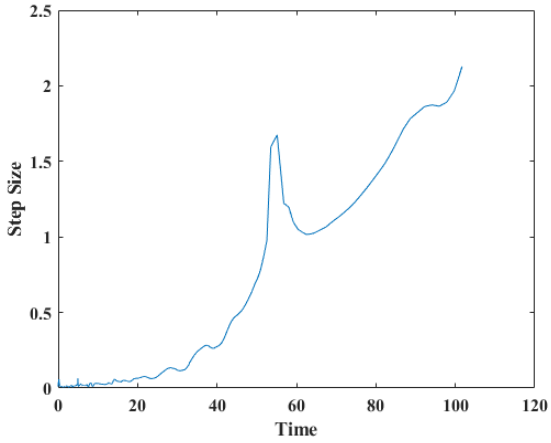


Figure 91: Step Size of 100 seconds simulation

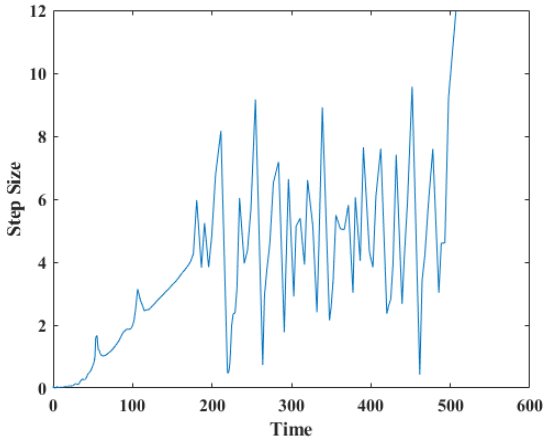


Figure 92: Step Size of 500 seconds simulation

The simulations needed 7.08 and 7.857 seconds respectively. From all the simulations performed we observed that the vehicles tend to the local minimum areas created by the new U Function. Depending on the design of the controller we can shape our system from lane-free to a lane-based model. For example if we change the power 4 to 1:

$$U_{new}(y) = \left(\frac{1}{a^2 - y^2} - \frac{1}{a^2}\right) + var_1 e^{-(y-var_2)^2/(2var_3^2)} + var_4 e^{-(y-var_5)^2/(2var_6^2)}, -a < y < a$$

$$U'_{new}(y) = \left(\frac{2y}{(a^2 - y^2)^2}\right) - \frac{var_1(y - var_2)e^{-\frac{(y-var_2)^2}{(2var_3^2)}}}{(var_3^2)} - \frac{var_4(y - var_5)e^{-\frac{(y-var_5)^2}{(2var_6^2)}}}{(var_6^2)}, -a < y < a$$

, and with small changes at the parameters of the Gaussian functions, $var_1 = 0.5$, $var_2 = -3$, $var_3 = \frac{1.5}{\sqrt{2}}$, $var_4 = var_1$, $var_5 = -var_2$, $var_6 = var_3$, we receive the following results:

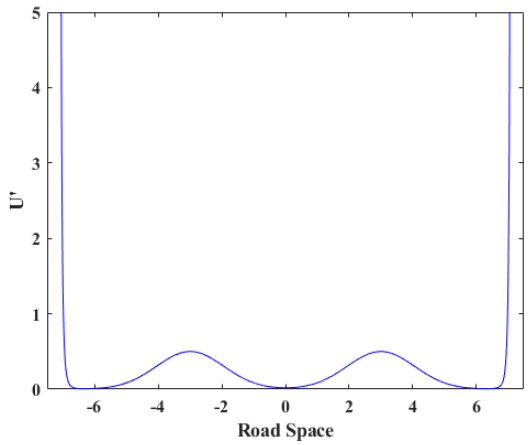


Figure 93: New U Function

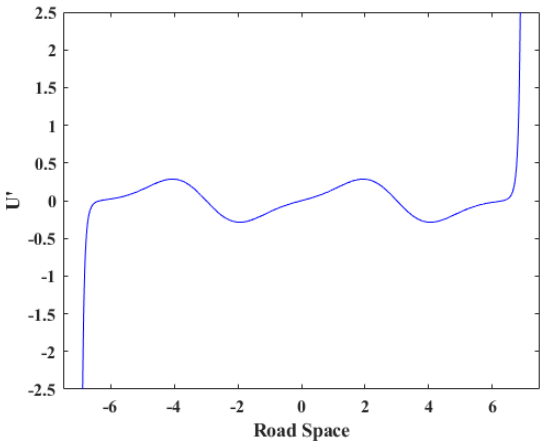


Figure 94: Derivative of New U Function

Next we will present the results for the Adaptive Method, the vehicles trajectories will be shown at Figure 95 and 96, with axes of time and lateral position, as also the step sizes are presented at Figures 97 and 98, for 100 and 500 second simulations, respectively.

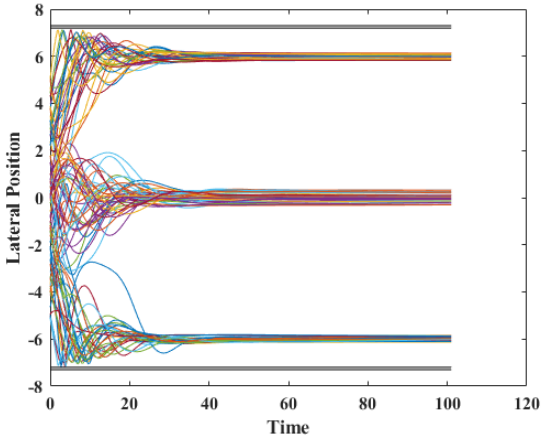


Figure 95: Adaptive Method Trajectories of 100 seconds simulation

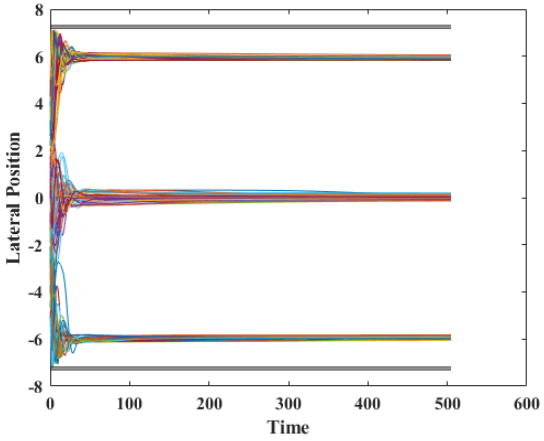


Figure 96: Adaptive Method Trajectories of 500 seconds simulation

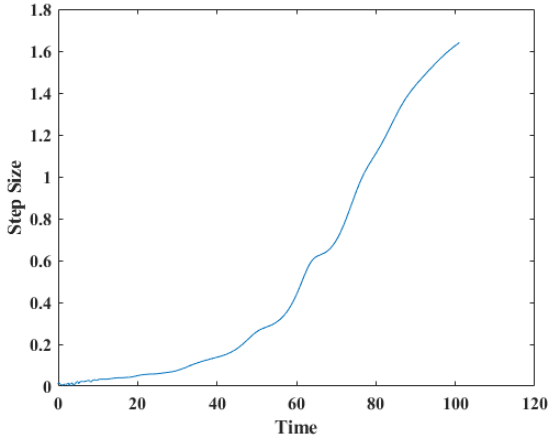


Figure 97: Adaptive Method Step Size of 100 seconds simulation

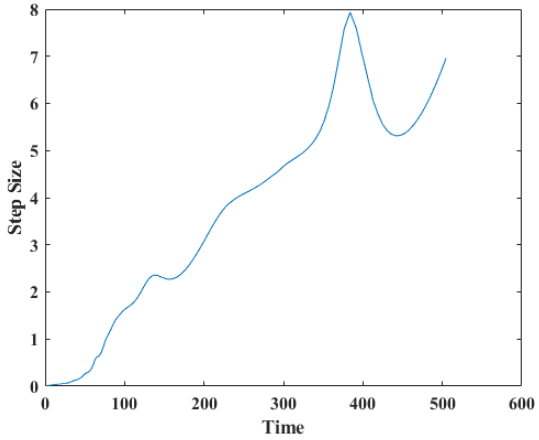


Figure 98: Adaptive Method Step Size of 500 seconds simulation

The simulations needed 6.677 and 7.061 seconds respectively. At Figures 99 to 102, are presented the results for the Adaptive Method through Systems Energy for 100 and 500 seconds simulations utilizing the new U Function.

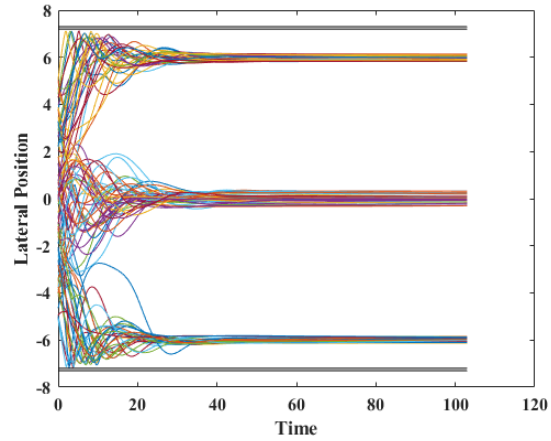


Figure 99: Adaptive Method through Systems Energy Trajectories for 100 seconds simulation

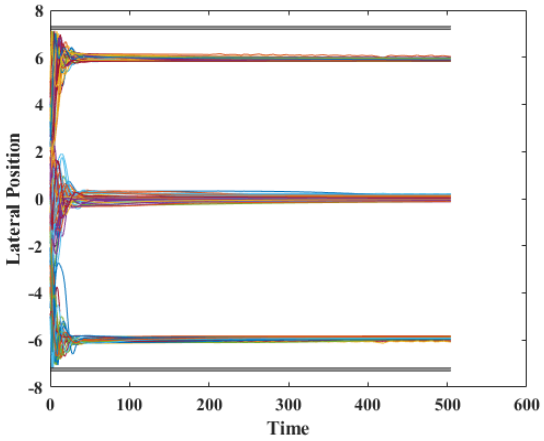


Figure 100: Adaptive Method through Systems Energy Trajectories for 500 seconds simulation

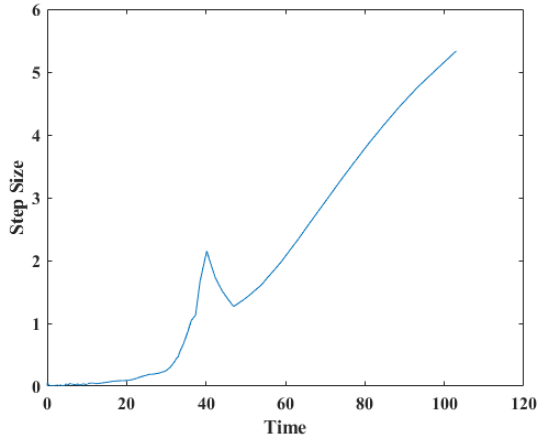


Figure 101: Adaptive Method through Systems Energy Step Size for 100 seconds simulation

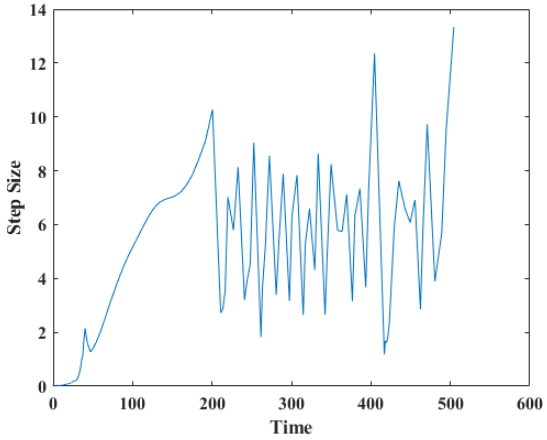


Figure 102: Adaptive Method through Systems Energy Step Size for 500 seconds simulation

The simulations needed 5.143 and 5.631 seconds respectively. Lastly, we have to investigate the new U and V Functions together and we will present results for the two different U Function set ups we have presented previously. At Figures 103 to 106 we present the results for the Adaptive Method with 8.379, 13.671, 8.898 and 12.865 seconds needed for the simulations, respectively.

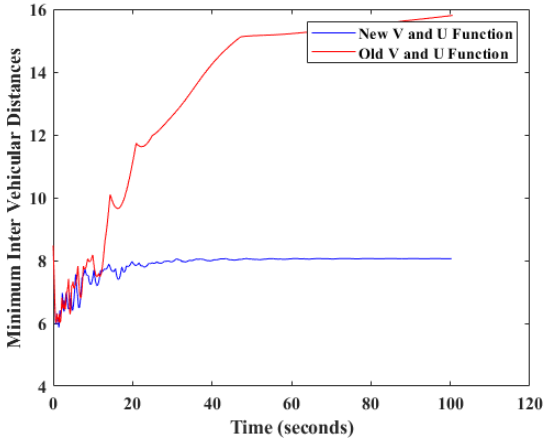


Figure 103: Adaptive Method 100 seconds simulation

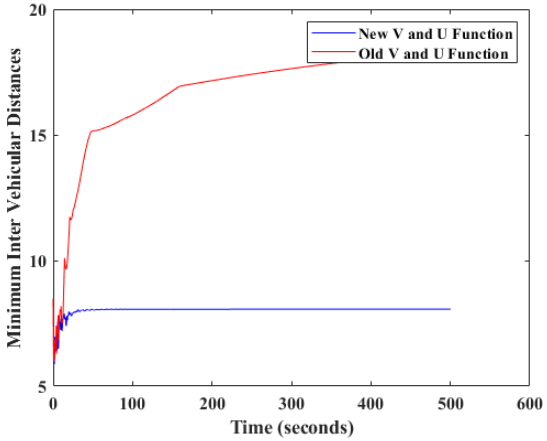


Figure 104: Adaptive Method 500 seconds simulation

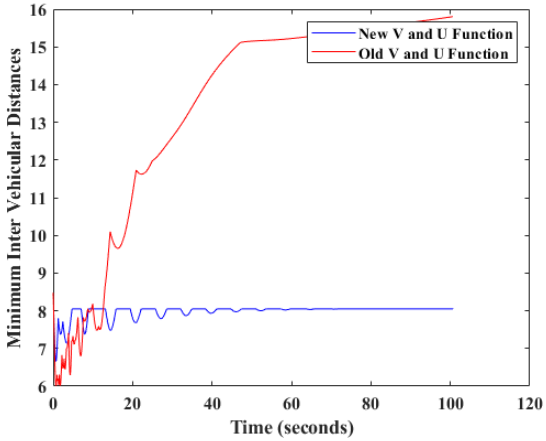


Figure 105: Adaptive Method 100 seconds simulation

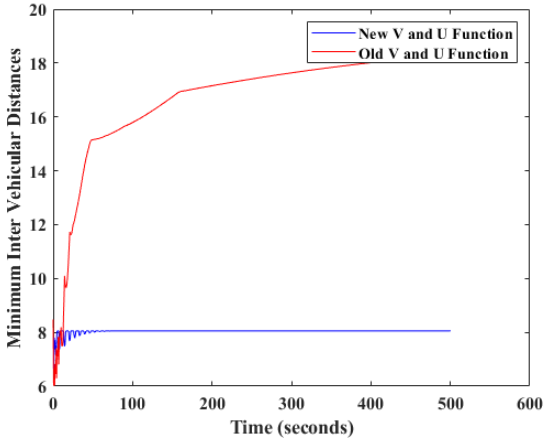


Figure 106: Adaptive Method 500 seconds simulation

At Figures 103 to 106, we present the results for the Adaptive Method through Systems Energy with 9.661, 18.12, 7.741 and 12.787 seconds needed for the simulations, respectively.

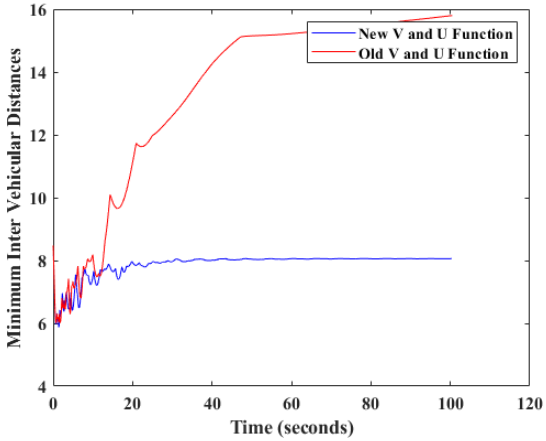


Figure 107: Adaptive Method through Systems Energy 100 seconds simulation

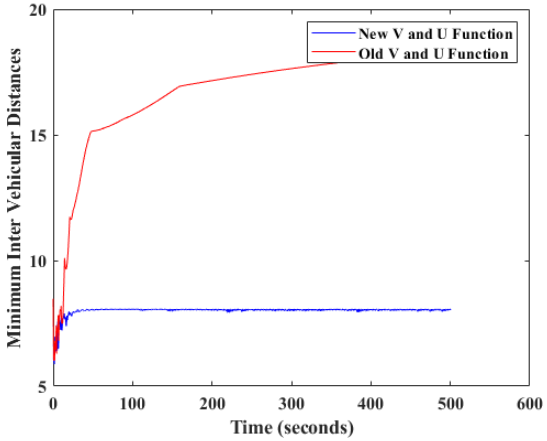


Figure 108: Adaptive Method through Systems Energy 100 seconds simulation

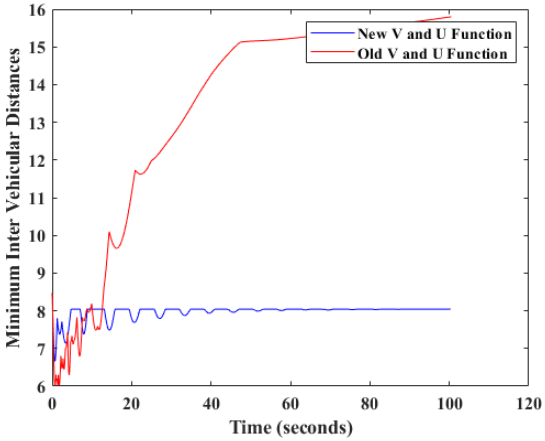


Figure 109: Adaptive Method through Systems Energy 100 seconds simulation

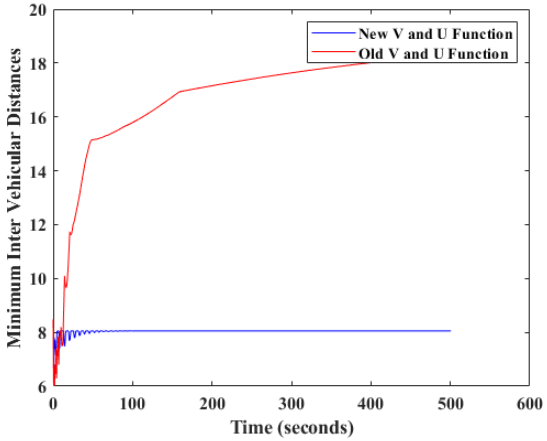


Figure 110: Adaptive Method through Systems Energy 500 seconds simulation

From the results presented above, we observe that even with the vehicles tending to multiple local minima, the new V Function makes the minimum inter-vehicular distances shorter. By forcing the U Function to transform our model into a lane-based one, we observe for both adaptive techniques that the vehicles reach the minimum inter-vehicular distances way faster. The new functions utilized and the results gained from various simulations are not the optimum and there is still potential for better potential repulsive functions. The investigation was performed in order to show that with small changes in the system's functions we can achieve different goals we may set.

References

[1] European Commission, Directorate-General for Mobility and Transport, (2021). *EU Transport in Figures: Statistical Pocketbook 2021*, *Publications Office.*

[2] "PHANTOM AUTO" WILL TOUR CITY. (1926, December 8). *The Milwaukee Journal Sentinel*.

[3] O'Toole, M., Lindell, D. & Wetzstein, G. Confocal Non-line-of-sight Imaging Based on the Light-cone Transform. *Nature* 555, 338–341 (2018). https://doi.org/10.1038/nature25489

[4] Yanumula, V. K., Typaldos, P., Troullinos, D., Malekzadeh, M., Papamichail, I., & Papageorgiou, M. (2021, September 19). Optimal Path Planning for Connected and Automated Vehicles in Lane-free Traffic. *2021 IEEE International Intelligent Transportation Systems Conference (ITSC).* https://doi.org/10.1109/ITSC48978.2021.9564698

[5] Mountakis, K. S., Karafyllis, I., Papamichail, I., & Wang, Y. (2021, February). Lane-Free Artificial-Fluid Concept for Vehicular Traffic. *Proceeding of the IEEE*, *109*(2), 114-121. https://doi.org/10.1109/jpoc.2020.3042681

[6] Karafyllis, I., Theodosis, D., Papageorgiou, M., (2022). Lyapunov Based Two-Dimensional Cruise Control of Autonomous Vehicles on Lane-Free Roads, *Automatica*, *145*. 110517. https://doi.org/10.1016/j.automatica.2022.110517

[7] Boyce, W. E., Diprima, R. C., (1997). *Elementary Differential Equations and Boundary Value Problems* (6th ed.). Wiley.

[8] Gupta, G. K., Sacks-Davis, R., Tischer, P. E., (1985). A Review of Recent Developments in Solving ODEs. *ACM Computing Surveys*, *17*(1), 5-47. https://doi.org/10.1145/4078.4079

[9] Canale, R. P., Chapra, S. C., (2016). *Numerical Methods for Engineers* (7th ed.). MC GRAW HILL

[10] Chapra, S. C., (2012). *Applied Numerical Methods with MATLAB for Engineers and Scientists* (3rd ed.). MC GRAW HILL

[11] LeVeque, R. J., (2007). *Finite Difference Methods for Ordinary and Partial Differential Equations: Steady-State and Time-dependent Problems* (1st ed.). Society for Industrial and Applied Mathematics.

[12] Hairer, E., Nørsett, S., P., Wanner, G., (1987) . *Solving Ordinary Differential Equations I: Nonstiff Problems.* Springer-Verlag Berlin and Heidelberg GmbH & Co. KG.

[13] Hairer, E., Lubich, C., Wanner, G., Berlin (2006). Numerical Integrators. *Geometric Numerical Integration: Structure-Preserving Algorithms for Ordinary Differential Equations*. Springer.

[14] Gaussian function., (2022,July 26). In Wikipedia. https://en.wikipedia.org/wiki/x Gaussian_function

[15] Quarteroni, A., Sacco, R., & Saleri, F. (2007). Numerical Solution of Ordinary Differential Equations. *Numerical Mathematics*, 37, 479-483. Springer.

[16] Stuart, A., M., Humphries, A., R., (1998*).* Numerical Methods for Initial Value Problems. *Dynamical Systems and Numerical Analysis*, 212-217. Cambridge University Press.

[17] Karafyllis, I., Grüne, L., (2011). Feedback Stabilization Methods for the Numerical Solution of Ordinary Differential Equations. *Discrete &Amp; Continuous Dynamical Systems – B*, *16*(1), 283-317. http://dx.doi.org/10.3934/dcdsb.2011.16.283

[18] Faccio, D., Velten, A., Wetzstein, G., (2020). Non-line-of-sight Imaging. *Nature Reviews Physics*, *2*(6), 318-327. https://doi.org/10.1038/s42254-020-0174-8

[19] Shampine, L. F., Gear, C. W., (1979). A User's View of Solving Stiff Ordinary Differential Equations. *SIAM Review*, *21*(1), 1-17. https://doi.org/10.1137/1021001

[20] Krogh, F. T., (1973). Algorithms for Changing the Step Size. *SIAM Journal on Numerical Analysis*, 10(5), 949–965. https://doi.org/10.1137/0710081

[21] Cooper, G. J., (1971). Error Bounds for Numerical Solutions of Ordinary Differential Equations. *Numerische Mathematik*, 18(2), 162–170. https://doi.org/10.1007/bf01436325

[22] Shampine, L. F., & Baca, L. S., (1984). Error Estimators for Stiff Differential Equations. *Journal of Computational and Applied Mathematics*, *11*(2), 197–207. https://doi.org/10.1016/0377-0427(84)90020-7

[23] Shampine, L. F., & Watts, H. A., (1976). Global Error Estimates for Ordinary Differential Equations. *ACM Transactions on Mathematical Software*, *2*(2), 172–186. https://doi.org/10.1145/355681.355687

[24] Shampine, L., (1986). Global Error Estimation with One-step Methods. *Computers & Mathematics with Applications*, *12*(7), 885–894. https://doi.org/10.1016/0898-1221(86)90032-5

[25] Fehlberg, E., (1969*). Low-Order Classical Runge-Kutta Formulas with Stepsize Control and their Application to some Heat Transfer Problems*. Washington: National Aeronautics and Space Administration.

[26] Fehlberg, E., (1964). New High-Order Runge-Kutta Formulas with Step Size Control for Systems of First-and Second-Order Differential Equations. *ZAMM - Journal of Applied Mathematics and Mechanics / Zeitschrift Für Angewandte Mathematik Und Mechanik*, *44*(S1). https://doi.org/10.1002/zamm.19640441310

[27] Karafyllis, I., Theodosis, D., Papageorgiou, M., (2022). Analysis and Control of a Non-local PDE Traffic Flow Model. *International Journal of Control*, *95*(3), pp. 660-678. https://doi.org/10.1080/00207179.2020.1808902

[28] Chavoshi, K., Kouvelas, A., Cooperative Distributed Control for Lane-less and Direction-less Movement of Autonomous Vehicles on Highway Networks. *(2020) - 9th Symposium of the European Association for Research in Transportation*, 2021.

[29] Diakaki, C., Papageorgiou, M., Papamichail, I., Nikolos, I., (2015). Overview and Analysis of Vehicle Automation and Communication Systems from a Motorway Traffic Management Perspective. *Transportation Research Part A: Policy and Practice, 75*, 147-165. https://doi.org/10.1016/j.tra.2015.03.015

[30] Karafyllis, I., Theodosis, D., Papageorgiou, M., (2020). Nonlinear Adaptive Cruise Control of Vehicular Platoons. *International Journal of Control*, 1-23. https://doi.org/10.1080/00207179.2021.1982015

[31] Malekzadeh, M., Papamichail, I., Papageorgiou M., Bogenberger, K., (2021). Optimal Internal Boundary Control of Lane-Free Automated Vehicle Traffic. *Transportation Research Part C: Emerging Technologies, 126*, 103060. https://doi.org/10.1016/j.trc.2021.103060

[32] Polack, P., Altché, F., d'Andréa-Novel B., de La Fortelle, A., (2017). The Kinematic Bicycle Model: A Consistent Model for Planning Feasible Trajectories for Autonomous Vehicles?. *IEEE Intelligent Vehicles Symposium (IV)*, 812-818. https://doi.org/10.1109/IVS.2017.7995816

Appendix A

Pseudocode

WHILE $t_z \leq end$

FOR $i \leftarrow 1\ to\ N$

FOR $j \leftarrow 1\ to\ N$

IF $i \neq j$

CALCULATE $dis_{i,j}$, distance between pairs of vehicles

CALCULATE $\dot{V}_{i,j}$, differential of the repulsive potential function V

CALCULATE $\sum_{j \neq i} V_{d_{i,j}} \frac{(x_i - x_j)}{d_{i,j}}, \sum_{j \neq i} V_{d_{i,j}} \frac{(y_i - y_j)}{d_{i,j}}$

ENDIF

ENDFOR

CALCULATE k, F, u

CALCULATE $vk_1, \theta k_1, yk_1, xk_1$

CALCULATE $xEuler, yEuler, \theta Euler, vEuler$

ENDFOR

FOR $i \leftarrow 1\ to\ N$

FOR $j \leftarrow 1\ to\ N$

IF $i \neq j$

CALCULATE $dis_{i,j}$, distance between pairs of vehicles using $xEuler$, $yEuler$

CALCULATE $\dot{V}_{i,j}$, differential of the repulsive potential function V

CALCULATE $\sum_{j \neq i} V_{d_{i,j}} \frac{(x_i - x_j)}{d_{i,j}}, \sum_{j \neq i} V_{d_{i,j}} \frac{(y_i - y_j)}{d_{i,j}}$

ENDIF

ENDFOR

CALCULATE k, F, u, using Euler positions

CALCULATE $vk_2, \theta k_2, yk_2, xk_2$

CALCULATE $x_{z+1}, y_{z+1}, \theta_{z+1}, v_{z+1}$

CALCULATE $sc_{x_i}, sc_{y_i}, sc_{v_i}, sc_{\theta_i}$

CALCULATE ERR_i

IF $x_{z+1}, y_{z+1}, \theta_{z+1}, v_{z+1} \notin \Omega$

SET $check = 1$

ENDIF

IF $i = N$

CALCULATE $error = norm2(ERR)$

IF $error \leq tolerance$ AND $check = 0$

SET $z \leftarrow z + 1$

SET $t_z \leftarrow t_{z-1} + h_{z-1}$

SET $h_z \leftarrow h_{z-1} \min \left(facmax, \max \left(facmin, fac\sqrt{\frac{1}{err}} \right) \right)$

ELSEIF $error > tolerance$ AND $check = 0$

SET $h_z \leftarrow h_z \min \left(facmax, \max \left(facmin, fac\sqrt{\frac{1}{err}} \right) \right)$

ELSEIF $check = 1$

SET $h_z \leftarrow \frac{h_z}{2}$

ENDIF

ENDIF

ENDFOR

ENDWHILE

Appendix B

100 vehicles

Set number 2

Vehicle Number	theta	velocity	lateral position	longitudinal position
1	0.009461077	28.46176834	2.795677104	13.95922391
2	0.022274153	32.33249443	0.469007119	21.72867077
3	-0.012827588	33.25486511	2.93900201	34.0442609
4	0.037624081	31.93664185	-0.810419402	34.16065904
5	-0.030685303	30.22478331	3.24022497	52.54079626
6	-0.012732311	32.86988625	-3.118374889	80.00985026
7	0.0045815	31.3066935	-0.113838372	85.52954368
8	0.038564669	32.68250288	3.698141376	87.14234151
9	0.015381691	31.54133294	-3.405901769	90.49241862
10	-0.024057444	28.56636128	2.384259538	102.4482315
11	0.013315795	27.1039155	3.999768156	128.1741163
12	0.015476985	30.30929262	-3.806895361	131.5693419
13	-0.036976015	30.53043449	3.142231273	143.1164595
14	-0.03358671	28.04521125	-5.025722004	147.5613519
15	0.000337534	31.21774011	-2.354352534	153.7180209
16	0.008939166	28.3489499	-1.651810968	162.9136272
17	-0.036682388	26.47050382	-3.544150308	170.3733734
18	0.010498578	28.52814063	0.247345904	175.7385916
19	-0.018129894	32.89376503	-1.195201635	191.1193512
20	-0.039574151	29.30102086	0.173286552	200.6381921
21	-0.005595038	27.54790138	4.446096428	201.5595497
22	-0.028703245	30.46769479	-4.982232146	211.0344743
23	-0.024774765	31.7394609	1.779003384	216.1981843
24	0.015012413	29.04751892	4.632856817	222.1671252
25	0.013612233	34.97679709	-3.394231488	223.1598483
26	0.003426608	33.95185435	-3.345326507	240.1409189
27	-0.000757752	34.50953834	1.317700778	244.8656982
28	-0.041221013	34.56250569	-1.262033316	252.0324418
29	0.032771144	26.20804273	-1.911199701	277.3921721
30	-0.015325108	30.16033988	3.229888802	278.1915658
31	0.012775861	32.37895549	3.182722905	331.4793018
32	0.026598344	33.94214697	0.637952314	358.6830622
33	0.036172517	27.14864803	4.440040607	362.7295813
34	0.025893456	28.9951006	0.411519326	371.5525273

35	-0.010455588	28.56656054	-4.093874644	383.9117973
36	0.026505822	28.05802393	2.209049634	391.8493022
37	-0.039547586	30.99978352	3.787452843	402.1479023
38	-0.025333	30.22836438	4.647419916	419.7258383
39	0.015569399	28.38106041	0.042079677	424.5242137
40	0.010340538	34.41965507	-3.863862894	429.7852466
41	0.031108043	34.71996363	-1.11747964	439.0774938
42	-0.003695277	27.94150924	2.045283169	450.1946863
43	0.034512677	27.45733356	2.633016506	477.0930911
44	-0.031460643	30.10480597	-3.686900399	483.1783448
45	-0.02537503	28.29339164	3.873198531	492.8979593
46	0.033253046	28.39339712	-0.848912111	498.7759953
47	-0.014815582	34.91717151	-1.072374817	512.7203695
48	-0.025891872	31.03163141	-2.399110921	541.6150526
49	-0.019800393	33.63026149	0.682216374	560.46278
50	0.003551556	28.93342377	-0.456957341	568.6998117
51	-0.036594849	28.11848866	5.019482487	583.9593143
52	0.016927012	32.53857002	0.702893406	585.6881804
53	0.011929654	27.23939207	0.48650985	596.5249496
54	0.014302297	31.30122553	-3.902969666	598.8613993
55	0.008875057	32.16820464	0.198793106	609.2554358
56	-0.011141705	34.46574392	-2.651047145	620.773699
57	0.002830257	33.11849695	2.369254421	625.8216007
58	-0.016155452	27.31605581	-1.57600981	634.6558945
59	-0.026749175	32.82928988	3.469896873	641.2763575
60	0.030840708	29.66690079	-1.502507104	648.8679973
61	-0.004125354	29.25953269	4.736085731	699.921544
62	0.005146043	34.09418776	0.477316119	717.0151038
63	0.016201032	30.11609372	0.966775102	733.3479135
64	0.024664775	27.50208778	-4.639215742	743.6906706
65	0.02570284	29.10010234	-3.489477725	752.5073116
66	0.037252594	26.45355505	-0.02124958	769.6165794
67	0.017229072	26.84227613	2.609282177	779.2375771
68	0.019238659	27.47312731	-0.236263475	786.4103211
69	-0.026215641	31.68635192	-5.078142099	798.134031
70	0.033152759	32.68985521	-1.071410071	807.8979338
71	-0.002627178	33.01299337	-0.57672703	819.849151
72	0.034963998	30.68727506	-4.008978406	832.9818901
73	0.003545107	26.2017949	1.764612951	834.7894176
74	-0.030110829	31.39544601	1.496887757	843.6605763
75	0.031564554	29.04350005	2.900200018	866.6531158
76	0.021231727	33.05945149	-3.741407197	869.9684439

77	0.023188157	34.06285315	4.287692921	875.6433911
78	-0.022284793	29.83315816	-2.177158872	884.5514494
79	-0.031201653	34.98330517	-1.168379112	894.200159
80	0.02471655	33.4769165	-0.599870909	907.7140815
81	0.033258962	31.65957455	0.731629081	918.5311996
82	-0.012583091	33.0901342	-3.354346104	954.4991991
83	0.003260734	34.84547124	-2.36137265	963.2559502
84	-0.008431012	27.61389857	2.833205277	969.411005
85	-0.034512555	29.89965471	-2.613563845	971.8354058
86	0.025043853	29.74180486	-0.085630639	986.916977
87	-0.021593713	34.67527572	3.148651564	999.4958746
88	-0.003263875	26.96795525	-1.723239876	1016.304934
89	-0.018415623	33.30135963	-3.990598215	1027.437369
90	-0.034303634	30.13573462	0.318075104	1037.118239
91	0.007374042	31.63056248	-4.946444149	1043.86597
92	-0.042440486	33.08475736	-2.064677993	1053.545217
93	-0.022533052	34.32231695	0.480410417	1060.95216
94	0.042341403	34.35002449	5.044380789	1070.330696
95	0.021214642	34.91061153	-0.031039369	1080.77797
96	-0.035266378	34.28305102	-1.399007778	1096.595876
97	-0.036982204	29.88015236	-5.097711134	1103.095748
98	0.02517868	32.99903782	0.116810275	1106.610037
99	-0.042525563	29.77300977	3.279982584	1113.377716
100	-0.024458062	31.97467448	-2.813049741	1118.725897