Mathematical Modelling of Engineering Problems
Vol. 13, No. 2, February, 2026, pp. 441-452
Journal homepage: http://iieta.org/journals/mmep

Computation of Feasible Commands for Route Planning of a Three-Wheeled Autonomous Vehicle Robot

Masiala Mavungu*, Daniel Mashao

Faculty of Engineering and Built Environment, University of Johannesburg, Johannesburg PO Box 524, South Africa

Corresponding Author Email: masmavp@yahoo.fr

https://doi.org/10.18280/mmep.130220

Received: 20 October 2025
Revised: 13 December 2025
Accepted: 28 December 2025
Available online: 15 March 2026

Keywords:
autonomous vehicle, control theory, intelligent system, mobile robot, nonholonomic constraint, performance evaluation, three-wheeled vehicle

ABSTRACT

There is considerable innovation in autonomous vehicles and robotics. The Internet of Vehicles is impacting the world and creating many opportunities. This study aims to determine feasible commands and system responses that bring a three-wheeled autonomous vehicle from a prescribed starting point to a final point while minimizing energy. Mathematical models (in the form of a nonlinear system of five ordinary differential equations) are used to represent the vehicle. The total running cost of energy is defined by a functional integral, and then the problem is reformulated as an optimal control problem. Computer programs are developed to efficiently solve the fused control-free state-costate system, and then computational simulations are produced to demonstrate the effectiveness of the approach. The main results include the feasible commands (which serve as controls), the feasible system response defined by the feasible states, from which the feasible trajectory and the feasible speed functions are derived. Such results enable the industrial robot designers and managers to predict the trajectory of the robot based on the given commands, and to study the vehicle's performance for rational and well-informed decision-making.

1. INTRODUCTION

Currently, there is significant and substantial growth in the fields of computer and wireless networking, as well as automation and mechatronics. The technologies of the Internet of Things, robotics, automation, and driverless vehicles are making major contributions to industrial and human development overall, creating numerous research opportunities.

This paper is related to the fields of robotics, artificial intelligence, signal processing, image processing, computer vision, nonlinear dynamic optimization, optimal control of a dynamical system, engineering management, etc. It is robotics because it deals with a three-wheeled autonomous vehicle, which is a robot, an intelligent dynamical system built and managed. It involves significant and considerable mechanics and electronics. His motion is subject to obstacle avoidance which involves signal and image processing and analysis as well as computer vision by the sensor. It is related to optimal control and nonlinear optimization because the motion characteristics and the optimal path are mostly computed and predicted. The feasible commands, the resulting system response functions defined by the states, are computed such that a performance index is optimized. Most of time the aim is to find the optimal trajectory, to compute the optimal states in general, to minimize the total travelled distance, to find the shortest path, to minimize the total running cost, to minimize the total duration, to maximize utility for the robot's user, to optimize any other function. The obtained results of computations cause the industrial robot designers and managers to make rational and judicious decisions.

This paper focuses on modelling and controlling a three-wheeled autonomous vehicle robot and then planning its path. The objective is to compute feasible control strategies to drive the vehicle from a specified initial state to a final such that the running energy is minimized. The running energy is defined by a functional integral whose integrand is the rate of consumption of energy.

It is motivated by the fact that human beings are subject to complex tasks to perform manually and under very constrained deadlines. In order to minimize human effort in accomplishing certain tasks, many complex industrial tasks are robotized. Once a robot is built, industrial work can be planned and performed, and scientific and academic research projects can be elaborated.

This paper's methodology is the development of mathematical models and computational simulations. Mathematical models are developed to formulate, describe clearly, and to analyze the problem, while computational simulations are developed to enable summarizing and interpreting the results convincingly and concisely. The analytical solutions of the problem, as formulated, are not easy to obtain, so numerical solutions are the ones the study focuses on.

The main contributions are the developed mathematical models to describe and analyze the kinematics of the vehicle, the derivation of feasible commands considered as control strategies for the autonomous vehicle, the derivation of the

system responses defined by the state and costate functions, the development of computer programs written in Octave/MATLAB, the development of two versions of computer programs written in Scilab to confirm or question the results obtained with Octave/MATLAB, and all the resulting computational simulations. This paper proposes to the readers a way of developing the kinematic model of a three-wheeled autonomous vehicle. Such a kinematic model is developed in terms of a system of five Ordinary Differential Equations (ODEs) with five variables for the state controlled by three commands. The system of equations is vectorized to enable the development of a MATLAB function or a function under any other similar matrix laboratory computer programming language. Managerially, this paper allows the readers to plan and analyze the motion of a three-wheeled autonomous vehicle and that of any autonomous vehicle in an optimal manner. It allows industrial operators, managers, and any other decision makers to act intelligently and reliably. It provokes scientific and academic leaders to dream as much as possible in the development of scientific and academic research projects. Works in the bibliography are based on autonomous vehicles in general and on three-wheeled autonomous vehicles in particular. Pandey et al. [1] designed and implemented a Proportional Integral and Derivative (PID) controller to control the speed in a digital control framework. Chen et al. [2] designed a trajectory controller based on a predictive control model based for trajectory tracking situations in narrow corridors. Alfiany et al. [3] developed and analyzed a kinematic and a dynamic model of an autonomous pedicab. Batti et al. [4] developed and implemented Fuzzy control to monitor a non-holonomic unicycle mobile robot. Alfiany et al. [5] proposed a new design of an autonomous vehicle, Becak. Alghanim et al. [6] modelled and controlled a skid-steering mobile robot, Argo J5, with a custom-built landing platform. Alghanim [7] developed a dynamic model and implemented many controllers for the trajectory following of the Argo 5. Mishra et al. [8] evaluated and emulated the movement of the vehicle in a 2-dimensional plane. The study of Mellah et al. [9] was based on the supervision of the health state of the vehicle actuators after a first detection of faults had been performed. Saeedi and Kazemi [10] developed novel methods for asymptotically tracking a prescribed path. Saypulaev et al. [11] presented a mathematical model of the spatial kinematics of an omnidirectional wheel, symmetrically located on a circle. Xu et al. [12] presented a universal kinematics model for a robot with four caster cambered wheels. Xue and Chen [13] focused on control-related topics. Trojnacki [14] performed an analysis of the problem of modelling the dynamics of a Three-Wheeled mobile robot with a front wheel driven and steered. Bin et al. [15] described the kinematics model of a Two-Wheeled self-balancing autonomous mobile robot. Li et al. [16] discussed the mechanical design and dynamic modeling of this type of mobile robot platform. Indiveri [17] addressed the issue of kinematics modeling, singularity analysis, and motion control for a generic vehicle equipped with N Swedish wheels. Mellah et al. [18] proposed an approach for actuator fault detection and isolation in a four-mechanism wheeled mobile robot. Koubaa et al. [19] illustrated trajectory tracking control of a nonholonomic system, which is a mobile robot. Sofwan et al. [20] presented the development of an omni-wheeled mobile robot based on inverse kinematics and Odometry for local and indoor navigation purposes, such as for automatic warehousing in industry or healthcare environments. Labakhua et al. [21] proposed a motion control study of a mobile robot manipulator, provided with Swedish wheels. Pei and Kleeman [22] introduced a new fourth parameter that accounts for wheel slip proportional to the linear acceleration of the robot. This study, based on the study of Alfiany et al. [5], is organized in the following manner:

The development of mathematical models for the objective functional and for the robot's kinematics is performed in Section 2. Such a robot diagram is given in Figure 1. The diagram in which are shown its components is represented by Figure 2. The computation of the system Hamiltonian is performed in Section 3. The derivation of the normal equations of optimality, the computation of feasible controls, and the derivation of Optimality Conditions are performed in Section 4. The vectorization of the combined state-costate system is performed in Section 5. Computational simulations are provided in Section 6.

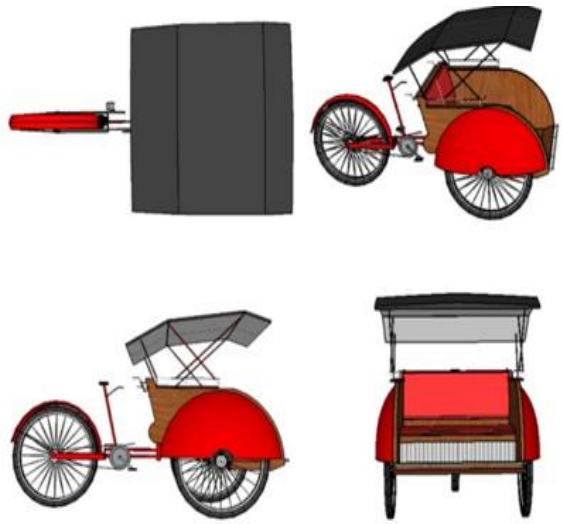

Figure 1. The appearance of Indonesia traditional pedicab, called Becak

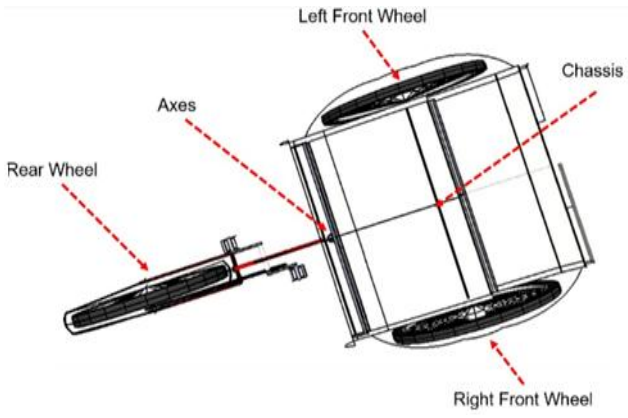

Figure 2. Component of Becak

2. MATHEMATICAL MODELS

2.1 Kinematic control system

The three-wheeled vehicle is as follows:

$$\frac{dx}{dt} = c_1 \omega \cos(\theta + \beta) \quad (1)$$

$$\frac{dy}{dt} = c_1 \omega \sin(\theta + \beta) \quad (2)$$

$$\frac{d\theta}{dt} = c_2 \omega \cos(\beta) \tan(\gamma) \quad (3)$$

where, (x, y) is the position of the robot in the xy horizontal plane, θ is the angle defined by the vehicle direction and the x axis, $(\theta + \beta)$ is the course separating the x axis and the direction of the robot. The angles for steering and slipping, defining a closed-loop system, are as follows:

$$\frac{d\beta}{dt} = -b\beta + b\mu \quad (4)$$

$$\frac{d\gamma}{dt} = -c\gamma + c\tau \quad (5)$$

where, γ is the steering angle, $c_1 = R$ and $c_2 = \frac{R}{L}$ are constants of proportionality where R and L are respectively the radius of each wheel and the length of the robot. c_1 and c_2 are directly proportional to the radius of the wheels and inversely proportional to the length of the axle. The length of the axle is the distance between the front wheels.

2.2 Objective functional

$$J(\omega, \mu, \tau) = \int_{t_0}^{t_f} (\omega^{`2} + \mu^2 + \tau^2) dt \quad (6)$$

where, t_0 and t_f are respectively the lower and upper bounds of the time interval of the vehicle motion, ω, μ, and τ are respectively the heading, the slip, and the steering angular velocities of the three-wheeled vehicle. Such commands are used as control variables.

2.3 Problem statement

Consider the kinematic model of the three-wheeled vehicle given by Eqs. (1)-(5) and the functional integral representing the running energy cost, the aim is to compute feasible commands to bring the system from a starting point to a final point while minimizing the performance criteria.

3. HAMILTONIAN, NORMAL EQUATIONS AND EFFECTIVE COMMANDS

To minimize the objective functional, it will suffice to minimize the Hamiltonian defined as follows:

$$H = h(\omega, \mu, \tau) + \sum_{k=1}^{5} \alpha_k f_k(t, x, y, \theta, \beta, \gamma, \omega, \mu, \tau) \quad (7)$$

where,

$h(\omega, \mu, \tau) = \omega^2 + \mu^2 + \tau^2$ is the rate at which the energy consumed by the vehicle.

$f_1(t, x, y, \theta, \beta, \gamma, \omega, \mu, \tau) = c_1 \omega \cos(\theta + \beta)$ is the right-hand side of Eq. (1).

$f_2(t, x, y, \theta, \beta, \gamma, \omega, \mu, \tau) = c_1 \omega \sin(\theta + \beta)$ is the right-hand side of Eq. (2), etc.

$$f_3(t, x, y, \theta, \beta, \gamma, \omega, \mu, \tau) = c_2 \omega \cos(\beta) \tan(\gamma).$$

$$f_4(t, x, y, \theta, \beta, \gamma, \omega, \mu, \tau) = -b\beta + b\mu$$

$$f_5(t, x, y, \theta, \beta, \gamma, \omega, \mu, \tau) = -c\gamma + c\tau$$

The feasible commands are solutions to the following equations:

$$\begin{aligned} \frac{\partial H}{\partial \omega} &= 2\omega + \alpha_1(c_1 \cos(\theta + \beta)) \\ &+ \alpha_2(c_1 \sin(\theta + \beta)) \\ &+ \alpha_3(c_2 \cos(\beta) \tan(\gamma)) = 0 \end{aligned} \quad (8)$$

$$\frac{\partial H}{\partial \mu} = 2\mu + b\alpha_4 = 0 \quad (9)$$

$$\frac{\partial H}{\partial \tau} = 2\tau + c\alpha_5 = 0 \quad (10)$$

The feasible controls are as follows:

$$\begin{aligned} \omega^* = -0.5(&\alpha_1(c_1 \cos(\theta + \beta)) \\ &+ \alpha_2(c_1 \sin(\theta + \beta)) \\ &+ \alpha_3(c_2 \cos(\beta) \tan(\gamma))) \end{aligned} \quad (11)$$

$$\mu^* = -0.5 b \alpha_4 \quad (12)$$

$$\tau^* = 0.5 c \alpha_5 \quad (13)$$

4. DERIVATION OF FIRST-ORDER CONDITIONS

If triplet $(\omega^*, \mu^*, \tau^*)^T$ is the optimal command of the above problem, and that $(x^*, y^*, \theta^*, \beta^*, \gamma^*)^T$ is the corresponding response from the system, then a costate vector exists, denoted $\boldsymbol{\alpha} = (\alpha_1, \alpha_2, \alpha_3, \alpha_4, \alpha_5)^T$ such that

$$\frac{d\alpha_1}{dt} = -\frac{\partial H}{\partial x^*} = 0 \quad (14)$$

$$\frac{d\alpha_2}{dt} = -\frac{\partial H}{\partial y^*} = 0 \quad (15)$$

$$\begin{aligned} \frac{d\alpha_3}{dt} &= -\frac{\partial H}{\partial \theta^*} \\ &= c_1 \omega^* (\alpha_1 \sin(\theta^* + \beta^*) - \alpha_2 \cos(\theta^* + \beta^*)) \end{aligned} \quad (16)$$

$$\begin{aligned} \frac{d\alpha_4}{dt} = -\frac{\partial H}{\partial \beta^*} &= c_1 \omega^* (\alpha_1 \sin(\theta^* + \beta^*) \\ &- \alpha_2 \cos(\theta^* + \beta^*)) \\ &+ c_2 \alpha_3 \omega^* \sin(\beta) \tan(\gamma) + b\alpha_4 \end{aligned} \quad (17)$$

$$\frac{d\alpha_5}{dt} = -\frac{\partial H}{\partial \gamma^*} = c\alpha_5 - c_2 \alpha_3 \omega^* \cos(\beta^*) \sec^2(\gamma^*) \quad (18)$$

$$J(\omega^*, \mu^*, \tau^*) \leq J(\omega, \mu, \tau) \text{ for } (\omega, \mu, \tau) \in \mathbb{R}^3 \quad (19)$$

5. VECTORIZATION OF THE COMBINED CONTROLLED STATE-COSTATE SYSTEM

The computer programming language that is going to be used is a matrix-based programming language. To optimize the processing time and the management of the memory in handling the system of ODEs and all the other involved stuff, it is important to redefine the joined state and costate system of ODEs either as a vector and/or as a matrix.

By letting $u^* = [\omega^*, \mu^*, \tau^*]^T$, $Y = [x^*, y^*, \theta^*, \beta^*, \gamma^*]^T$, $\alpha = [\alpha_1, \alpha_2, \alpha_3, \alpha_4, \alpha_5]^T$ and then $z = [Y, \alpha]^T$.

Below is the joined state-costate system of ODEs:

$$\frac{dz_1}{dt} = c_1 u_1{}^* cos\,(z_3 + z_4) \quad (20)$$

$$\frac{dz_2}{dt} = c_1 u_1{}^* sin\,(z_3 + z_4) \quad (21)$$

$$\frac{dz_3}{dt} = c_2 u_1{}^* cos\,(z_4) tan\,(z_5) \quad (22)$$

$$\frac{dz_4}{dt} = -bz_4 + bu_2{}^* \quad (23)$$

$$\frac{dz_5}{dt} = -cz_5 + cu_3{}^* \quad (24)$$

$$\frac{dz_6}{dt} = 0 \quad (25)$$

$$\frac{dz_7}{dt} = 0 \quad (26)$$

$$\frac{dz_8}{dt} = c_1 u_1{}^* (z_6 \sin(z_3 + z_4) - z_7 cos(z_3 + z_4)) \quad (27)$$

$$\frac{dz_9}{dt} = c_1 u_1{}^* (z_6 \sin(z_3 + z_4) - z_7 cos(z_3 + z_4)) + c_2\, z_8\, u_1{}^*\, sin(z_4)\, tan(z_5) + bz_9 \quad (28)$$

$$\frac{dz_{10}}{dt} = cz_{10} - c_2\, z_8\, u_1{}^* cos(z_4)\, sec^2(z_5) \quad (29)$$

where,

$$u_1{}^* = -0.5(z_6(c_1\, cos(z_3 + z_4)) + z_7(c_1\, sin(z_3 + z_4)) + z_8(c_2\, cos(z_4)\, tan(z_5))) \quad (30)$$

$$u_2{}^* = -0.5bz_9 \quad (31)$$

$$u_3{}^* = 0.5cz_{10} \quad (32)$$

All the above equations are applicable to Figure 3.

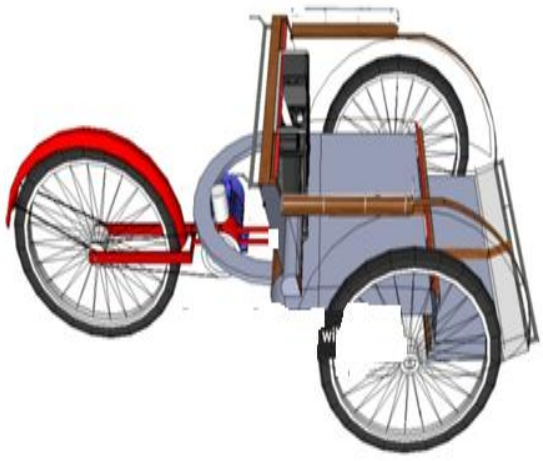

Figure 3. Becak design (side view)

6. COMPUTATIONAL SIMULATIONS

Below are the Octave/MATLAB-developed algorithms, which are as follows: dsdt=becak(t,s) codes the joined Eqs. (20)-(29) system involving expressions (30)-(32). [t,s]=Runge(fs,t0,tf,N,s0), codes a 4th-order Runge-Kutta numerical method. The third program denoted main_becak, as the main program, it contains script codes to invoke Runge(fs,t0,tf,N,s0) to solve the system coded by dsdt=becak(t,s). The Octave/MATLAB function associated to Eqs. (20)-(29) together with Eqs. (30)-(32) is given by Appendix Table 1.

To compute the solution of the system defined by the function dsdt = becak(t,s), which is an initial value problem, a function denoted fourth-order Runge-Kutta numerical method is coded into a MATLAB function. Such a function is given by Appendix Table 2.

A main function is developed and used to call the function [t,s]=Runge (fs,t0,tf,M,s0), which is defined by Appendix Table 3.

Noticing that each statement preceded by % does not affect the execution of the computer program. It is just a comment for the reader who is not familiar with MATLAB codes to know what the current statement code is performing.

To get a clear picture of the performance of the three-wheeled autonomous vehicle, three cases are considered for the initial state and for the initial costate.

Case 1: Initial condition **z0** = [3;1;0;0;0;-ones(5,1)]

With this initial condition, we have [3;1;0;0;0] as the initial state vector and [-1;-1;-1;-1;-1] as the initial costate vector. (3;1) is the starting point for the robot in the *XY* horizontal plane. The initial heading angle is 0, the initial slip angular velocity is 0, and the initial stearing angular velocity is 0. After running the program function main_becak, we obtained the feasible control strategies, the feasible trajectory of the robot, the feasible speed of the robot, the feasible state functions, and the feasible costate functions, given respectively by Figures 4-8.

The above plotted control functions are respectively $\omega(t)$, $\mu(t)$, and $\tau(t)$, which are respectively the heading angular velocity, the slip angular velocity, and the steering angular velocity of the three-wheeled autonomous.

Figure 5 gives the planned trajectory of the mobile robot. The starting point is (3;1).

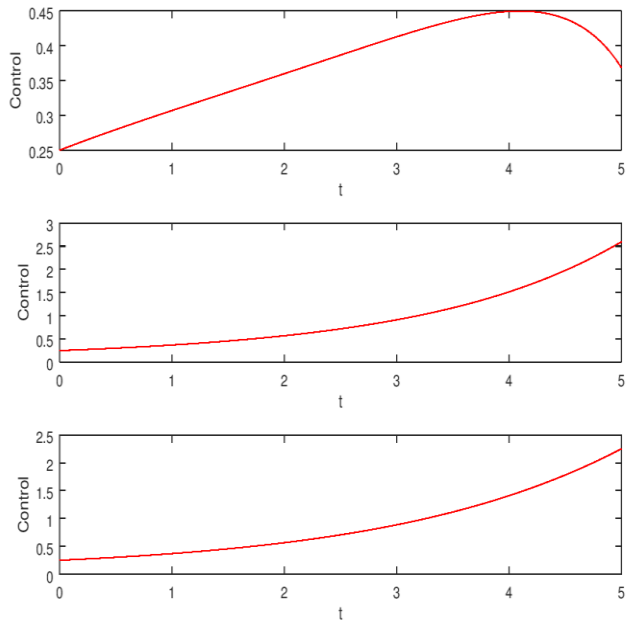

Figure 4. Feasible control strategies

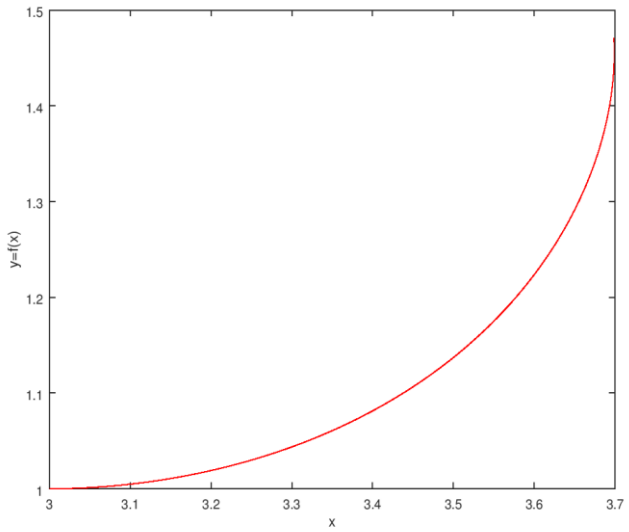

Figure 5. Feasible trajectory

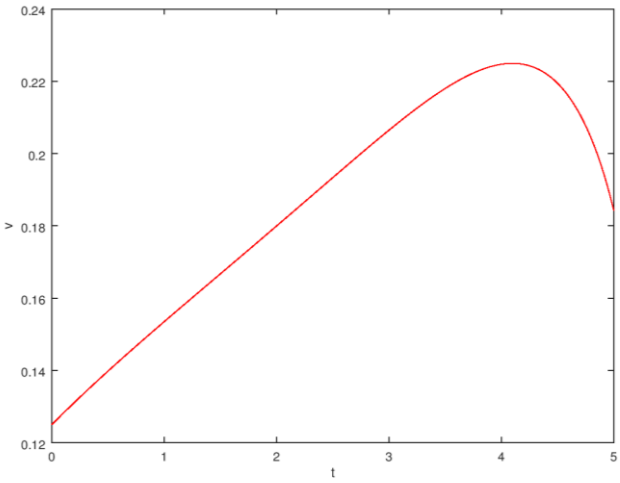

Figure 6. Feasible speed

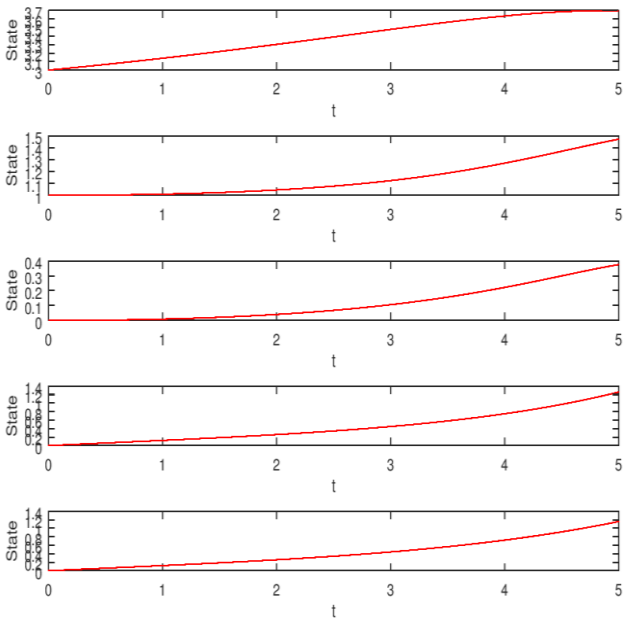

Figure 7. Feasible state functions

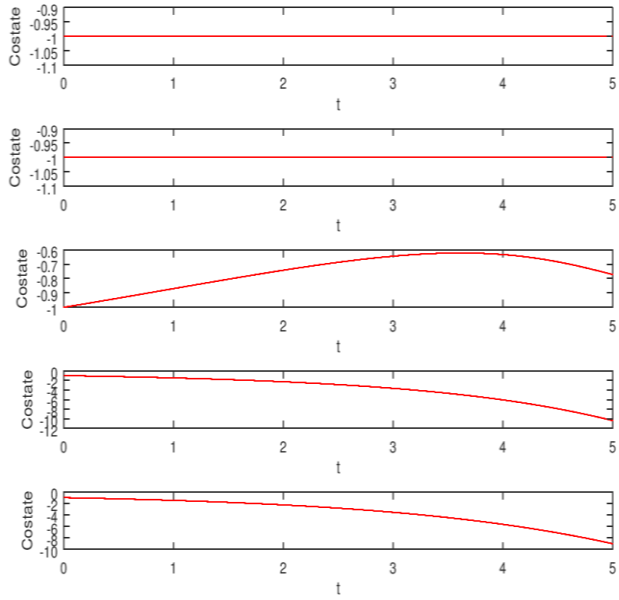

Figure 8. Feasible costate functions

Figure 6 gives the speed function of the robot. One can see that at time t = 0 s, the time has a speed not equal to zero. At time t = 5 s, the robot is still in motion. From the starting time to the final time, the robot did not stop.

Figure 7 gives the state functions which from the top to the bottom are respectively $x(t)$, $y(t)$, $\theta(t)$, $\beta(t)$ and $\gamma(t)$. Each state function is increasing on the interval [0,5].

Figure 8 represents the costate functions. They are adjoint functions to the state functions. The first two costate functions are constant.

Case 2: Initial state **z0** = [0;0;0;0;0;-5*ones(5,1)]

After running the program function main_becak, the following results are obtained: the feasible control strategies, the feasible trajectory of the robot, the feasible speed of the robot, the feasible state functions, and the feasible costate functions given respectively by Figures 9-13.

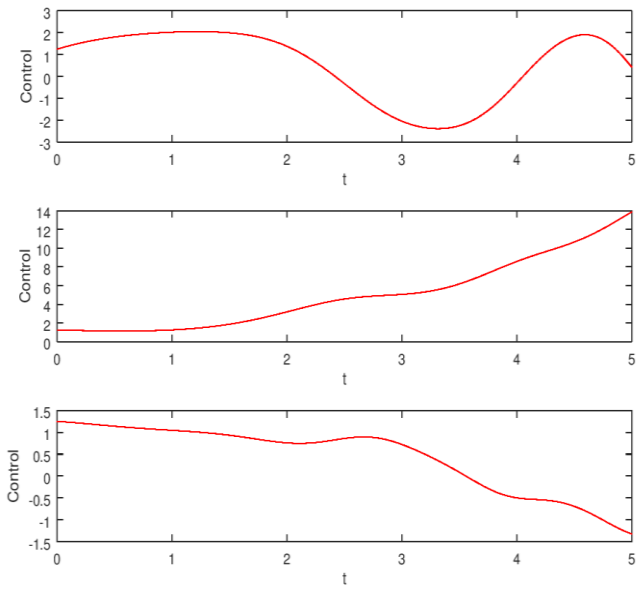

Figure 9. Feasible control strategies

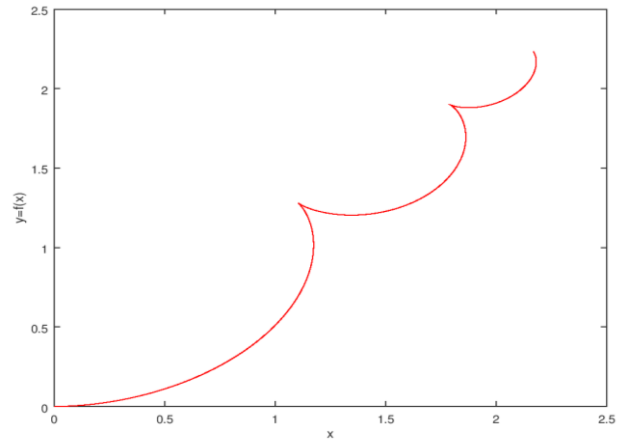

Figure 10. Feasible trajectory

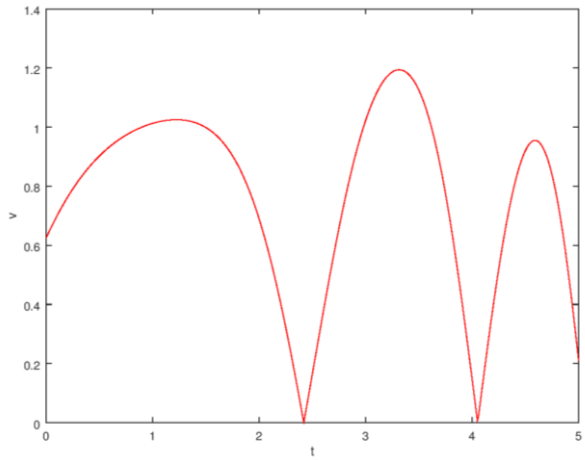

Figure 11. Feasible speed

With the new initial condition, the above plotted control functions are respectively, (ω), $\mu(t)$, and $\tau(t)$, which are respectively the heading angular velocity, the slip angular velocity, and the steering angular velocity of the three-wheeled autonomous.

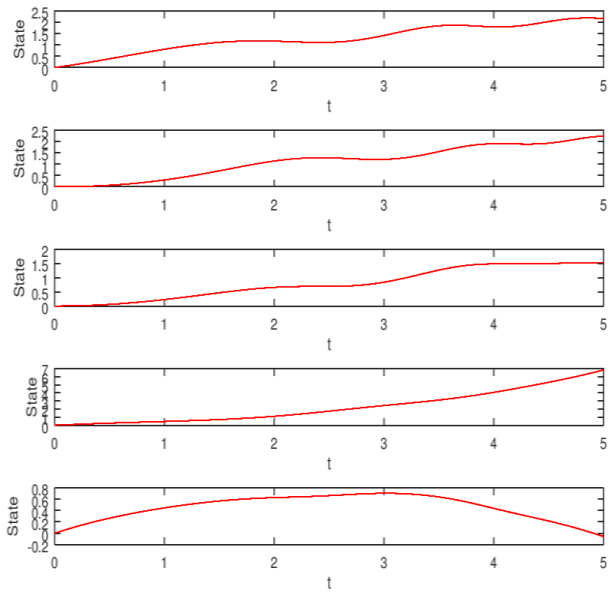

Figure 12. Feasible state functions

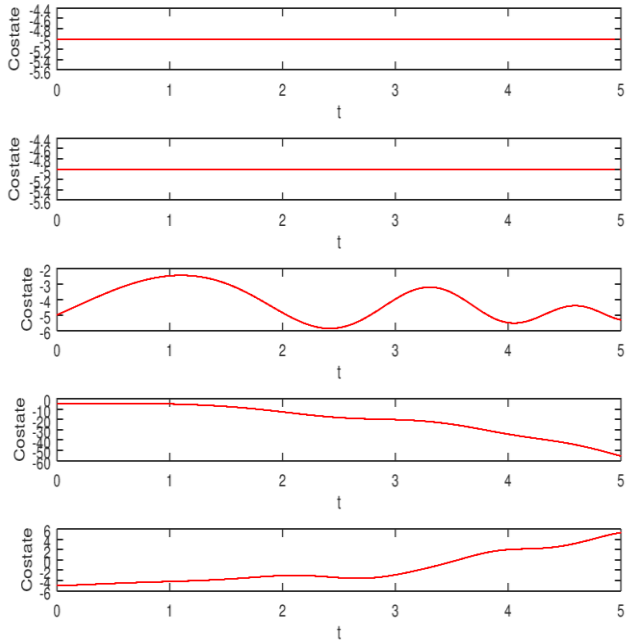

Figure 13. Feasible costate functions

Unlike the first case, the starting point is (0,0). One can notice the difference in the trajectory of the first case. For the first case, the mobile robot moves directly from the starting time to the final time, while in this case, the mobile is spending much time taking two small breaks.

One can notice that the mobile robot takes two small breaks in the interval [2s,3s] and in the interval [4s,5s].

Figure 12 gives the state functions which from the top to the bottom are respectively, $x(t)$, $y(t)$, $\theta(t)$, $\beta(t)$ and $\gamma(t)$. One can see the difference with first case state functions.

Figure 13 gives the costate functions which from the top to the bottom are respectively, $x(t)$, $y(t)$, $\theta(t)$, $\beta(t)$ and $\gamma(t)$. One can see the difference with first case state functions.

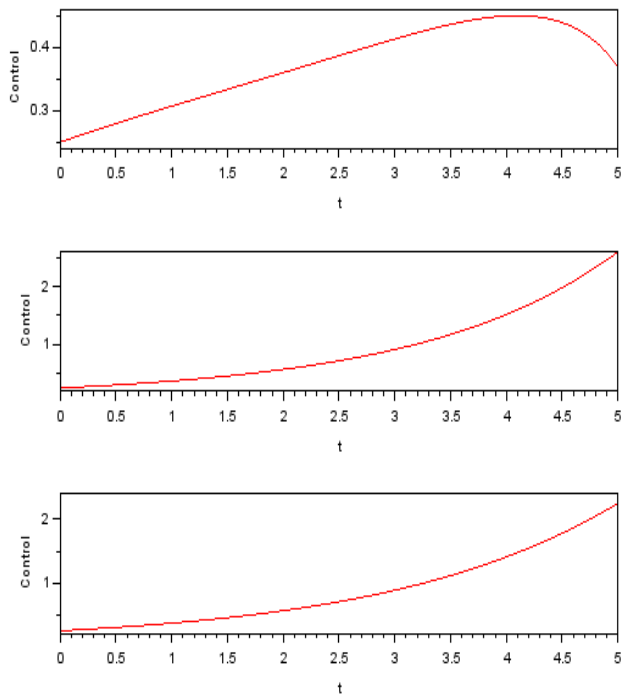

Figure 14. Feasible control strategies

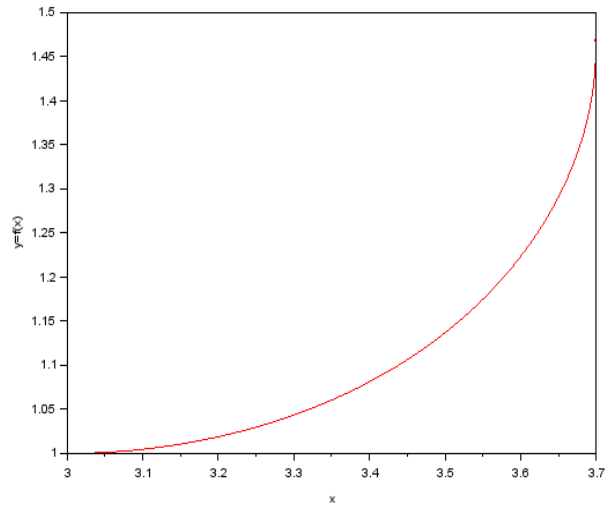

Figure 15. Feasible trajectory

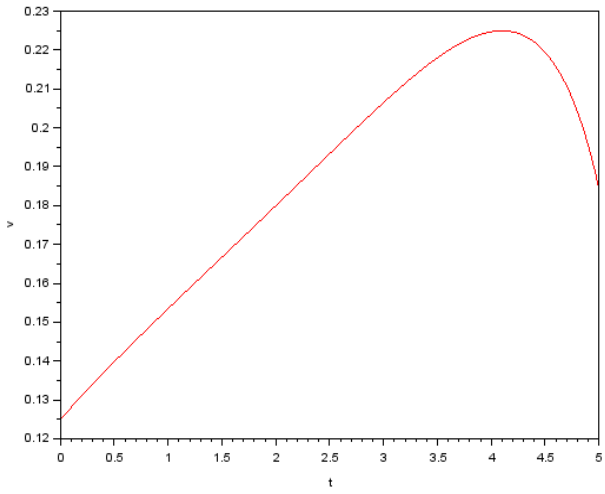

Figure 16. Feasible speed

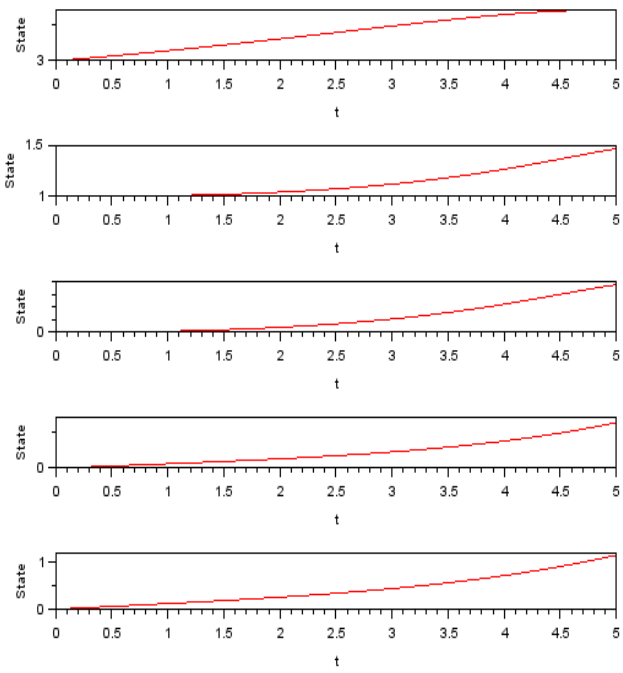

Figure 17. Feasible states

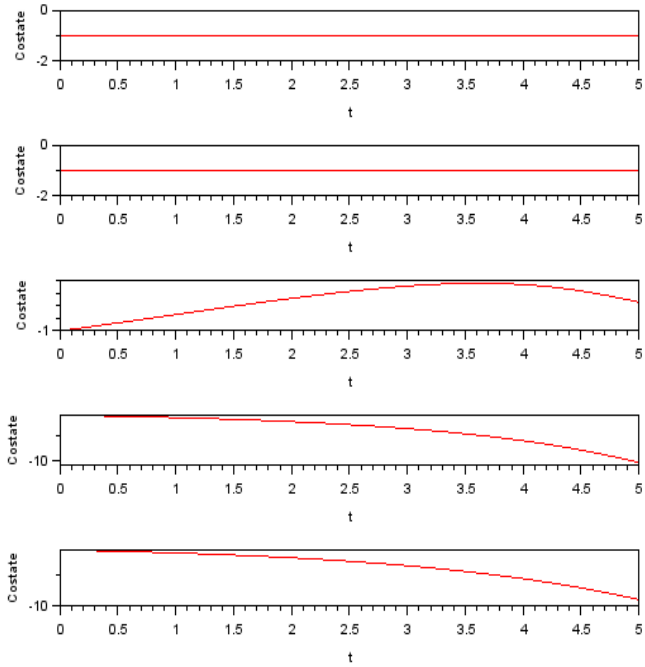

Figure 18. Feasible costates

To confirm the results, the same problem was solved by using a Computer Programming Language called Scilab. Notice that each statement preceded by //does not affect the execution of the computer program. The program given in Appendix Table 4 was developed.

After running the codes in Appendix Table 4, the results are the following: the feasible control strategies, the feasible trajectory of the robot, the feasible speed of the robot, the feasible state functions, and the feasible costate functions given respectively by Figures 14-18.

Case 1: Initial condition **z0** = [3;1;0;0;0;-ones(5,1)]

Scilab programming language has been used to solve the same problem and compare the results obtained with Octave/MATLAB. One can notice that the results represented by Figures 4-8 (obtained with Octave/MATLAB) are equal to those represented respectively by Figures 14-18 (Obtained with Scilab).

Case 2: Initial state **z0** = [0;0;0;0;0;-5*ones(5,1)];

After running the codes in Appendix Table 4 for Case 2, the results are the feasible control strategies, the feasible trajectory of the robot, the feasible speed of the robot, the feasible state functions, and the feasible costate functions, given respectively by Figure 14 and Figures 19-23.

As said above, Scilab has been used to solve the same problem and compare the results obtained with Octave / MATLAB. One can notice that the results represented by Figures 9-13 (obtained with Octave/MATLAB) are equal to those represented respectively by Figures 19-23 (Obtained with Scilab).

The same results for case 1 and case 2 can be obtained by running the computer program in Appendix Table 5.

Any reader familiar with Scilab programming can run the above program to compare the results with the previous functions.

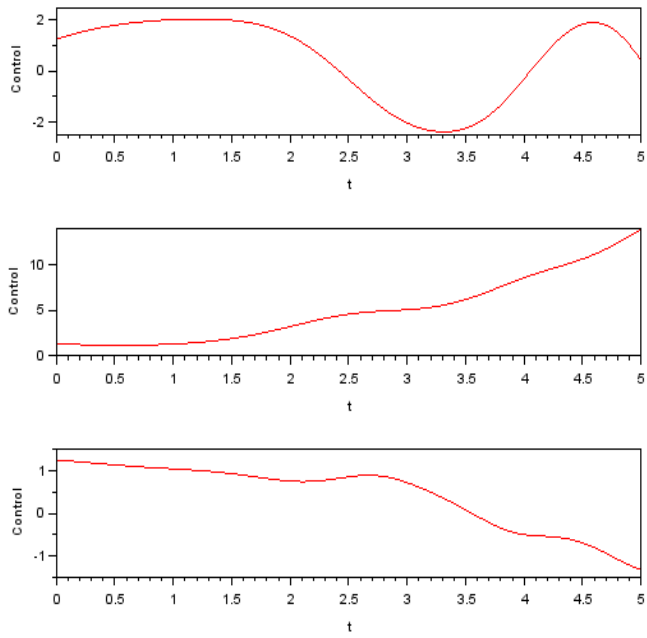

Figure 19. Feasible control strategies

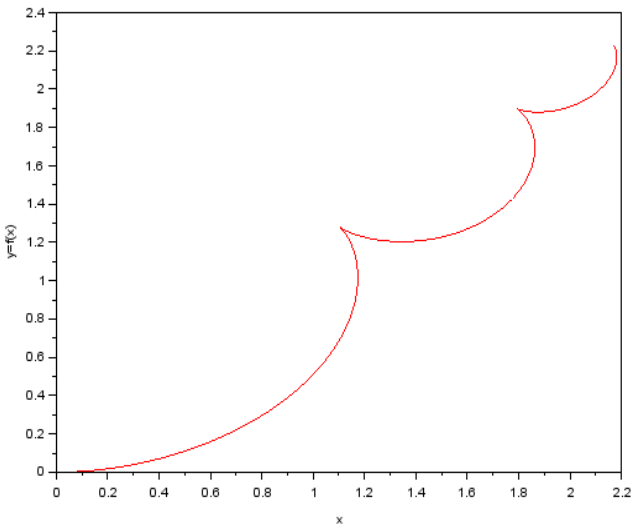

Figure 20. Feasible trajectory

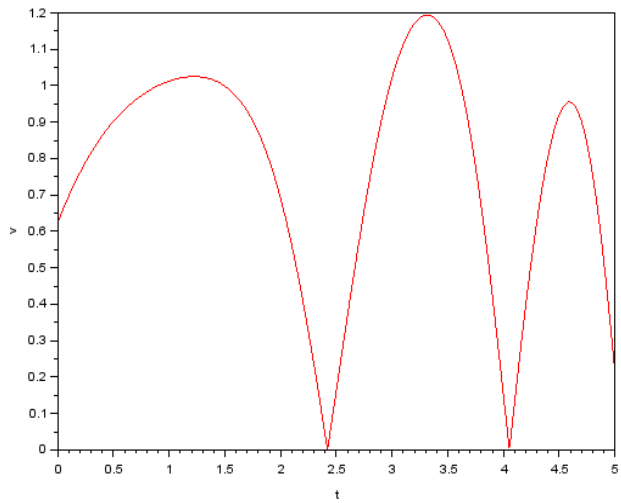

Figure 21. Feasible speed

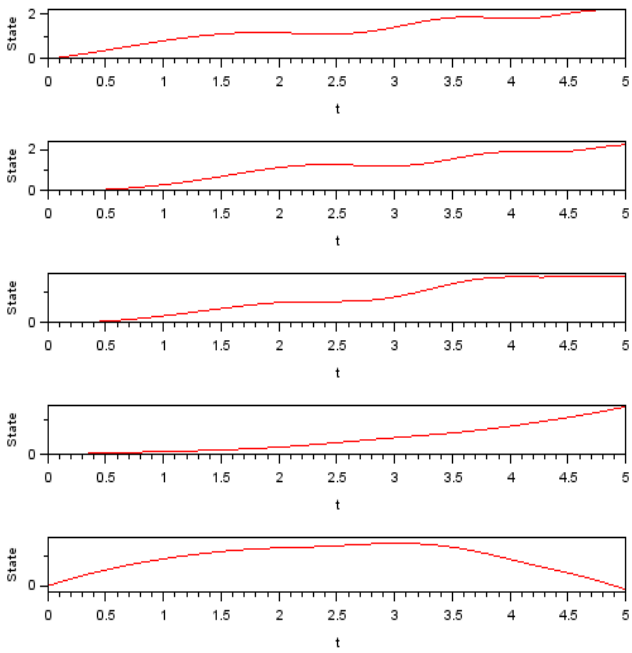

Figure 22. Feasible states

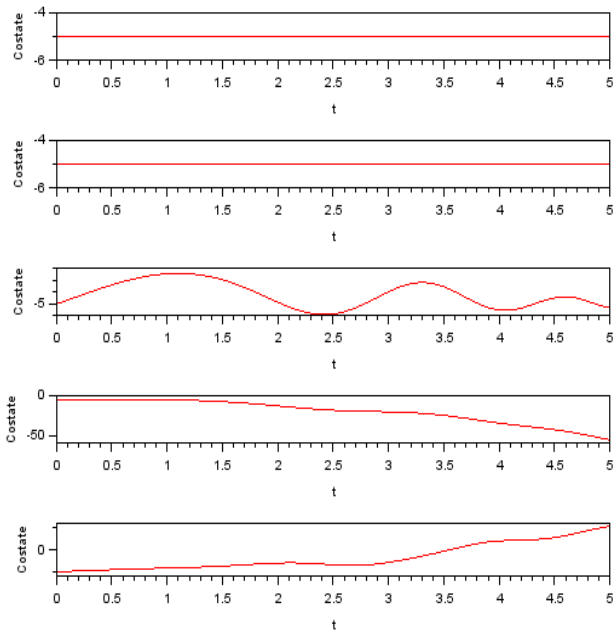

Figure 23. Feasible costates

7. CONCLUSION

This study used optimal control theory for controlling the motion of an autonomous three-wheeled vehicle. The first-order optimality conditions derive three feasible commands and a system of ODEs involving state and costate variables. Initial conditions were imposed on the joined control-free state-costate system to obtain an initial value problem, which was solved using a developed Octave/MATLAB computer program. The Scilab computer programming language was also used to check the results obtained with Octave and MATLAB. Computational simulations are designed and presented to convince the reader about the accuracy, the effectiveness, and the efficiency of the results. Compared to other papers which control the same type of robot, this paper

has the advantage of setting up the dependence of the feasible states on the feasible commands in an automatic way by using the command optimality conditions. Such an approach enables the manipulation of the control system without any simplification. Unlike other papers, this paper has provided computer programs so that any reader familiar with computer programming can check the accuracy of the results. If boundary conditions are imposed on the problem, the original problem would still be an optimal control problem. Specific data conditions may lead to the obtention of a singular Jacobian, which can constitute some computational limitations. Two programming environments are used to enrich the quality of the work and to convince the readers of the reliability of the results.

ACKNOWLEDGMENT

The author is grateful to the University of Johannesburg, Faculty of Engineering and Built Environment for the funding.

REFERENCES

[1] Pandey, A., Jha, S., Chakravarty, D. (2017). Modeling and control of an autonomous three wheeled mobile robot with front steer. In 2017 First IEEE International Conference on Robotic Computing (IRC), Taichung, Taiwan, pp. 136-142. https://doi.org/10.1109/IRC.2017.67

[2] Chen, C.L., Li, J., Li, M., Xie, L. (2018). Model predictive trajectory tracking control of automated guided vehicle in complex environments. In 2018 IEEE 14th International Conference on Control and Automation (ICCA), Anchorage, AK, USA, pp. 405-410. https://doi.org/10.1109/ICCA.2018.8444247

[3] Alfiany, N., Hamid, N., Jati, G., Ma'sum, M.A., Jatmiko, W. (2019). Kinematics and dynamics analysis of an autonomous three-wheeled bicycle modeling. In 2019 4th Asia-Pacific Conference on Intelligent Robot Systems (ACIRS), Nagoya, Japan, pp. 17-21. https://doi.org/10.1109/ACIRS.2019.8935966

[4] Batti, H., Jabeur, C.B., Seddik, H. (2019). Fuzzy logic controller for autonomous mobile robot navigation. In 2019 International Conference on Control, Automation and Diagnosis (ICCAD), Grenoble, France, pp. 1-6. https://doi.org/10.1109/ICCAD46983.2019.9037922

[5] Alfiany, N., Jati, G., Hamid, N., Ramadhani, R.A., Santika, M.W.D., Jatmiko, W. (2020). Kinematics and simulation model of autonomous Indonesian "Becak" Robot. In 2020 IEEE Region 10 Symposium (TENSYMP), Dhaka, Bangladesh, pp. 1692-1695. https://doi.org/10.1109/TENSYMP50017.2020.9230782

[6] Alghanim, M.N., Valavanis, K.P., Rutherford, M.J. (2019). Modeling, control, and wheel-terrain interaction dynamics of the UGV Argo J5. In 2019 18th European Control Conference (ECC), Naples, Italy, pp. 1116-1123. https://doi.org/10.23919/ECC.2019.8796270

[7] Alghanim, M.A. (2019). Modeling and control of the UGV Argo J5 with a custom-built landing platform. Doctoral dissertation, University of Denver.

[8] Mishra, D., Yadav, R.S., Agrawal, K.K. (2020). Kinematic modelling and emulation of robot for traversing over the pipeline in the refinery. Microsystem Technologies, 26(3): 1011-1020. https://doi.org/10.1007/s00542-019-04615-9

[9] Mellah, S., Graton, G., El Adel, E.M., Ouladsine, M., Planchais, A. (2021). Health state monitoring of 4-mecanum wheeled mobile robot actuators and its impact on the robot behavior analysis. Journal of Intelligent & Robotic Systems, 102(4): 86. https://doi.org/10.1007/s10846-021-01446-7

[10] Saeedi, M.A., Kazemi, R. (2013). Stability of three-wheeled vehicles with and without control system. Internal Journal of Automotive Engineering, 3(1): 343-355.

[11] Saypulaev, G.R., Saypulaev, M.R., Demidov, A.A., Sokiranskaya, E.K., Semenyakina, E.S., Lipatov, A.A. (2025). Mathematical modeling of the spatial kinematics of an omni-platform balancing on a spherical wheel. In 2025 7th International Youth Conference on Radio Electronics, Electrical and Power Engineering (REEPE), Moscow, Russian Federation, pp. 1-6. https://doi.org/10.1109/REEPE63962.2025.10971113

[12] Xu, H., Xue, K., Peng, F., Yu, S., Gao, X., Ouyang, Q., Chang, Q., Lu, Z. (2007). 3-D kinematics modeling for mobile robot with steering castered-and-cambered wheels. In 2007 International Conference on Mechatronics and Automation, Harbin, China pp. 1345-1350. https://doi.org/10.1109/ICMA.2007.4303745

[13] Xue, D., Chen, Y. (2007). Solving control related mathematic problems in MATLAB. In 2007 International Conference on Mechatronics and Automation, Harbin, China, pp. nil37-nil38. https://doi.org/10.1109/ICMA.2007.4303498

[14] Trojnacki, M. (2013). Modeling and motion simulation of a three-wheeled mobile robot with front wheel driven and steered taking into account wheels' slip. Archive of Applied Mechanics, 83: 109-124. https://doi.org/10.1007/s00419-012-0636-2

[15] Bin, H., Zhen, L.W., Feng, L.H. (2010). The kinematics model of a two-wheeled self-balancing autonomous mobile robot and its simulation. In 2010 Second International Conference on Computer Engineering and Applications, Bali, Indonesia, pp. 64-68. https://doi.org/10.1109/ICCEA.2010.169

[16] Li, J., Gao, X., Huang, Q., Du, Q., Duan, X. (2007). Mechanical design and dynamic modeling of a two-wheeled inverted pendulum mobile robot. In 2007 IEEE International Conference on Automation and Logistics, Jinan, China, pp. 1614-1619. https://doi.org/10.1109/ICAL.2007.4338830

[17] Indiveri, G. (2009). Swedish wheeled omnidirectional mobile robots: Kinematics analysis and control. IEEE Transactions on Robotics, 25(1): 164-171. https://doi.org/10.1109/TRO.2008.2010360

[18] Mellah, S., Graton, G., El Adel, E.M., Ouladsine, M., Planchais, A. (2020). 4-mecanum wheeled mobile robot actuator fault detection & isolation using unknown input observer-based approach. In 2020 European Control Conference (ECC), Petersburg, Russia, pp. 1442-1447. https://doi.org/10.23919/ECC51009.2020.9143984

[19] Koubaa, Y., Boukattaya, M., Dammak, T. (2016). Adaptive dynamic tracking control of uncertain wheeled mobile robot including actuator dynamics. In 2016 17th International Conference on Sciences and Techniques of Automatic Control and Computer Engineering (STA), Sousse, Tunisia, pp. 374-379.

https://doi.org/10.1109/STA.2016.7952018
[20] Sofwan, A., Mulyana, H.R., Afrisal, H., Goni, A. (2019). Development of omni-wheeled mobile robot based-on inverse kinematics and odometry. In 2019 6th International Conference on Information Technology, Computer and Electrical Engineering (ICITACEE), Semarang, Indonesia, pp. 1-6. https://doi.org/10.1109/ICITACEE.2019.8904418
[21] Labakhua, L., Martins, I., Igor, M. (2017). Control of a mobile robot with Swedish wheels. In 2017 IEEE International Conference on Power, Control, Signals and Instrumentation Engineering (ICPCSI), Chennai, India, pp. 267-272. https://doi.org/10.1109/ICPCSI.2017.8392225
[22] Pei, Y., Kleeman, L. (2017). A novel odometry model for wheeled mobile robots incorporating linear acceleration. In 2017 IEEE International Conference on Mechatronics and Automation (ICMA), Takamatsu, Japan, pp. 1396-1403. https://doi.org/10.1109/ICMA.2017.8016021

APPENDIX

Table 1. State and costate systems merged and coded

```
function dsdt=becak(t,s)
c1=0.5; c2=0.5; b=0.5; c=0.5; % These are Constants of
proportionality.
dsdt=zeros(10,1);
dsdt(1)=c1*(-
0.5*(s(6)*(c1*cos(s(3)+s(4)))+s(7)*(c1*sin(s(3)+s(4))) +
s(8)*(c2*cos(s(4))*tan(s(5)))))*cos(s(3)+s(4));
dsdt(2)=c1*(-
0.5*(s(6)*(c1*cos(s(3)+s(4)))+s(7)*(c1*sin(s(3)+s(4))) +
s(8)*(c2*cos(s(4))*tan(s(5)))))*sin(s(3)+s(4));
dsdt(3)=c2*(-
0.5*(s(6)*(c1*cos(s(3)+s(4)))+s(7)*(c1*sin(s(3)+s(4))) +
s(8)*(c2*cos(s(4))*tan(s(5)))))*cos(s(4))*tan(s(5));
dsdt(4)=-b*s(4)+b*(-0.5*b*s(9));
dsdt(5)=-c*s(5)+c*(-0.5*c*s(10));
dsdt(6)=0;
dsdt(7)=0;
dsdt(8)=c1*(-
0.5*(s(6)*(c1*cos(s(3)+s(4)))+s(7)*(c1*sin(s(3)+s(4))) +
s(8)*(c2*cos(s(4))*tan(s(5)))))*(s(6)*sin(s(3)+s(4))-
s(7)*cos(s(3)+s(4)));
dsdt(9)=c1*(-0.5*(s(6)*(c1*cos(s(3)+s(4))) +
s(7)*(c1*sin(s(3)+s(4))) +
s(8)*(c2*cos(s(4))*tan(s(5)))))*(s(6)*sin(s(3)+s(4)) –
s(7)*cos(s(3)+s(4))) + c2*(-0.5*(s(6)*(c1*cos(s(3)+s(4))) +
s(7)*(c1*sin(s(3)+s(4)))+s(8)*(c2*cos(s(4))*tan(s(5)))))*s(8)*si
n(s(4))*tan(s(5))+b*s(9);
dsdt(10)=c*s(10) – c2*(-0.5*(s(6)*(c1*cos(s(3)+s(4)))+
s(7)*(c1*sin(s(3)+s(4)))+s(8)*(c2*cos(s(4))*tan(s(5)))))*s(8)*co
s(s(4))*sec(s(5))*sec(s(5));
```

Table 2. Runge-Kutta coded

```
function [t,s] = runge(fs,t0,tf,M,s0)
h=(tf-t0)./(M-1);%The step size;
s = zeros(M,length(s0));
s(1,:) = s0.'; %Each mesh point of the solution is a row vector
for i = 2:M
p1 = feval(fs,t(i-1),s(i-1,:));
p2 = feval(fs,t(i-1)+(h/2),s(i-1,:)+(h/2)*p1');
p3 = feval(fs,t(i-1)+(h/2),s(i-1,:)+(h/2)*p2');
p4 = feval(fs,t(i-1)+h,s(i-1,:)+h*p3');
s(i,:) = s(i-1,:)+(h/6)*(p1'+2*p2'+2*p3'+p4');
end
```

Table 3. Main function

```
function main_becak
clear all
clc
format short
c1=0.5; c2=0.5; b=0.5; c=0.5; %c1=R; c2=R/L;
% R is the radius of each wheel; L is the distance between the front
and the rear wheels.
t0=0;
tf=5;
N=501; % N is the number of discrete points.
h=(tf-t0)/(N-1);% Step size.
t=t0:h:tf; % vector of discrete times
s=zeros(N,10); % Initialization of s
s0=[0;0;0;0;0;-5*ones(5,1)];
t=t';
[t,s]=runge('becak',t0,tf,N,s0); % The main function calls
function runge to solve the system of equations.
control1=-
0.5*(s(:,6).*(c1*cos(s(:,3)+s(:,4)))+s(:,7).*(c1*sin(s(:,3)+s(:,4)))
+ s(:,8).*(c2*cos(s(:,4)).*tan(s(:,5))));
control2=-0.5*b*s(:,9);
control3=-0.5*c*s(:,10);
control=[control1,control2,control3]; % Grouping all the control
functions into a vector.
dx=c1*(-
0.5*(s(:,6).*(c1*cos(s(:,3)+s(:,4)))+s(:,7).*(c1*sin(s(:,3)+s(:,4)))
+ s(:,8).*(c2*cos(s(:,4)).*tan(s(:,5))))).*cos(s(:,3)+s(:,4)); % The
velocity along x axis.
dy=c1*(-
0.5*(s(:,6).*(c1*cos(s(:,3)+s(:,4)))+s(:,7).*(c1*sin(s(:,3)+s(:,4)))
+ s(:,8).*(c2*cos(s(:,4)).*tan(s(:,5))))).*sin(s(:,3)+s(:,4)); % The
velocity along y axis.
dTheta=c2*(-0.5*(s(:,6).*(c1*cos(s(:,3)+s(:,4))) +
s(:,7).*(c1*sin(s(:,3)+s(:,4))) +
s(:,8).*(c2*cos(s(:,4)).*tan(s(:,5))))).*cos(s(:,4)).*tan(s(:,5)); %
Heading angular velocity.
dDelta=-b*s(:,4)+b*(-0.5*b*s(:,9)); % is the steering angular
velocity.
dBeta=-c*s(:,5)+c*(-0.5*c*s(:,10)); % is the slipping angular
velocity.
speed=(dx.^2 + dy.^2).^(0.5);
plot(s(:,1),s(:,2),'r'); xlabel('x');ylabel('y=f(x)'); % Plot of the
feasible trajectory for the robot.
print Robot_threeWheeled_Trajectory.png; % Converting the plot
into png format.
plot(t,speed,'r'); xlabel('t');ylabel('v'); % Plot of the feasible
speed for the robot.
print Robot__threeWheeled_speed.png; % Converting the plot
into png format.
for k=1:5
 subplot(5,1,k); plot(t,s(:,k),'r');xlabel('t');ylabel('State'); % Plot
of 5 feasible states functions for the robot.
end
print Robot_States_threeWheeled.png; % Converting the plot into
png format.
for k=1:5
 subplot(5,1,k); plot(t,s(:,5+k),'r');xlabel('t');ylabel('Costate');
end
print Robot_Costates_threeWheeled.png; % Converting the plot
into png format.
for k=1:3
 subplot(3,1,k);
plot(t,control(:,k),'r');xlabel('t');ylabel('Control'); % Plot of the
feasible controls.
end
print Robot_Controls_threeWheeled.png; % Converting the plot
into png format.
```

Table 4. Scilab confirmation program

```
function dsdt=becak(t,s)
c1=0.5; c2=0.5; b=0.5; c=0.5; // These are Constants of
proportionality.
dsdt=zeros(10,1);
dsdt(1)=c1*(-
0.5*(s(6)*(c1*cos(s(3)+s(4)))+s(7)*(c1*sin(s(3)+s(4))) +
s(8)*(c2*cos(s(4))*tan(s(5)))))*cos(s(3)+s(4));
dsdt(2)=c1*(-
0.5*(s(6)*(c1*cos(s(3)+s(4)))+s(7)*(c1*sin(s(3)+s(4))) +
s(8)*(c2*cos(s(4))*tan(s(5)))))*sin(s(3)+s(4));
dsdt(3)=c2*(-
0.5*(s(6)*(c1*cos(s(3)+s(4)))+s(7)*(c1*sin(s(3)+s(4))) +
s(8)*(c2*cos(s(4))*tan(s(5)))))*cos(s(4))*tan(s(5));
dsdt(4)=-b*s(4)+b*(-0.5*b*s(9));
dsdt(5)=-c*s(5)+c*(-0.5*c*s(10));
dsdt(6)=0;
dsdt(7)=0;
dsdt(8)=c1*(-
0.5*(s(6)*(c1*cos(s(3)+s(4)))+s(7)*(c1*sin(s(3)+s(4)))+s(8)*(c2
*cos(s(4))*tan(s(5)))))*(s(6)*sin(s(3)+s(4))-s(7)*cos(s(3)+s(4)));
dsdt(9)=c1*(-
0.5*(s(6)*(c1*cos(s(3)+s(4)))+s(7)*(c1*sin(s(3)+s(4)))+s(8)*(c2
*cos(s(4))*tan(s(5)))))*(s(6)*sin(s(3)+s(4))-
s(7)*cos(s(3)+s(4)))+c2*(-
0.5*(s(6)*(c1*cos(s(3)+s(4)))+s(7)*(c1*sin(s(3)+s(4)))+s(8)*(c2
*cos(s(4))*tan(s(5)))))*s(8)*sin(s(4))*tan(s(5))+b*s(9);
dsdt(10)=c*s(10)-c2*(-0.5*(s(6)*(c1*cos(s(3)+s(4)))+
s(7)*(c1*sin(s(3)+s(4)))+s(8)*(c2*cos(s(4))*tan(s(5)))))*s(8)*co
s(s(4))*sec(s(5))*sec(s(5));
endfunction

t0=0; h=0.01; tN=5; s0=[3;1;0;0;0;-
ones(5,1)];//s0=[5.0;2.0;%pi/2;1.0;0;-ones(5,1)]
t=t0:h:tN; s = ode(s0,t0,t,becak); s=s';
control1=-
0.5*(s(:,6).*(c1*cos(s(:,3)+s(:,4)))+s(:,7).*(c1*sin(s(:,3)+s(:,4)))
+ s(:,8).*(c2*cos(s(:,4)).*tan(s(:,5))));
control2=-0.5*b*s(:,9);
control3=-0.5*c*s(:,10);
control=[control1,control2,control3]; // Grouping all the control
functions into a vector.
dx=c1*(-
0.5*(s(:,6).*(c1*cos(s(:,3)+s(:,4)))+s(:,7).*(c1*sin(s(:,3)+s(:,4)))
+ s(:,8).*(c2*cos(s(:,4)).*tan(s(:,5))))).*cos(s(:,3)+s(:,4)); // The
velocity along x axis.
dy=c1*(-
0.5*(s(:,6).*(c1*cos(s(:,3)+s(:,4)))+s(:,7).*(c1*sin(s(:,3)+s(:,4)))
+ s(:,8).*(c2*cos(s(:,4)).*tan(s(:,5))))).*sin(s(:,3)+s(:,4)); // The
velocity along y axis.
dTheta=c2*(-0.5*(s(:,6).*(c1*cos(s(:,3)+s(:,4))) +
s(:,7).*(c1*sin(s(:,3)+s(:,4))) +
s(:,8).*(c2*cos(s(:,4)).*tan(s(:,5))))).*cos(s(:,4)).*tan(s(:,5)); //
Heading angular velocity.
dDelta=-b*s(:,4)+b*(-0.5*b*s(:,9)); // is the steering angular
velocity.
dBeta=-c*s(:,5)+c*(-0.5*c*s(:,10)); // is the slipping angular
velocity.
speed=(dx.^2 + dy.^2).^(0.5);
plot(s(:,1),s(:,2),'r'); xlabel('x');ylabel('y=f(x)'); // Plot of the
feasible trajectory for the robot.
xs2png(0,'C:\Users\Guest\Documents\26january2024\Temporary
_25may2024\Robot_threeWheeled_Trajectory8.png'); //
Converting the plot into png format.
plot(t,speed,'r'); xlabel('t');ylabel('v'); // Plot of the feasible speed
for the robot.
xs2png(0,'C:\Users\Guest\Documents\26january2024\Temporary
_25may2024\Robot__threeWheeled_speed8.png'); // Converting
the plot into png format.
for k=1:5
 subplot(5,1,k); plot(t,s(:,k),'r');xlabel('t');ylabel('State'); // Plot of
5 feasible states functions for the robot.
end
xs2png(0,'C:\Users\Guest\Documents\26january2024\Temporary
_25may2024\Robot_States_threeWheeled8.png'); // Converting
the plot into png format.
for k=1:5
 subplot(5,1,k); plot(t,s(:,5+k),'r');xlabel('t');ylabel('Costate');
end
xs2png(0,'C:\Users\Guest\Documents\26january2024\Temporary
_25may2024\Robot_Costates_threeWheeled8.png'); //
Converting the plot into png format.
for k=1:3
 subplot(3,1,k); plot(t,control(:,k),'r');xlabel('t');ylabel('Control');
// Plot of the feasible controls.
end
xs2png(0,'C:\Users\Guest\Documents\26january2024\Temporary
_25may2024\Robot_Controls_threeWheeled8.png'); //
Converting the plot into png format.
// To run the program, go to the command windows (console)
and use the command " exec('C:\Users\Guest\becak1.sce', -1) "
```

Table 5. An equivalent Scilab computer program

```
//file becak2.sce
c1=0.5; c2=0.5; b=0.5; c=0.5; // These are Constants of
proportionality.
deff('dz=f(x,s)','dz=[c1*(0.5*(s(6)*(c1*cos(s(3)+s(4)))+s(7)*(c1
*sin(s(3)+s(4)))+s(8)*(c2*cos(s(4))*tan(s(5)))))*cos(s(3)+s(4));
c1*(0.5*(s(6)*(c1*cos(s(3)+s(4)))+s(7)*(c1*sin(s(3)+s(4)))+s(8)
*(c2*cos(s(4))*tan(s(5)))))*sin(s(3)+s(4));
c2*(-
0.5*(s(6)*(c1*cos(s(3)+s(4)))+s(7)*(c1*sin(s(3)+s(4)))+s(8)*(c2
*cos(s(4))*tan(s(5)))))*cos(s(4))*tan(s(5));
-b*s(4)+b*(-0.5*b*s(9));-c*s(5)+c*(-
0.5*c*s(10));0;0;c1*(0.5*(s(6)*(c1*cos(s(3)+s(4)))+s(7)*(c1*sin
(s(3)+s(4)))+s(8)*(c2*cos(s(4))*tan(s(5)))))*(s(6)*sin(s(3)+s(4))
-s(7)*cos(s(3)+s(4)));c1*(-
0.5*(s(6)*(c1*cos(s(3)+s(4)))+s(7)*(c1*sin(s(3)+s(4)))+s(8)*(c2
*cos(s(4))*tan(s(5)))))*(s(6)*sin(s(3)+s(4))-
s(7)*cos(s(3)+s(4)))+c2*(-
0.5*(s(6)*(c1*cos(s(3)+s(4)))+s(7)*(c1*sin(s(3)+s(4)))+s(8)*(c2
*cos(s(4))*tan(s(5)))))*s(8)*sin(s(4))*tan(s(5))+b*s(9);;',
        ;c*s(10)-c2*(-0.5*(s(6)*(c1*cos(s(3)+s(4)))+
s(7)*(c1*sin(s(3)+s(4)))+s(8)*(c2*cos(s(4))*tan(s(5)))))*s(8)*co
s(s(4))*sec(s(5))*sec(s(5))',
         'w=[s1;s2;s3;s4;s5;s6;s7;s8;s9;s10]']);
t0=0; h=0.01; tN=5; s0=[3;1;0;0;0;-ones(5,1)];
t=t0:h:tN; s = ode(s0,t0,t,becak); s=s';
control1=-
0.5*(s(:,6).*(c1*cos(s(:,3)+s(:,4)))+s(:,7).*(c1*sin(s(:,3)+s(:,4)))
+ s(:,8).*(c2*cos(s(:,4)).*tan(s(:,5))));
control2=-0.5*b*s(:,9);
control3=-0.5*c*s(:,10);
control=[control1,control2,control3]; // Grouping all the control
functions into a vector.
dx=c1*(-
0.5*(s(:,6).*(c1*cos(s(:,3)+s(:,4)))+s(:,7).*(c1*sin(s(:,3)+s(:,4)))
+ s(:,8).*(c2*cos(s(:,4)).*tan(s(:,5))))).*cos(s(:,3)+s(:,4)); // The
velocity along x axis.
dy=c1*(-
0.5*(s(:,6).*(c1*cos(s(:,3)+s(:,4)))+s(:,7).*(c1*sin(s(:,3)+s(:,4)))
+ s(:,8).*(c2*cos(s(:,4)).*tan(s(:,5))))).*sin(s(:,3)+s(:,4)); // The
velocity along y axis.
dTheta=c2*(-0.5*(s(:,6).*(c1*cos(s(:,3)+s(:,4))) +
s(:,7).*(c1*sin(s(:,3)+s(:,4))) +
s(:,8).*(c2*cos(s(:,4)).*tan(s(:,5))))).*cos(s(:,4)).*tan(s(:,5)); //
Heading angular velocity.
dDelta=-b*s(:,4)+b*(-0.5*b*s(:,9)); // is the steering angular
velocity.
```

```
dBeta=-c*s(:,5)+c*(-0.5*c*s(:,10)); // is the slipping angular velocity.
speed=(dx.^2 + dy.^2).^(0.5);
plot(s(:,1),s(:,2),'r'); xlabel('x');ylabel('y=f(x)'); // Plot of the feasible trajectory for the robot.
xs2png(0,'C:\Users\Guest\Documents\26january2024\Temporary_25may2024\Robot_threeWheeled_Trajectory9.png'); // Converting the plot into png format.
plot(t,speed,'r'); xlabel('t');ylabel('v'); // Plot of the feasible speed for the robot.
xs2png(0,'C:\Users\Guest\Documents\26january2024\Temporary_25may2024\Robot__threeWheeled_speed9.png'); // Converting the plot into png format.
for k=1:5
 subplot(5,1,k); plot(t,s(:,k),'r');xlabel('t');ylabel('State'); // Plot of 5 feasible states functions for the robot.
end
```

```
xs2png(0,'C:\Users\Guest\Documents\26january2024\Temporary_25may2024\Robot_States_threeWheeled8.png'); // Converting the plot into png format.
for k=1:5
 subplot(5,1,k); plot(t,s(:,5+k),'r');xlabel('t');ylabel('Costate');
end
xs2png(0,'C:\Users\Guest\Documents\26january2024\Temporary_25may2024\Robot_Costates_threeWheeled9.png'); // Converting the plot into png format.
for k=1:3
 subplot(3,1,k); plot(t,control(:,k),'r');xlabel('t');ylabel('Control'); // Plot of the feasible controls.
end
xs2png(0,'C:\Users\Guest\Documents\26january2024\Temporary_25may2024\Robot_Controls_threeWheeled9.png'); // Converting the plot into png format.
// To run the program, go to the command windows (console) and use the command " exec('C:\Users\Guest\becak2.sce', -1) "
```