Article **Journal Applied Computing**
December 2020 Vol.4 No.15 1-8

Comparison of numerical methods in code as solvers for simulation of robotic systems

Comparación de métodos numéricos en código como solucionadores para simulación de sistemas robóticos

TORRES-DEL CARMEN, Felipe de Jesús†*, JARAMILLO-HERNÁNDEZ, Ricardo, DÍAZ-SÁNCHEZ, Arnaldo and NÚÑEZ-ALTAMIRANO, Diego Alfredo

Universidad de Guanajuato. Engineering Division Campus Irapuato-Salamanca, Department of Mechanical Engineering.

ID 1st Author: *Felipe de Jesús, Torres-Del Carmen* / **ORC ID:** 0000-0001-5792-2098, **CVU CONACYT ID:** 170819

ID 1st Coauthor: *Ricardo, Jaramillo-Hernández* / **ORC ID:** 0000-0002-9212-2261

ID 2nd Coauthor: *Arnaldo, Díaz-Sánchez* / **ORC ID:** 0000-0003-1334-3515

ID 3rd Coauthor: *Diego Alfredo, Núñez-Altamirano* / **ORC ID:** 0000-0002-6709-8108

DOI: 10.35429/JCA.2020.15.4.1.8 Received July 10, 2020; Accepted December 30, 2020

Abstract

This research introduces the development of the implementation and comparison of algorithms of numeric methods which solve a system of ordinary differential equations, commonly known like solvers. These were applied to a robotic system with 4 grades of freedom in open loop based on the non-linear dynamic model in the joint space. The performance of the robotic system solution simulated on Matlab®/Simulink® with S-Function has been assumed to be the reference criterion to contrast the results that were get from codification of the solvers. Moreover, some inferences were set for each one of the algorythms, for instance, simulation time and computing cost. The analysis of the results lets account the implementation of the code of the numeric methods for simulation purposes, thus, it may aid for the optimization of simulation times and computing cost.

Solvers, Numeric methods, Robotic systems

Resumen

Este trabajo presenta el desarrollo de la implementación y comparación de algoritmos de métodos numéricos que resuelvan un sistema de ecuaciones diferenciales ordinarias, comúnmente conocidos como solucionadores. Los cuales fueron aplicados a un sistema robótico de 4 grados de libertad, en lazo abierto, basado en el modelo dinámico no lineal en el espacio de articulación. Se ha considerado el desempeño de la solución del sistema robótico, a través de Matlab®/Simulink® y con el uso de la S-Function, como el criterio de referencia para comparar los resultados obtenidos de la codificación de los solucionadores. Además, se hacen inferencias de los tiempos de simulación de cada uno de los algoritmos. El análisis de los resultados permite considerar la implementación del código de los métodos numéricos para propósitos de simulación, que contribuyan a optimizar los tiempos de simulación y costo computacional.

Solucionadores, Métodos numéricos, Sistemas robóticos

Citation: TORRES-DEL CARMEN, Felipe de Jesús, JARAMILLO-HERNÁNDEZ, Ricardo, DÍAZ-SÁNCHEZ, Arnaldo and NÚÑEZ-ALTAMIRANO, Diego Alfredo. Comparison of numerical methods in code as solvers for simulation of robotic systems. Journal Applied Computing. 2020. 4-15:1-8.

* Correspondence of the Author (Email: fdj.torres@ugto.mx)
† Researcher contributing as first author.

Introduction

The use of technology in industrial automation processes has been focused on the application of robotic systems. Rigid manipulative robots in particular have been the subject of research and development for more than two decades. Furthermore, in higher-level educational programs related to robotics, they have been distinguished by the synthesis and analysis of robotic arms of various degrees of freedom (g.d.l.) such as the type PUMA, SCARA, SCORBOT, etc.

The cost of a rigid manipulator is not affordable for the purposes of teaching-learning processes. Therefore, the use of simulations allows to understand the dynamic behavior of a robotic system based on its mathematical model, which is a system of nonlinear ordinary differential equations (ODE), whose solution is carried out through the implementation of method algorithms. numerical, commonly known as solvers.

The most widely used simulation platform in robotics issues is Matlab® / Simulink®, where the solvers that the platform itself has pre-installed are applied; even the simulations in Matlab® / Simulink® have served as a reference for simulations carried out through other software. In this way, in (Velarde *et al*., 2010) a complete simulation of a 5 g.d.l. robot is made. for trajectory tracking in Matlab® / Simulink®; (Gouasmi *et al*., 2012) presents the simulation of the movement of a 2-R robot with a revolutionary configuration, where it is simulated through Solidworks® and compared with the simulation in Matlab® / Simulink®; en (Alshamasin *et al*., 2012) simulates the dynamics of a SCARA robot by means of solid-dynamics software and is verified by the simulation run in Matlab® / Simulink®. Over time, various simulation softwares have been developed, however, Matlab® / Simulink® continues to be the simulation platform used, for example, in (Domazetovska *et al*., 2019), (Cheng *et al*., 2019), (Yoo, 2019), (Orta, 2019) and (Alwan *et al*., 2019).

It is important to note that the configuration of the block diagram in Simulink® allows the use of various solvers, fixed-pitch and variable-pitch, which are already pre-installed in Matlab® software. Even, results of comparisons between the different solvers have been reported, as in (Eshkabilov, 2020) and (Korotchenko and Smoryakova, 2019). However, algorithms in solver code have not been compared to simulate nonlinear systems, which could optimize simulation times and hardware resources.

This work aims to compare the performance of the algorithms of numerical methods such as ODE solvers, developed and implemented in code in Matlab®, which are applied to the dynamic model of a 4 g.d.l. SCARA manipulator robot; the comparison is made taking as a reference the results obtained in simulation carried out on the Simulink® platform using S-Function and the ode45 solver.

The rest of the document is organized as follows: in the Dynamic Model section the mathematical model of the SCARA robot is described; In the ODE Solvers section the algorithms of numerical methods used in the comparison are detailed; the Simulink® Diagram section presents the block model of the SCARA robot simulation, as well as the description of the S-Function that has been used. The Results section shows and analyzes the comparisons and, finally, the conclusions are presented in the corresponding section.

SCARA robot dynamic model

The robotic system to be used in the methodology of this work is a rigid manipulator SCARA (Selective Compliance Articulated Robot Arm) of 4 g.d.l. According to Fig. 1, the arm consists of 3 rotational joints θ_1, θ_2 and θ_3; as well as a translational joint, denoted by d_4.

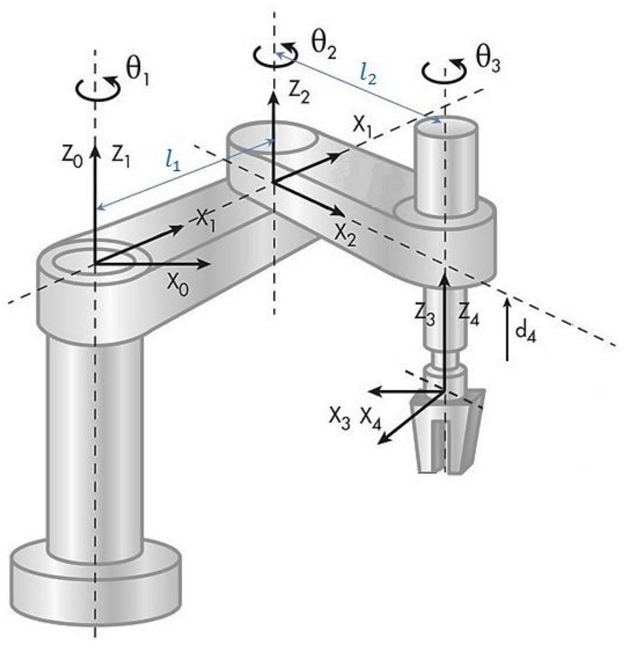


Figure 1 SCARA robot of 4 g.d.l.
Source: Our elaboration

The dynamics of the movement in the articulation space of this manipulator, it is possible to know it through its mathematical model, which is obtained through the Euler-Lagrange methodology, where it has been assumed that the angular positions θ_1, θ_2 and θ_3 will be the generalized coordinates q_1, q_2 and q_3, respectively, and the vertical displacement d_4 of the end effector will be the generalized coordinate q_4. Thus, the dynamic model of the robot, in its matrix representation, is given by:

$$M_i(q_i)\ddot{q}_i + C_i(q_i, \dot{q}_i)\dot{q}_i + B_i\dot{q}_i + g_i(q_i) = \tau_i \qquad (1)$$

Where M_i (q_i) $\in R^n$ is the inertia matrix, C_i (q_i, q_i) $\in R^{(n\times n)}$ is the Coriolis matrix and centrifugal forces, Bi is a diagonal matrix of the viscous friction coefficients of each joint, g_i $q_i \in R^n$ is the vector of gravitational forces, $\tau_i \in R^n$ is the vector of input torques or torques. So,

$$\begin{bmatrix} M_{11} & M_{12} & M_{13} & M_{14} \\ M_{21} & M_{22} & M_{23} & M_{24} \\ M_{31} & M_{32} & M_{33} & M_{34} \\ M_{41} & M_{42} & M_{43} & M_{44} \end{bmatrix} \begin{bmatrix} \ddot{q}_1 \\ \ddot{q}_2 \\ \ddot{q}_3 \\ \ddot{q}_4 \end{bmatrix} + \begin{bmatrix} C_{11} & C_{12} & C_{13} & C_{14} \\ C_{21} & C_{22} & C_{23} & C_{24} \\ C_{31} & C_{32} & C_{33} & C_{34} \\ C_{41} & C_{42} & C_{43} & C_{44} \end{bmatrix} \begin{bmatrix} \dot{q}_1 \\ \dot{q}_2 \\ \dot{q}_3 \\ \dot{q}_4 \end{bmatrix} + \begin{bmatrix} B_1 & 0 & 0 & 0 \\ 0 & B_2 & 0 & 0 \\ 0 & 0 & B_3 & 0 \\ 0 & 0 & 0 & B_4 \end{bmatrix} \begin{bmatrix} \dot{q}_1 \\ \dot{q}_2 \\ \dot{q}_3 \\ \dot{q}_4 \end{bmatrix} + \begin{bmatrix} 0 \\ 0 \\ 0 \\ m_4 g \end{bmatrix} = \begin{bmatrix} \tau_1 \\ \tau_2 \\ \tau_3 \\ \tau_4 \end{bmatrix} \qquad (2)$$

Where:

$$\begin{aligned} M_{11} = I_1 + I_2 + I_3 + I_4 + m_1 lc_1^2 + m_2 l_1^2 \\ + m_2(lc_2^2 + 2l_1 lc_2 \cos q_2) \\ + (m_3 + m_4)(l_1^2 + l_2^2 \\ + 2l_1 l_2 \cos q_2) \end{aligned}$$

$$\begin{aligned} M_{12} = I_2 + I_3 + I_4 + m_2(lc_2^2 + l_1 lc_2 \cos q_2) \\ + (m_3 + m_4)(l_2^2 + l_1 l_2 \cos q_2) \end{aligned}$$

$$M_{13} = M_{23} = M_{33} = I_3 + I_4$$

$$M_{14} = M_{24} = M_{34} = 0$$

$$M_{22} = I_2 + I_3 + I_4 + m_2 lc_2^2 + (m_3 + m_4) l_2^2$$

$$M_{44} = m_4$$

$$\begin{aligned} C_{11} = -m_2 l_1 lc_2 \sin q_2 \dot{q}_2 \\ - (m_3 + m_4) l_1 l_2 \sin q_2 \dot{q}_2 \end{aligned}$$

$$\begin{aligned} C_{12} = -m_2 l_1 lc_2 \sin q_2 (\dot{q}_1 + \dot{q}_2) \\ - (m_3 + m_4) l_1 l_2 \sin q_2 (\dot{q}_1 + \dot{q}_2) \end{aligned}$$

$$\begin{aligned} C_{21} = m_2 l_1 lc_2 \sin q_2 \dot{q}_1 \\ + (m_3 + m_4) l_1 l_2 \sin q_2 \dot{q}_1 \end{aligned}$$

$$\begin{aligned} C_{13} = C_{14} = C_{22} = C_{23} = C_{24} = C_{31} = C_{32} \\ = C_{33} = C_{34} = C_{41} = C_{42} = C_{43} \\ = C_{44} = 0 \end{aligned}$$

ODE solvers

Numerical methods are algorithms that allow obtaining non-trivial solutions. In particular, a dynamic system such as the SCARA robot is modeled by means of a set of second order nonlinear ordinary differential equations. Therefore, it is necessary to integrate the system of equations twice to obtain the function that represents the angular and translational displacement of each of the robot's joints. This is achieved through the implementation of numerical method algorithms known as solvers.

In the literature there is a great diversity of solvers: those called fixed-step solvers in which the increment of the time vector is a constant number and in each increment the set of ODE is solved; and the so-called variable pitch solvers, in which the increase in the time vector is variable and the set of ODE is also solved at that instant of time given by each increment.

Euler's method

Let be an initial value problem,

$$\frac{dy}{dt} = f(t,y), y(a) = y_0 \quad (3)$$

To approximate its solution on the interval [a, b], it is necessary to divide the interval into N equal subintervals, such that $t_i = a + ih$ for i = 0,1,…, N with $h = \frac{(b-a)}{N}$.

Where h is known as the step size.

Assuming that y (t) is twice continuously differentiable in [a, b], the Euler integration method is given by (Kharab and Guenther, 2019):

$$y_{i+1} = y_i + hf(t_i, y_i), i = 0,1, \dots, N-1 \quad (4)$$

The code made for this solver is developed in MATLAB® language.

Runge-Kutta method of 4th order

The fourth order Runge-Kutta integration method (RK4) is given by (Kharab and Guenther, 2019):

$$y_{i+1} = y_i + \frac{h}{6}(k_1 + 2k_2 + 2k_3 + k_4) \quad (5)$$

For $i = 0,1, \dots, N-1$, con:

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

$$k_2 = f\left(t_i + \frac{h}{2}, y_i + \frac{h}{2}k_1\right)$$

$$k_3 = f\left(t_i + \frac{h}{2}, y_i + \frac{h}{2}k_2\right)$$

$$k_4 = f(t_i + h, y_i + hk_3)$$

This algorithm has been implemented in MATLAB® language.

Matlab® ode45 solver

Matlab® software has several differential equation solvers pre-installed, including ode45, which is a numerical method based on the 4th and 5th order Runge-Kutta with an adjustment to make a variable step size, which uses a large / small step depending on the function to be solved, if it is smooth enough (continuously differentiable) (Yang *et al*., 2020).

In (Eshkabilov, 2020) it is indicated that the ode45 solver is the one recommended for most ODE problems, therefore, it is the solver that has been chosen to be considered as a reference in the simulations carried out with the other numerical methods.

Simulation in Matlab® / Simulink®

Based on the literature consulted, the simulation platform commonly used for robotic system applications is Simulink® from Matlab®, which also serves as a reference to compare simulations carried out in other software.

In this work, the simulation of the angular behavior of the joints of a SCARA-type robot manipulator was carried out, based on its non-linear dynamic model in the joint space, through the coding of the S-Function Level-1 block of Matlab® / Simulink®. The variable step ode45 solver and a simulation time of 10 seconds have been configured.

An S-Function is a Simulink® one-block computer language description, written in MATLAB®, C, C ++, or Fortran. S-Functions are compiled as MEX files, which are dynamically linked subroutines that the interpreter automatically loads and executes. In particular, the S-Function Level-1 has been encoded in MATLAB® language, therefore, an .m file has been created that is called by the S-Function block. The code used is detailed in Annex A. Fig. 2 shows the simulation scheme carried out in Simulink®.

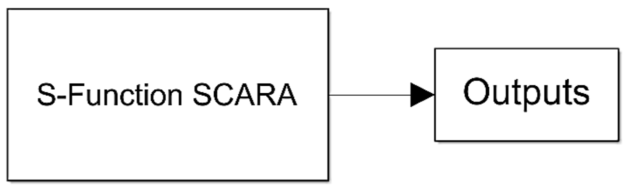
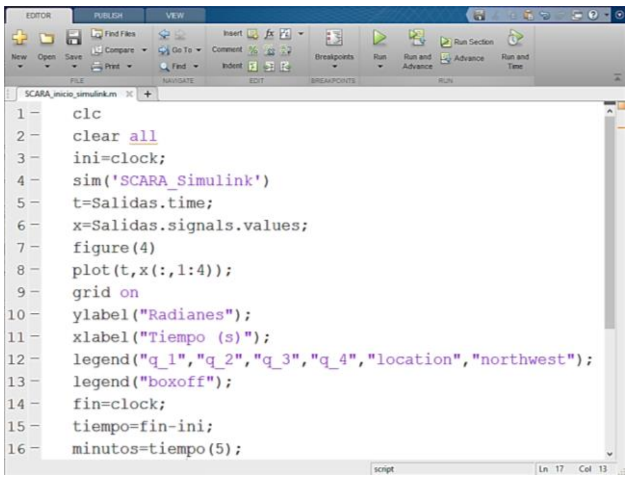


Figure 2 Simulation diagram in Simulink®
Source: Our elaboration

The S-Function Level-1 block allows avoiding the use of different blocks such as integrator, constant, step, gain, etc. and their respective connections, which makes the simulation diagram long and confusing. At the output of the S-Function Level-1 block, a block has been placed to save the value of the variables and the data can be accessed from the Matlab workspace. It is important to note that the option to save the data of the output variables will allow to graph the results of the simulations with the other solvers to make the comparison.

In addition to the block diagram in simulink, programming lines were coded in MATLAB® language to indicate the initial count of the time the simulation takes when it is sent to run, as well as the time at the end and lines of code for the generation of the corresponding graphs, as shown in Fig. 3.



```
clc
clear all
ini=clock;
sim('SCARA_Simulink')
t=Salidas.time;
x=Salidas.signals.values;
figure(4)
plot(t,x(:,1:4));
grid on
ylabel("Radianes");
xlabel("Tiempo (s)");
legend("q_1","q_2","q_3","q_4","location","northwest");
legend("boxoff");
fin=clock;
tiempo=fin-ini;
minutos=tiempo(5);
```

Figure 3. Code to start the simulation in Simulink®
Source: Our elaboration

Simulation results

The results of the simulations carried out of the dynamic behavior of the SCARA robot are presented below. Note that the simulation in Simulink® is taken as the reference path for the other simulations.

The trajectory of each of the robot's joints is plotted, starting from the introduction of excitation input pairs to initiate the movement of the robotic arm. The pairs used are given in Table 1.

	Input torque
q_1	$10sin(0.002\pi t)\ Nm$
q_2	$1+5cos(20\pi t)\ Nm$
q_3	$1+2sin(2\pi t)\ Nm$
q_4	$30\ N$

Table 1 Input pairs to the robotic system
Source: Our elaboration

In all cases, the initial positions of the joints is 0 rad, in addition, the simulation time is 10 seconds.

The simulation in Simulink® was performed with the variable pitch ode45 solver, configured with a maximum pitch and relative tolerance of 1e-3 for both. The RK4 and Euler solvers were carried out with an integration step of 0.01 seconds.

In Fig. 4-7 the trajectories of the joints q_1, q_2, q_3 and q_4 are shown; where the black line represents the result of the simulation carried out in Simulink®, the dotted blue line is the simulation of the 4th order Runge-Kutta solver, the red line corresponds to the code simulation of the ode45 solver (preloaded in Matlab®), finally, with a dotted green line is the trajectory that results from the simulation with the Euler solver.

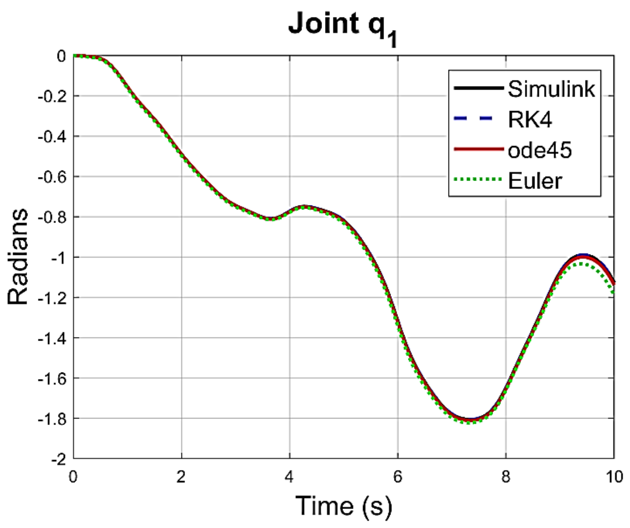


Figure 4 Path of joint q_1 with all solvers
Source: Our elaboration

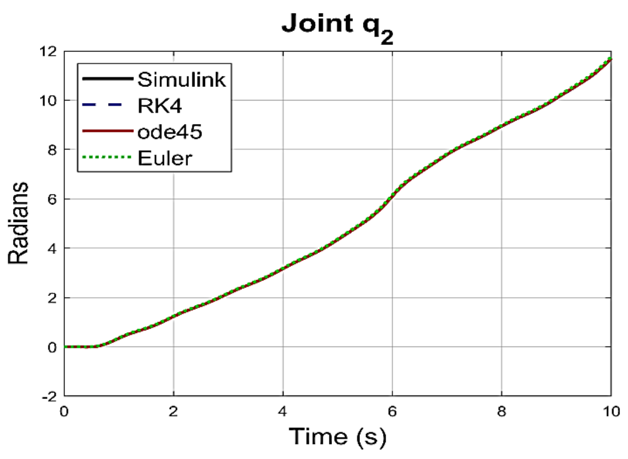


Figure 5 Path of joint q_2 with all solvers
Source: Our elaboration

All the solvers used obtain the same results in joints q_2-q_4 of the SCARA robot, with the exception of joint q_1. This is detailed in Fig. 8, which is a close-up of this trajectory in the time period with the greatest difference between the solvers.

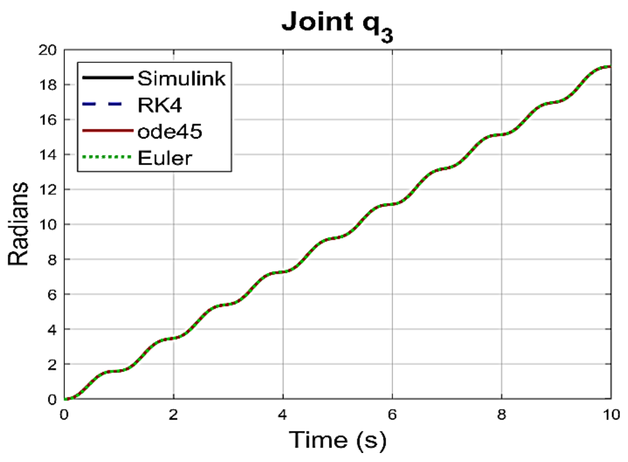


Figure 6 Path of joint q_3 with all solvers
Source: Our elaboration

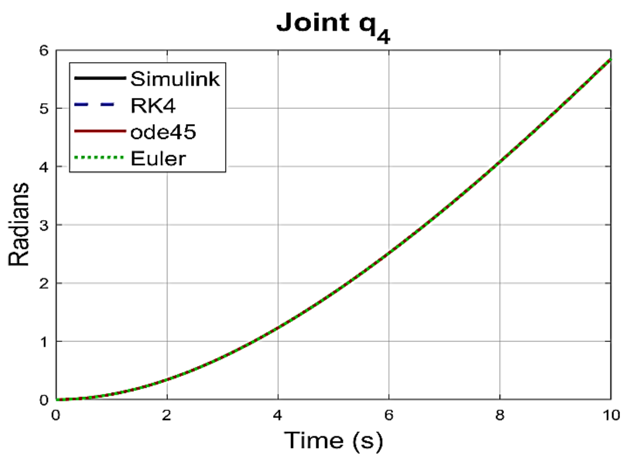


Figure 7 Path of joint q_4 with all solvers
Source: Our elaboration

It is observed that in joint q_1, the RK4 solver correctly follows the path given by the simulation carried out in Simulink®; the other solvers used (ode45 in code and Euler in code) move away from the reference path, this is due to the integration step, which can be reduced to achieve a better approach to the solution offered by Simulink®, however, would have an impact on runtime.

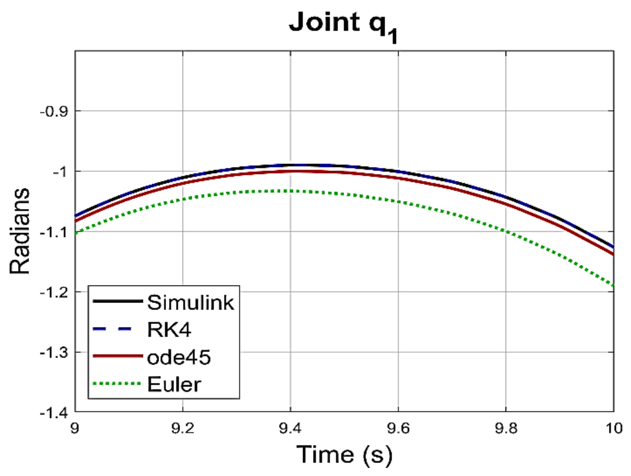


Figure 8 Zoom in on the path of joint q_1 with all solvers
Source: Our elaboration

In addition to the comparison of the trajectories, the analysis is done based on the execution time that each simulation takes to weight the solver with the best performance. For this purpose, the methodology that was followed was to carry out 10 simulations with each of the solvers, then obtain the average of the execution times, these are shown in Table 2.

	Euler code	**ode45 code**	**RK4 code**	***Simulink®***
1	0.75	0.81	0.78	2.76
2	0.81	0.76	0.75	2.71
3	0.79	0.80	0.76	2.64
4	0.83	0.83	0.8	2.72
5	0.81	0.78	0.83	2.73
6	0.83	0.83	0.79	2.73
7	0.78	0.80	0.79	2.69
8	0.75	0.78	0.76	2.70
9	0.77	0.85	0.82	2.69
10	0.87	0.83	0.78	2.93
Average	0.799	0.807	0.786	2.73

Table 2 Solver execution times in seconds
Source: Our elaboration

Conclusions

Various algorithms of numerical methods, known as solvers, can be used to simulate the dynamic behavior of robotic systems such as the SCARA robot.

In the simulations carried out as teaching-learning activities, widely known software such as Matlab® / Simulink® is commonly used, which uses preloaded solvers in the software installation. However, the code implementation of the numerical methods: Rnge-Kutta of 4th order and the Euler method have shown a correct performance in the developed simulation compared to the simulation done in Simulink®.

According to the graphs of the trajectories, the solver that fully followed the Simulink® simulation is the Runge-Kutta of 4th order in code. In addition, this same solver was the one that presented the best performance according to the execution time, it even had an average execution time less than the time of the Euler method, which is known for its coding simplicity.

Acknowledgments

The authors want to thank the support provided by the University of Guanajuato, Campus Irapuato-Salamanca Engineering Division, for the realization of this project.

References

Alshamasin, M. S., Ionescu, F., & Al-Kasasbeh, R. T. (2012). Modelling and simulation of a SCARA robot using solid dynamics and verification by MATLAB/Simulink. *International Journal of Modelling, Identification and Control, 15*(1), 28-38.

Alwan, H. M. y Rashid, Z. H. (2019). Motion Control of Three Links Robot Manipulator (Open Chain) with Spherical Wrist. *Al-Khwarizmi Engineering Journal, 15*(2), 13-23.

Cheng, L. I. U., Guo-Hua, C. A. O. y Yong-Yin, Q. U. (2019, October). Motion simulation of Delta parallel robot based on Solidworks and Simulink. In *2019 IEEE 3rd Advanced Information Management, Communicates, Electronic and Automation Control Conference (IMCEC)* (pp. 1683-1686). IEEE.

Domazetovska, S., Mickoski, H. y Djidrov, M. (2019). Kinematic modeling and analysis of serial manipulator. *Mechanical Engineering–Scientific Journal*.

Eshkabilov, S. L. (2020). *Practical MATLAB Modeling with Simulink: Programming and Simulating Ordinary and Partial Differential Equations*. Apress.

Eshkabilov, S. L. (2020). *Practical MATLAB Modeling with Simulink: Programming and Simulating Ordinary and Partial Differential Equations*. Apress.

Gouasmi, M., Ouali, M., Fernini, B., & Meghatria, M. H. (2012). Kinematic modelling and simulation of a 2-R robot using solidworks and verification by MATLAB/Simulink. *International Journal of Advanced Robotic Systems, 9*(6), 245.

Kharab, A., & Guenther, R. (2019). *An introduction to numerical methods: a MATLAB® approach*. CRC press.

Korotchenko, A. G. y Smoryakova, V. M. (2019). On a comparison of several numerical integration methods for ordinary systems of differential equations. In *International Conference on Numerical Computations: Theory and Algorithms* (pp. 406-412). Springer, Cham.

Orta Hernández, M. (2019). Modelado, control y simulación de manipuladores aéreos incluyendo modelos de sensores.

Velarde-Sanchez, J. A., Rodriguez-Gutierrez, S. A., Garcia-Valdovinos, L. G., & Pedraza-Ortega, J. C. (2010, February). 5-DOF manipulator simulation based on MATLAB-Simulink methodology. In *2010 20th International Conference on Electronics Communications and Computers (CONIELECOMP)* (pp. 295-300). IEEE.

Yang, W. Y., Cao, W., Kim, J., Park, K. W., Park, H. H., Joung, J., ... & Im, T. (2020). *Applied numerical methods using MATLAB*. John Wiley & Sons.

Yoo, D. S. (2019). 3D Modeling and Balancing Control of Two-link Underactuated Robots using Matlab/Simulink. *Journal of information and communication convergence engineering*, *17*(4), 255-260.