ISSN No:-2456-2165 https://doi.org/10.38124/ijisrt/25apr256

Solution of Damped Spring Vibration Model of Motor Vehicles with MATLAB Application

(Model of Motor Vehicle Damped Spring Vibration)

Gilang Nitiasya Fawzi[1]; Yani Ramdani[2]

[1,2]Mathematics Study Program
The Faculty of Mathematics and Natural Sciences
Universitas Islam Bandung, Indonesia

Publication Date: 2025/04/17

Abstract: This study analyzes the application of the Runge Kutte 4 method, Runge Kute Gill, and Milne Method in two-wheeled motor vehicles. The study results show that: (1) Performance of the Milne Method: The Milne method is effective for solving damped oscillatory systems with small time steps and provides good results, although with a slight increase in error compared to the Runge Kutte 4 method. (2) Advantages The Milne method shows more efficiency in the number of function evaluations per step. It is suitable for applications requiring more significant time steps without requiring accuracy as high as the Runge Kutte 4 method. (3) Disadvantages The Milne method has lower accuracy than the Runge Kutte 4 method, especially for small time steps and requires an initial step using another method, adding to the complexity of the Implementation.

Keywords*: Analysis; Runge Kutta; Runge Kutta Gill; Milne Method; Two-Wheeled Motorized Vehicles.*

How to Cite: Gilang Nitiasya Fawzi; Yani Ramdani. (2025). Solution of Damped Spring Vibration Model of Motor Vehicles with MATLAB Application. *International Journal of Innovative Science and Research Technology*, 10(4), 314-327. https://doi.org/10.38124/ijisrt/25apr256.

I. INTRODUCTION

Motorized vehicles are a means of transportation that is very much needed by the community. Comfort and safety of travel are among the most essential criteria when choosing a vehicle. High speed demands the vibration-damping system the most. Thus, motorized vehicles require springs that function as shock absorbers on the road and wheel vibrations so that they are not directly transmitted to the vehicle body. Springs are components that function to receive dynamic loads and have elastic properties. Springs are also referred to as flexible objects; they can return to their original position even though they have received external force. Vibrations that occur if a load is attached or hung to a spring are one example of vibration.

Vibrations can occur if a system is disturbed from its stable equilibrium position. Vibrations in springs are divided into simple harmonic motion and damped harmonic motion. Simple harmonic motion is the vibration of an object that occurs continuously, and there are no resistance or damping factors. Simple harmonic motion can also be interpreted as a system that vibrates with a restoring force directly proportional to its deviation's negative. The restoring force is a force that works in the direction of returning the mass to its equilibrium position. In reality, vibrations in an object will not occur continuously because there are resistance factors in the form of air friction and internal factors that cause the vibrations to decrease over time and eventually stop slowly; such vibrations of objects are usually referred to as damped harmonic motion [1]. This phenomenon can be formed in a mathematical model.

Mathematics is a deductive science. The concepts in mathematics are hierarchical, structured, logical, and systematic, ranging from the simplest to the most complex [2]. Mathematics has a vital role in formulating real-life problems into a model, such as the problem of vibrations in springs. The vibration model in a damped spring is an ordinary differential equation that can be solved using analytical and numerical methods. An analytical method is a method that can provide actual results (exact solutions) and solve some mathematical issues. At the same time, the numerical method is an approach to the exact solution of a mathematical problem. There are two methods to find solutions to differential equations numerically: the one-step and multistep methods. This paper discusses the solution of motor vehicles' damped spring vibration model using the Runge-Kutta Method, Runge-Kutta Gill Method, and Milne Method. Runge-Kutta Gill Method is a family of fourth-order Runge-Kutta methods, which include one-step methods where the next step solution depends on the previous solution value. Milne Method has many steps that can be used to find the approximate value of ordinary

ISSN No:-2456-2165 https://doi.org/10.38124/ijisrt/25apr256

differential equations and have predictor and corrector values.

II. LITERATURE REVIEW

A. Model Osilasi

Oscillation occurs when a system receives disturbances from its equilibrium position with repetitive motion. Many oscillations are easily recognized, for example, a small boat swinging up and down, a clock pendulum swinging left and right, and a vibrating guitar string. A system that exhibits simple harmonic motion is an object attached to a spring [6]. Figure 1 shows an ordinary spring that withstands compression and elongation and hangs vertically from a fixed point. The spring's lower end is tied to an object with mass m. The mass m is considered so large that the mass of the spring can be ignored. If the object is pulled down a certain distance and then released, the object will move. The following will determine the motion of the mechanical system, assuming the object moves vertically. To achieve this goal, consider the forces acting on the object during its movement. These forces will lead to a differential equation, and by solving this equation, the displacement will be obtained as a function of time. The downward direction is chosen as the positive direction, the force acting downward is considered positive, and the upward force is considered harmful [6].

The force acting on an object is the gravitational force (g), $F_1 = mg$ with m being the object's mass and g (980 cm/sec2) being the acceleration due to gravity. Next, consider the spring force F_2 which works on objects, namely, $F_2 = -ks$ with s being the vertical displacement of the object (note that the upper end of the spring is stationary), while the constant k is called the spring modulus. If the object is at rest, then the gravitational force and the spring force are in balance, and the resultant force is zero so that the following mathematical form is obtained,

$$F_1 + F_2 = mg - ks_0 = 0 \quad (1)$$

With s_0 is the change in the length of the spring when the object is at rest, namely the static equilibrium position. The object's displacement is measured from the static equilibrium position $y = 0$, symbolized by $y = y(t)$, which represents time, with the downward direction as the positive direction, as shown in Figure 1. According to Hooke's law, this shift causes an additional force of *-ky* that works on objects. Thus, the resultant force acting on the object's position $y(t)$ is

$$F_1 + F_2 - ky = -ky \quad (2)$$

Suppose the damping of a system is so tiny that it can be neglected so that equation (2) is the resultant of all the forces acting on the object. The differential equation is obtained using Newton's second law with the force being $m \times y'' = F$. In this condition, the resultant of all forces acts with arbitrary time. In this case, the acceleration is a second-order ordinary differential equation that has the form $y'' = \frac{d^2y}{dt^2}$ and the consequent is given by equation (2) then,

$$my'' + ky = 0 \quad (3)$$

so that a linear differential equation with constant coefficients governs the system's motion. The damping force has a direction opposite to the object's motion at that time, and it is assumed that this force is proportional to the velocity of $y' = \frac{dy}{dt}$the object. So, the damping force is in the form of$F_3 = -cy'$. The damping constant has a positive value. The resultant force acting on the object is$F_1 + F_2 + F_3 = -ky - cy$. So, according to Newton's second law, the form is obtained,

$$my'' + ky + cy' = 0 \quad (4)$$

and it is seen that a linear differential equation with constant coefficients determines a damped mechanical system. The free oscillations of an object on a spring can be seen in Figure 1. Now assume the presence of a variable force$r(t)$ acting on the system. The differential equation relating to this situation is obtained from equation (4) by adding the forces of$r(t)$; this addition produces equation (5)

$$my'' + cy' + ky = r(t) \quad (5)$$

$r(t)$ called input or driving force; the solution is called output as the system's response to the driving force. The resulting motion is called forced motion, which is different from free motion related to equation (5), namely motion without external motion $r(t)$. Periodic input is a special case of interest, sinusoidal, with the form of $r(t) = F_0 \cos(\omega t)$ with $(F_0 > 0, \omega > 0)$. So, the differential equation that appears now is equation (6)

$$my'' + cy' + ky = F_0 cos\omega t \quad (6)$$

The suspension system on vehicles, especially motorcycles, functions as a shock absorber when the car is moving. The suspension also maintains the motorcycle's balance, making it easy to control. Thus, the main task of the suspension is to make the rider feel comfortable with reduced shocks caused by road contours, reduce vibrations due to engine work, and stabilize the vehicle when turning. The research of Makoto Yokoyama and Masato Yamagishi [8] explains how the active steering system scheme with one motor and the handlebar suspension is composed of springs and dampers, as in the following picture,

ISSN No:-2456-2165 https://doi.org/10.38124/ijisrt/25apr256

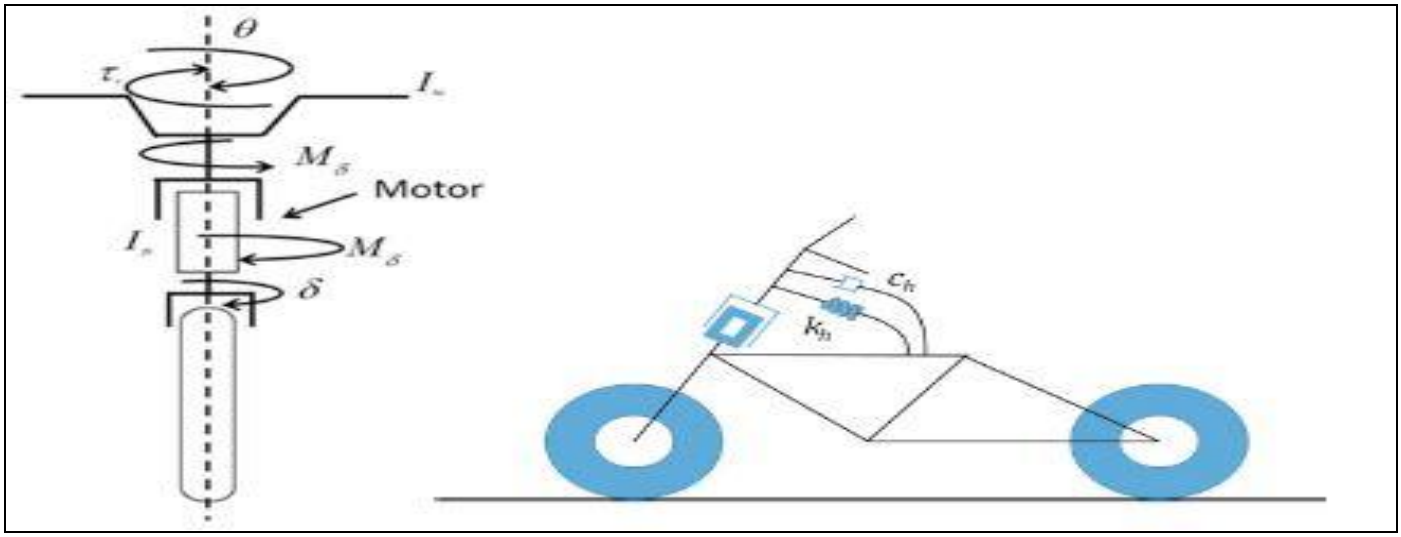

Fig 1: Spring System on Motorbike
(Source: Yokoyama and Yamagishi, 2022)

You can use the Runge-Kutta method to find the solution to equation (6). The Runge-Kutta method is a method that provides greater accuracy of results and does not require derivatives of a function. The general form of the Runge-Kutta method is equation (7)

$$x_{i+1} = x_i + \Phi(t_i, x_i, h) \quad (7)$$

With $\Phi(t_i, x_i, h)$ is an incremental function, which is the average slope on an interval and is used to extrapolate from old values x_i to new value x_{i+1}s throughout the interval *h*. The addition function can be written in general form in equation (8),

$$\Phi = a_1 k_1 + a_2 k_2 + \cdots + a_n k_n \quad (8)$$

If a differential equation is known to have the form of $\frac{dx}{dt} = f(x_i, t_i)$ then based on the Runge Kutta method approximation with a_i is a constant whit k_i for $i = 1, 2, 3, \ldots, n$ is,

$$k_1 = f(t_i, x_i) \quad (9)$$

$$k_2 = f(t_i + p_i h, x_i + q_{11} k_1 h) \quad (10)$$

Next can be obtained

$$k_3 = f(t_i + p_i h, x_i + q_{21} k_1 h + q_{22} k_2 h)$$

$$k_n = f(t_i + p_{n-1} h, x_i + q_{n-1,2} k_1 h + q_{n-1,2} k_2 h + \cdots + q_{n-1,n-1} k_{n-1} h)$$

Where the function $f(t_i, x_i)$ represents the ordinary differential equation to be approximated, *t* is the time variable, *x* is the value of the approximate solution at that time, and h is the time step or interval length. The values of *p* and *q* are constants. The value of *k* indicates the sequential relationship. The value k_1 appears in the equation of k_2 both, which also appears in the equation of k_3 and so on. This sequential relationship makes the Runge-Kutta method efficient for computer calculations [9]. Several types of Runge-Kutta methods depend on the value of *n* (order) used. For example, for $n = 1$ called the first-order Runge-Kutta method, also called the Euler method, which is obtained from $k_1 = f(t_i, x_i)$equation (9)

$$\Phi = a_1 k_1 = a_1 f(t\_i, x\_i)$$

For$a_1 = 1$ then the equation becomes,

$$x_{i+1} = x_i + \mathrm{f}(t_i, x_i) h$$

In the Runge-Kutta method, after the value of n is determined, the values of a, p, and q are then sought by equating equation (8) with the terms of the Taylor series in the form of a function$f(x)$:

$$f(x + h) = f(x) + h \,.\, f'(x) + \frac{h^2}{2!} \,.\, f''(x) + \frac{h^3}{3!} \,.\, f'''(x) + \cdots$$

With $f'(x), f''(x)$, and so on are the first, second and so on derivatives of the function *f*(*x*) at point *x*. The methods often used are the second-order Runge-Kutta method (RK2), the third-order Runge-Kutta method (RK3) and the fourth-order Runge-Kutta method (RK4). The second-order Runge-Kutta method has the following form,

$$x_{i+1} = x_i + (a_1 k_1 + a_2 k_2) h \quad (11)$$

With values and as in equations (9) and (10) and for values a_1, a_2, p_1, and q_{11} evaluated by equating equation (11) with the second-order Taylor series of the function f(x) is:

$$f(x + h) = f(x) + h \,.\, f'(x) + \frac{h^2}{2} \,.\, f''(x)$$

The value is obtained $a_1 + a_2 = 1$ and $a_2 p_1 = a_2 q_{11} = \frac{1}{2}$ by selecting $a_1 = \frac{1}{2}$, $a_2 = \frac{1}{2}$, and $p_1 = q_{11} = 1$. Next, substitute these values into equation (11) so that the second-order Runge-Kutta method formula is obtained as follows.

ISSN No:-2456-2165 https://doi.org/10.38124/ijisrt/25apr256

$$x_{i+1} = x_i + \frac{1}{2}(k_1 + k_2)h$$

With

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

and

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

The third-order Runge-Kutta method is derived in the same way as the second-order Runge-Kutta method for *n* = 3. The result of this derivation is six equations with eight unknowns. Therefore, two unknowns must be determined first to obtain the remaining six unknowns. The third-order Runge-Kutta method has the following form,

$$x_{i+1} = x_i + \frac{1}{6}(k_1 + 4k_2 + k_3)h$$

With

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

And

$$k_3 = f(t_i + h, x_i + k_1 h + 2k_2 h)$$

The fourth-order Runge-Kutta method is the most accurate compared to the second-and third-order Runge-Kutta methods. Therefore, the fourth-order Runge-Kutta method is often used to solve a differential equation. The fourth-order Runge-Kutta method (RK4) is derived similarly to the second-order Runge-Kutta method (RK2) for the value of *n* = 4. The fourth-order Runge-Kutta method has the following form,

$$x_{i+1} = x_i + \tfrac{1}{6}(k_1 + 2k_2 + 2k_3 + k_4)h \quad (12)$$

with

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

$$k_2 = f(t_i + \frac{1}{2}h, x_i + \frac{1}{2}k_1 h)$$

$$k_3 = f(t_i + \frac{1}{2}h, x_i + \frac{1}{2}k_2 h)$$

$$k_4 = f(t_i + h, x_i + \frac{1}{2}k_3 h)$$

This fourth-order Runge-Kutta method has a higher level of solution accuracy than the previous-order Runge-Kutta method. It is also easy to program and stable, with small errors in truncation and rounding [9].

The Runge-Kutta Gill (RKG) method belongs to the family of methods of the RK4. RKG has four intermediate calculation constants combined with other constants *r*, *s*, *t*, and you. The Runge-Kutta Gill method has the following form,

$$x_{i+1} = x_i + \tfrac{1}{6}(k_1 + k_4) + \tfrac{1}{3}(sk_2 + uk_3) \quad (13)$$

With

$$k_1 = hf(t_i, x_i)$$

$$k_2 = hf(t_i + \frac{1}{2}h, x_i + \frac{1}{2}k_1)$$

$$k_3 = hf(t_i + \frac{1}{2}h, x_i + rk_1 + sk_2)$$

$$k_4 = hf(t_i + h, x_i + tk_2 + uk_3)$$

$$r = \tfrac{\sqrt{2}-1}{2}, t = -\tfrac{\sqrt{2}}{2}, s = \tfrac{2-\sqrt{2}}{2}, u = 1 + \tfrac{\sqrt{2}}{2},$$

$i = 0,1,2,\cdots, n-1$ and n is the number of steps or iterations. Milne's method is a multistep method, so called because it requires more than one value $f(t, x(t))$. The main goal of the multistep method is to use information from several previous points $x_i, x_{i-1}, x_{i-2}, \ldots, x_1$ to calculate the estimated value x_{i+1} better. Milne's method uses a predictor equation to estimate the value x_{i+1} from $f_{i-3}, f_{i-2}, f_{i-1}$, and f_i while the corrector equation is used to calculate the value x_{i+1} better. The one-step numerical method is used to obtain the value of $f_{i-3}, f_{i-2}, f_{i-1}$, and f_i on the predictor equation. The Lagrange polynomial approximates the predictor equation in the Milne method, and the points used are $t_{i-3}, t_{i-2}, t_{i-1} t_i$ as follows: From these four points, it can be formed as follows,

$$f(x, t(x)) = \frac{(t - t_{i-2})(t - t_{i-1})(t - t_i)f_{i-3}}{(t_{i-3} - t_{i-2})(t_{i-3} - t_{i-1})(t_{i-3} - t_i)} + \frac{(t - t_{i-3})(t - t_{i-1})(t - t_i)f_{i-2}}{(t_{i-2} - t_{i-3})(t_{i-2} - t_{i-1})(t_{i-2} - t_i)} + \frac{(t - t_{i-3})(t - t_{i-2})(t - t_i)f_{i-1}}{(t_{i-1} - t_{i-3})(t_{i-1} - t_{i-2})(t_{i-1} - t_i)} + \frac{(t - t_{i-3})(t - t_{i-2})(t - t_{i-1})f_i}{(t_i - t_{i-3})(t_i - t_{i-2})(t_i - t_{i-1})}$$

$$f(x, t(x)) = \frac{1}{-6h^3}(t - t_{i-2})(t - t_{i-1})(t - t_i)f_{i-3} + \frac{1}{2h^3}(t - t_{i-3})(t - t_{i-1})(t - t_i)f_{i-2} - \frac{1}{2h^3}(t - t_{i-3})(t - t_{i-2})(t - t_i)f_{i-1} + \frac{1}{2h^3}(t - t_{i-3})(t - t_{i-2})(t - t_{i-1})f_i$$

ISSN No:-2456-2165 https://doi.org/10.38124/ijisrt/25apr256

Then $f(x, t(x))$, it is substituted into the equation.

$$x_{i+1} = x_{i-3} + \int_{t_{i-3}}^{t_{i+1}} f(x, t(x))\, dt$$

And integrated so that the predictor equation is obtained, namely equation (14).

$$x_{i+1} = x_{i-3} + \frac{4h}{3}(2f_{i-2} - f_{i-1} + 2f_i) \quad (14)$$

Then, for the corrector equation, the value is also approximated by the Lagrange polynomial. However, the points used are t_{i-1}, t_i and t_{i+1}, then it can be formed into:

$$f(x,t(x)) = \frac{(t-t_i)(t-t_{i+1})f_{i-1}}{(t_{i-1}-t_i)(t_{i-1}-t_{i+1})} + \frac{(t-t_{i-1})(t-t_{i+1})f_i}{(t_i-t_{i-1})(t_i-t_{i+1})} + \frac{(t-t_{i-1})(t-t_i)f_{i+1}}{(t_{i+1}-t_{i-1})(t_{i+1}-t_i)}$$

$$f(x,t(x)) = \frac{1}{2h^2}(t-t_i)(t-t_{i+1})f_{i-1} - \frac{1}{h^2}(t-t_{i-1})(t-t_{i+1})f_i + \frac{1}{2h^2}(t-t_{i-1})f_{i+1}$$

$f(x, t(x))$Then, it is substituted into the equation.

$$x_{i+1} = x_{i-3} + \int_{t_{i-3}}^{t_{i+1}} f(x, t(x))\, dt$$

And integrated using boundaries$[t_{i-1}, t_{i+1}]$ so that the corrector equation [10] is obtained as follows,

$$x_{i+1} = x_{i-1} + \frac{h}{3}(f_{i-1} + 4f_i + f_{i+1}) \quad (15)$$

B. MATLAB

MATLAB is software for programming, analysis, and technical and mathematical computation based on matrices. MATLAB stands for Matrix Laboratory because it can solve matrix-form calculation problems. The first version of MATLAB was released in 1970 by Cleve Moler. Initially, MATLAB was designed to solve linear algebra equation problems. Over time, this program has continued to develop in terms of function and computing performance. MathWorks Inc. now produces the programming language. It combines programming, computing, and visualization processes in an easy-to-use work environment. MATLAB also has other general advantages, such as data analysis and exploration, algorithm development, modelling and simulation, plot visualization in 2D and 3D, and the development of graphical interface applications. In higher education, MATLAB is used as a learning tool for mathematical, engineering, and science programming at introductory and advanced levels, while in the industrial world, MATLAB is chosen as a tool for research, development, and analysis of industrial products. MATLAB can be operated on Windows, Linux, and macOS operating systems. In addition, MATLAB can also be connected to other external programming languages or applications, such as C, Java, .NET and Microsoft Excel. MATLAB also provides a toolbox that can be used for specific applications, such as signal processing, control systems, fuzzy logic, artificial neural networks, optimization, digital image processing, bioinformatics, simulation and various other technologies [11].

III. RESULT AND DISCUSSION

A. Dynamic Balance of Motorcycle Spring Vibrations

The principle of Newton's law is used to conduct a dynamic equilibrium analysis in a motorcycle vehicle; the sum of the forces at equilibrium is equal to zero, with the initial step of analyzing the forces and components in the car and the load on the vehicle. In Figure 3.1, it can be seen that a motorcycle has a shockbreaker (strut), wheels and tyres. The car can be achieved during operation by considering equations (3) and (4) obtained.

$$F = K.x \text{ and } F = m.x''$$

With x is a transfer, x'speedx'' is the acceleration, K is the spring constant, C is the damping constant, and m is the object's mass.

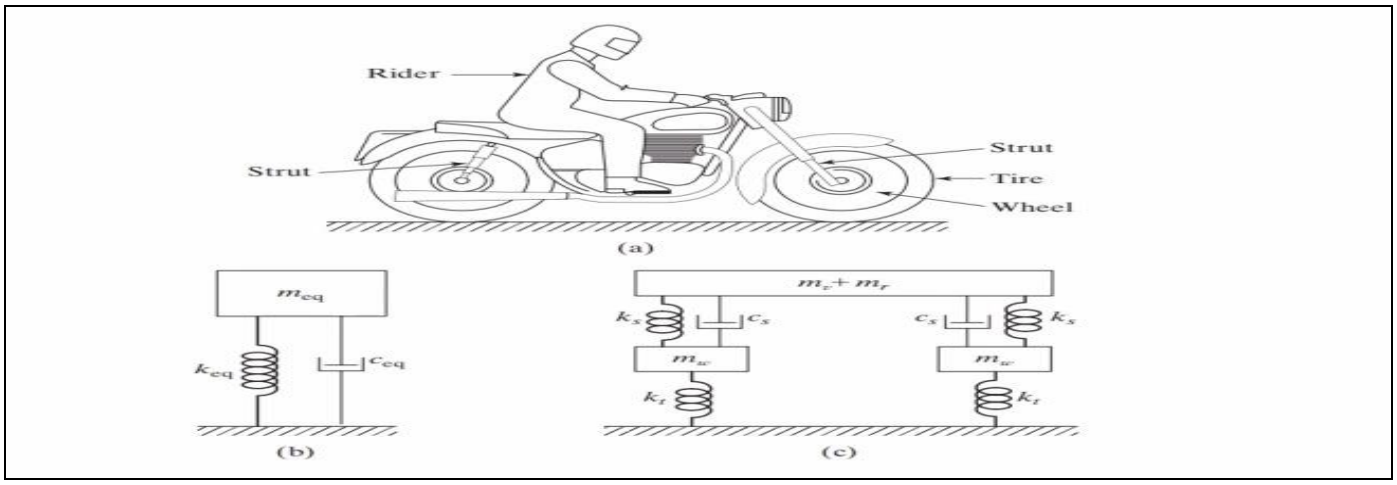

Fig 2: Motorcycle Suspension System
(Source: Kreyszig, 1993)

ISSN No:-2456-2165 https://doi.org/10.38124/ijisrt/25apr256

Given the equation

$$m_{eq} = m_v + m_r$$

with m_{eq} is the total mass, m_v is the mass of the vehicle, and m_r is the mass of the rider, then,

$$F = m.a = (m_v + m_r)x''$$

$$F = (m_v + m_r)x'' = k_s.x + c.x' + m.x''$$

$$F = k_s.x + c.x' + k_t.x + m.x''$$

Displacement is the change in the position of an object. Velocity refers to the movement of an object from one point to another in a unit of time. Furthermore, acceleration is the change in velocity in a unit of time. The concept of differential is used to calculate the change that occurs per unit of parameter, such as time. In this case, the derivative of displacement is velocity, while the derivative of velocity is acceleration. In addition, the spring force is related to the distance the spring is moved multiplied by the spring capacity constant. When the shock absorber is compressed, there is a displacement in the spring. Force is related to speed, such as displacement that was originally intact becomes shrunk in a certain time. The magnitude of the force is determined by the mass or pressure on the spring multiplied by the acceleration. The importance of force balance in vehicles is important to note. A system can be in equilibrium if the total force is equal to zero, as in equation (4). The ordinary differential equation of the spring vibration model has the form,

$$m\frac{d^2x}{dt^2} + c\frac{dx}{dt} + kx = 0$$

for m = 67,6 kg, c = 532,8 Ns/m, and k = 9102,46 N/m be obtained,

$$67{,}6\frac{d^2x}{dt^2} + 532{,}8\frac{dx}{dt} + 9102{,}46x = 0 \quad (16)$$

Assume one of the solutions of the ordinary differential equation (16) is$t = e^{kx}$, The characteristic equation is

$$(67{,}6k^2 + 532{,}8k + 9102{,}46)e^{kx} = 0$$

With $e^{kx} \neq 0$the discriminant of the equation above$D = -476{,}4865864 < 0$, then the roots of equation (16) are complex numbers, namely,

$$k_1 = -3{,}940828402 + 10{,}91428636\,i$$

$$k_2 = -3{,}940828402 - 10{,}91428636\,i$$

Then, the general solution of equation (16) is

$$x(t) = e^{-3{,}940828402t}(c_1\cos(10{,}91428636t) + c_2\sin(10{,}91428636t))$$

Assumed initial values $x(0)$ = 0.02 and $x'(0)$ = 0,05, then the value can be determined c_1 dan c_2, namely,

$$x(t) = e^{-3{,}940828402t}(c_1\cos(10{,}91428636t) + c_2\sin(10{,}91428636t)) \quad (17)$$

for t = 0, then obtained

$$x(0) = e^0(c_1\cos(0) + c_2\sin(0)$$

$$c_1 = x(0)$$

and

$$c_1 = 0.02$$

For initial requirements of $x(0) = 0$ so that$c_1 = 0.02$, and the first derivative of (17) is

$$x(t) = e^{-3{,}940828402t}(c_1\cos(10{,}91428636t) + c_2\sin(10{,}91428636t))$$

$$x'(t) = (-3{,}940828402\,c_1 + 10{,}91428636c_2)\,e^{-3{,}940828402t}\cos(10{,}91428636t) + (-10{,}91428636c_1 - 3{,}940828402c_2)\,e^{-3{,}940828402t}\sin(10{,}91428636t)$$

For$x'(0) = 0{,}05c_1 = 0.02$, and $c_2 = 0{,}0118$, then the specific solution has the form.

$$x(t) = e^{-3{,}940828402t}(0.02\cos(10{,}91428636t) + 0.0118\sin(10{,}91428636t)) \quad (18)$$

Table 1 presents data on the suspension specifications [11]. Based on Figure 2, the spring oscillates and reaches equilibrium at time t = 2, with maximum strain achieved at an x value of 0.020407, which then decreases over time.

From equation (18), the graph shown in Figure 3 is obtained.

Table 1: Data of Jupiter Z Motorcycle Suspension

No	Parameters	Symbol	Value	Unit
1	Suspension stiffness	k_1	4534,46	N/m
2	Wheel stiffness	k_2	4568	N/m
3	Suspension damping	C	532,8	Ns/m
4	Front body mass	m_v	63	Kg
5	Wheel mass	m_W	4,6	Kg

ISSN No:-2456-2165 https://doi.org/10.38124/ijisrt/25apr256

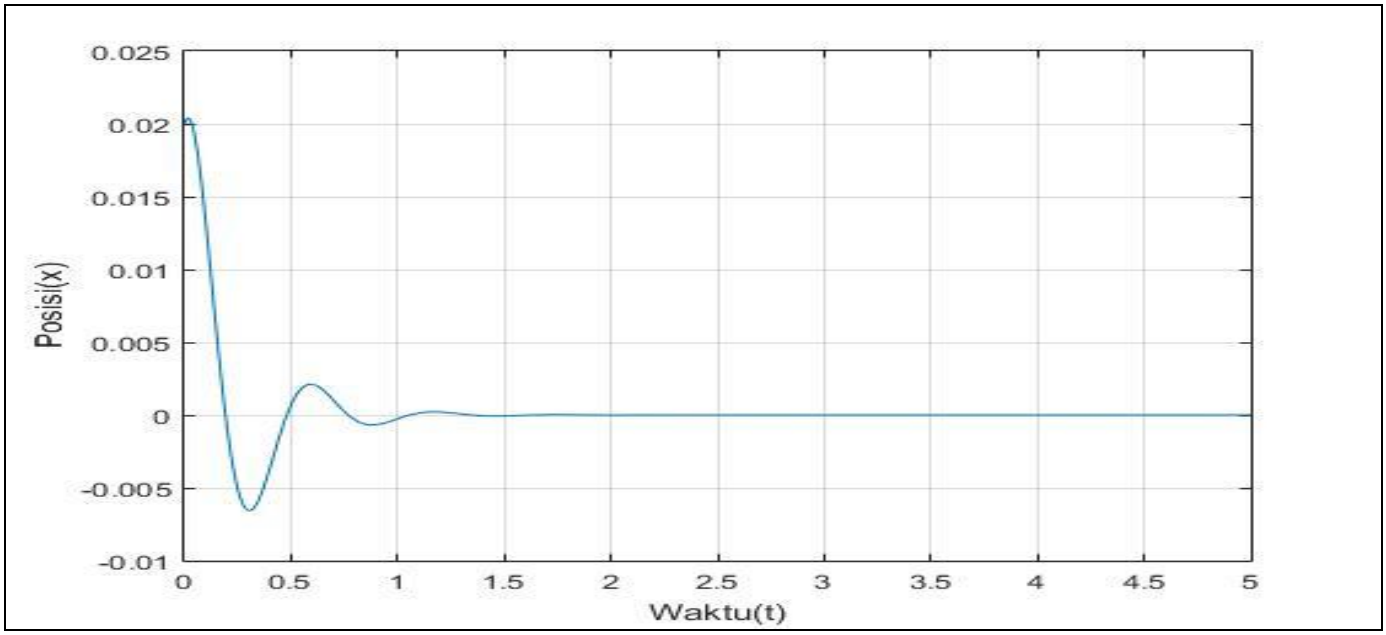

Fig 3: Vibration Graph of Special Solution

Table 2: First Point Simulation Results of Special Solution Equation

Time (t)	Position (x)
0	0.02
0.01	0.020348
0.02	0.020407
0.03	0.020198
0.04	0.019743
0.05	0.019067
0.06	0.018196
0.07	0.017155
0.08	0.015972
0.09	0.014671

B. Spring Vibration Solution with Runge-Kutta Gill Method

Uncertain systems can be model using high-order differential equations, which are often difficult to solve through analytical methods, so a code is needed to design the numerical method [8]. The frequency evaluation technique proposed [9] is a multi-step algorithm applied exponentially to first-order ordinary differential equations [10]. Determining the stability of the trajectory of a nonlinear dynamical system with respect to time is a challenge both analytically and numerically [11]. Given the second-order differential equation as follows,

$$m\frac{d^2x}{dt^2} + c\frac{dx}{dt} + kx = 0$$

Transformation of the spring vibration model into a first-order ordinary differential equation by example:

$$v = \frac{dx}{dt}, \qquad \frac{dv}{dt} = \frac{d}{dt}\cdot\frac{dx}{dt} = \frac{d^2x}{dt^2}$$

$$m\frac{d^2x}{dt^2} + kx + c\frac{dx}{dt} = 0$$

$$m\frac{dv}{dt} + kx + c.v = 0$$

$$m\frac{dv}{dt} = -kx - cv$$

$$\frac{dv}{dt} = -\frac{k}{m}x - \frac{c}{m}v$$

So, the form of the first-order differential equation system is,

$$\frac{dx}{dt} = f(t,x,v) = v$$

$$\frac{dv}{dt} = g(t,x,v) = -\frac{k}{m}x - \frac{c}{m}v$$

The Runge-Kutta Gill method has the following form,

$$x_{i+1} = x_i + \frac{1}{6}(k_1 + k_4) + \frac{1}{3}(sk_2 + uk_3) \tag{19}$$

With

$$k_1 = hf(t_i, x_i, v_i) = hv_i,\ l_1 = hg(t_i, x_i, v_i) = h(-\frac{k}{m}x_i - \frac{c}{m}v_i)$$

$$k_2 = hf(t_i + \frac{1}{2}h, x_i + \frac{1}{2}k_1, v_i + \frac{1}{2}l_1),\ k_2 = h(v_i + \frac{1}{2}l_1)$$

$$l_2 = hg(t_i + \frac{1}{2}h, x_i + \frac{1}{2}k_1, v_i + \frac{1}{2}l_1)$$

$$l_2 = h(-\frac{k}{m}(x_i + \frac{1}{2}k_1) - \frac{c}{m}(v_i + \frac{1}{2}l_1))$$

$$k_3 = hf(t_i + \frac{1}{2}h, x_i + rk_1 + sk_2, v_i + \frac{1}{2}l_2),\ k_3 = h(v_i + \frac{1}{2}l_2)$$

$$l_3 = hg(t_i + \frac{1}{2}h, x_i + rk_1 + sk_2, v_i + \frac{1}{2}l_2), l_3 = h(-\frac{k}{m}(x_i + rk_1 + sk_2) - \frac{c}{m}(v_i + \frac{1}{2}l_2))$$

ISSN No:-2456-2165 https://doi.org/10.38124/ijisrt/25apr256

$$k_4 = hf(t_i + h, x_i + tk_2 + uk_3, v_i + \frac{1}{2}l_3),\ k_4 = h(v_i + \frac{1}{2}l_3)$$

$$l_4 = hg(t_i + h, x_i + tk_2 + uk_3, v_i + \frac{1}{2}l_3), l_4 = h(-\frac{k}{m}(x_i + tk_2 + uk_3) - \frac{c}{m}(v_i + \frac{1}{2}l_3))$$

So that

$$x_{i+1} = x_i + \frac{1}{6}(k_1 + k_4) + \frac{1}{3}(sk_2 + uk_3)$$

$$v_{i+1} = v_i + \frac{1}{6}(l_1 + l_4) + \frac{1}{3}(sl_2 + ul_3)$$

Enter the r, s, t and your values with

$r = \frac{\sqrt{2}-1}{2}$; $t = -\frac{\sqrt{2}}{2}$; $s = \frac{2-\sqrt{2}}{2}$; $u = 1 + \frac{\sqrt{2}}{2}$

For $i = 0,1,2,\cdots,n-1$, with n is several steps or iterations, then based on the Runge Kutta Gill calculations that have been described, a MATLAB program will be created to obtain a solution from each iteration in the form of a table so that further analysis can be carried out and a graph can be produced that can provide a clearer picture.

The code below implements the Runge-Kutta-Gill method to determine the dynamic equilibrium of a motorcycle in equation (18). The program also compares the numerical results with the exact solution and calculates the error. The following are the steps to draw the graph: (1) Error versus time: The error graph is calculated by comparing the numerical solution with the exact solution; (2) Position versus time: Position $x(t)$ plotted against time t; (3) Speed against time: Speed of $v(t)$ plotted against time t. The processing results are shown in Figure 4.

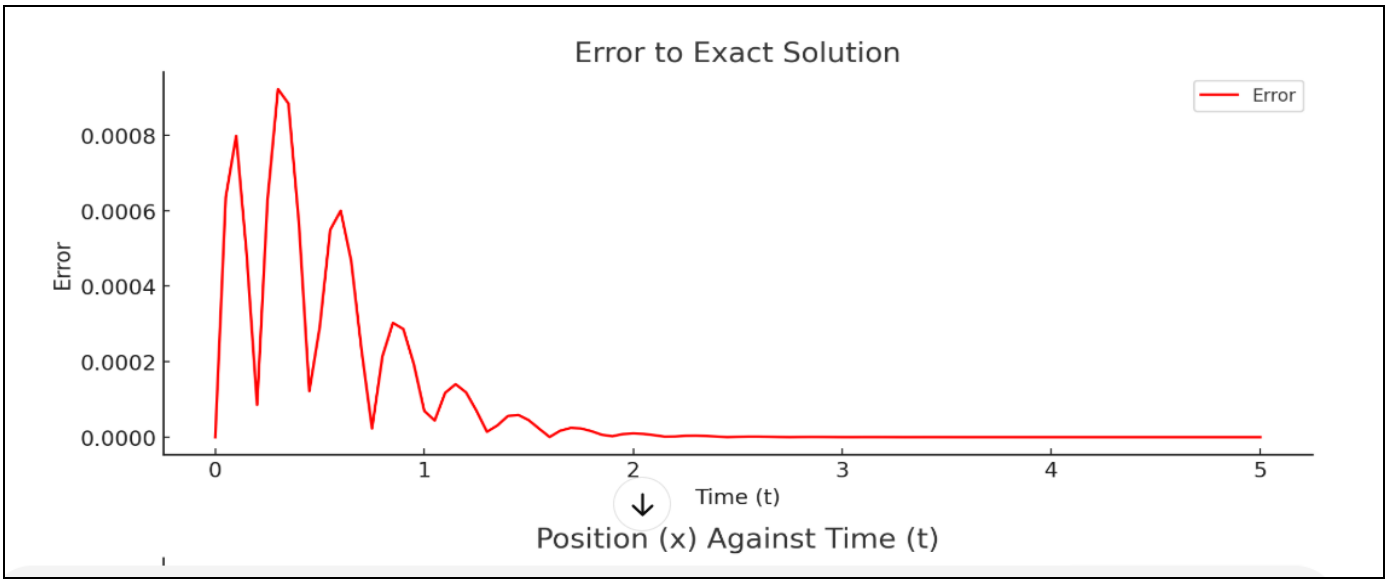

Fig 4: Error Graph against Exact Solution of Runge Kutta Gill Method

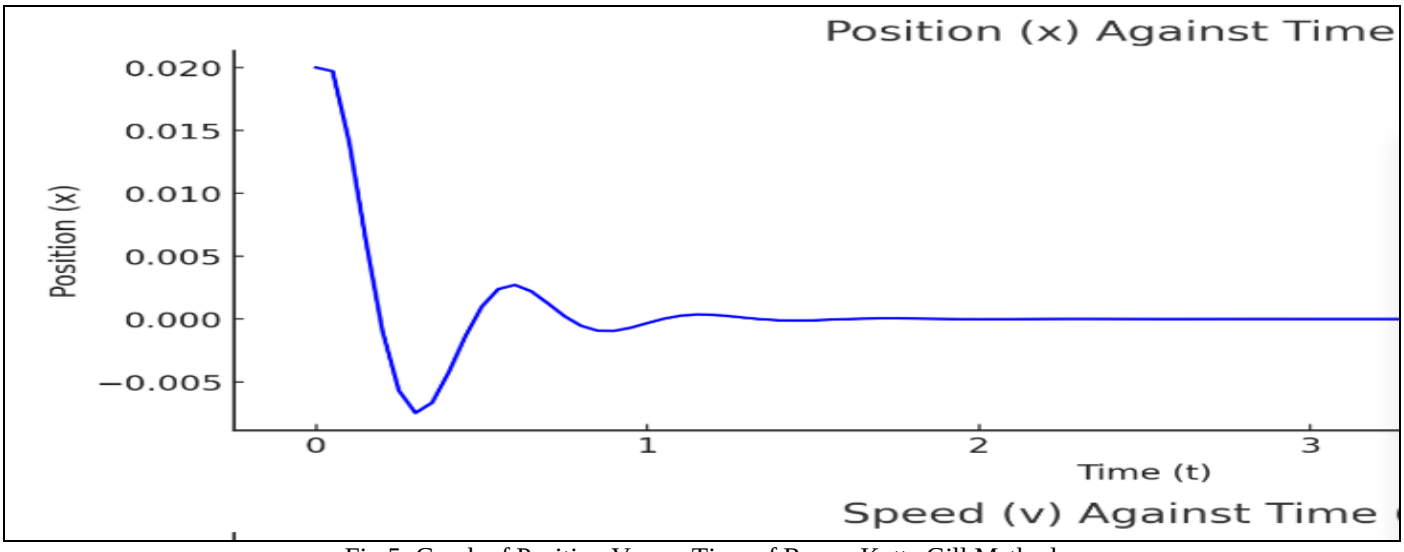

Fig 5: Graph of Position Versus Time of Runge Kutta Gill Method

ISSN No:-2456-2165 https://doi.org/10.38124/ijisrt/25apr256

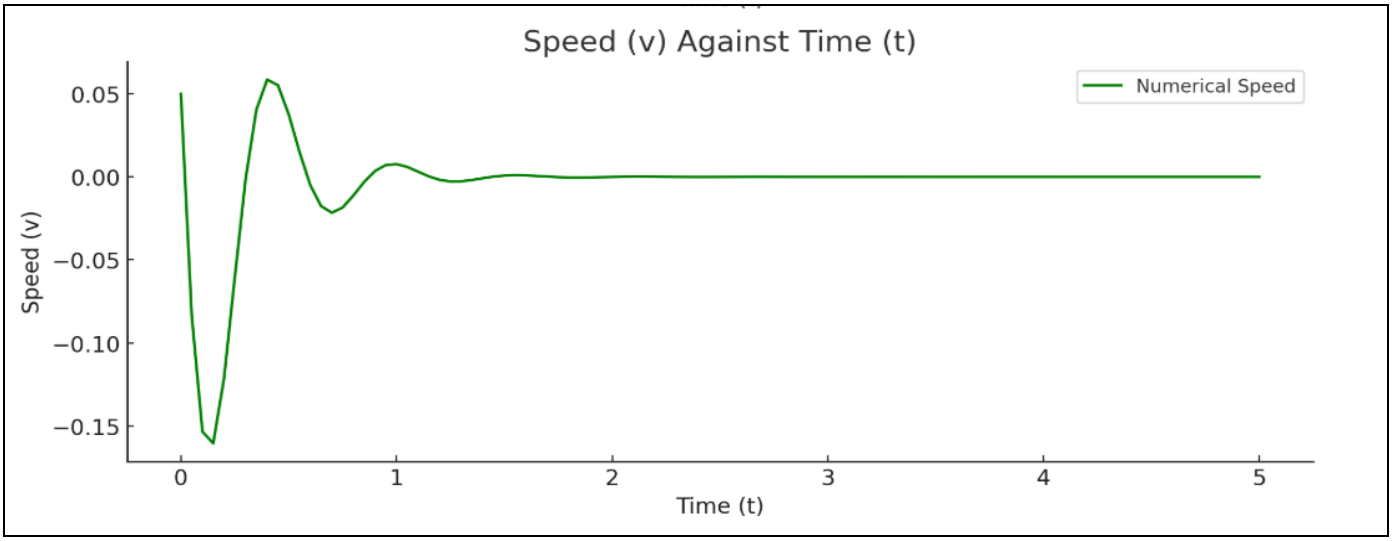

Fig 6: Graph of Speed Versus Time of Runge Kutta Gill Method

➢ *Script MATLAB Metode Runge Kutta Gill*

```
function [t, x, v] = rukg(g, t0, x0, v0, h, n)
% Initialize vectors t, x, dan v
t = zeros(n+1, 1);
x = zeros(n+1, 1);
v = zeros(n+1, 1);

% Set initial condition
t(1) = t0;
x(1) = x0;
v(1) = v0;

% Calculate constants r, s, t, dan u
r = (sqrt(2) - 1) / 2;
s = (2 - sqrt(2)) / 2;
t_koefisien = -sqrt(2) / 2;
u_koefisien = 1 + sqrt(2) / 2;

% Runge-Kutta-Gill Iterations
for i = 1:n
    % Calculates k1 and l1
    k1 = h * v(i);
    l1 = h * g(t(i), x(i), v(i));

    % Calculates k2 and l2
    k2 = h * (v(i) + 0.5 * l1);
    l2 = h * g(t(i) + 0.5 * h, x(i) + 0.5 * k1, v(i) + 0.5 * l1);

    % Calculates k3 and l3
    k3 = h * (v(i) + 0.5 * l2);
    l3 = h * g(t(i) + 0.5 * h, x(i) + r * k1 + s * k2, v(i) + 0.5 * l2);

    % Calculates k4 and l4
    k4 = h * (v(i) + 0.5 * l3);
    l4 = h * g(t(i) + h, x(i) + t_coefficient * k2 + u_coefficient * k3, v(i) + 0.5 * l3);

    % Update x dan v values using the Runge-Kutta-Gill formula
    x(i+1) = x(i) + (1/6) * (k1 + k4) + (1/3) * (s * k2 + u_coefficient * k3);
    v(i+1) = v(i) + (1/6) * (l1 + l4) + (1/3) * (s * l2 + u_coefficient * l3);

    % Update t value
    t(i+1) = t(i) + h;
end
    x_exact = (exp(-3.940828402*t)).*(0.02*cos(10.91428636*t) + 0.0118*sin(10.91428636*t));

% Calculating error
error = abs(x - x_exact)

figure;
plot(t, error);
xlabel('Time(t)');
ylabel('Error');
title('Error to exact solution');
end
```

➢ *Command Window untuk menjalankan script*

```
k = 9102.46;
m = 67.6;
c = 532.8;
g = @(t, x, v) -k/m*x - c/m*v;
t0 = 0;
x0 = 0.02;
v0 = 0.05;
h = 0.05; % steps
n = 100; % number of iteration
[t, x, v] = rukg(g, t0, x0, v0, h, n);
%  Displaying results
disp('Numeric result:')
disp('--------------------------------')
disp('    t        x        v   ')
disp('--------------------------------')
disp([t, x, v])
figure;
subplot(2, 1, 1);
plot(t, x);
xlabel('Time(t)');
ylabel('Position(x)');
```

ISSN No:-2456-2165 https://doi.org/10.38124/ijisrt/25apr256

```
title('Position (x) Against Time (v)');
subplot(2, 1, 2);
plot(t, v);
xlabel('Time(t)');
ylabel('Speed(v)');
title('Speed (v) Against Time (v)');
```

Based on Figure 4, Figure 5, and Figure 4, it can be seen that the spring oscillates and reaches equilibrium at t = 1.65, stretches maximally at a value of 0.02 for position x versus time and stretches maximally at a value of v = 0.0586 for velocity versus time and has the most significant error of 0.9220. This code and graph simulate a damped oscillatory system with an accurate numerical approach. The resulting graph is based on the expectations of the physical model, where damping causes a decrease in the amplitude of oscillations in position and velocity. These results will be compared with the 4th Order Runge Kutta Method. This code uses the 4th-order Runge-Kutta method to solve the system of differential equations in equation (18). This method is a high-accuracy numerical approach, similar to the previous Runge-Kutta-Gill method, but without specific additional constants. The program produces graphs including (1) Error to the exact solution, namely the error graph between the numerical solution and the exact solution; (2) Position x(t) versus time, namely the position of the object versus time using the numerical solution; and (3) Velocity v(t) against time, namely the velocity of the object against time using a numerical solution. Figure 6 is the resulting graph. Figure 5 and Figure 6 are the MATLAB scripts for the 4th Order Runge Kutta Method and the Command Window to run the script.

➢ *Script MATLAB Metode Runge Kutta Orde 4*

```
    function [t, x, v] = rk4(g, t0, x0, v0, h, n)
  % Initializes vectors t, x, dan v
  t = zeros(n+1, 1);
  x = zeros(n+1, 1);
  v = zeros(n+1, 1);

  % Sets the initial condition
  t(1) = t0;
  x(1) = x0;
  v(1) = v0;

  for i = 1:n
    % Calculates k1 and l1
    k1 = h * v(i);
    l1 = h * g(t(i), x(i), v(i));

    % Calculates k2 and l2
    k2 = h * (v(i) + 0.5 * l1);
    l2 = h * g(t(i) + 0.5 * h, x(i) + 0.5 * k1, v(i) + 0.5 * l1);

    % Calculates k3 and l3
    k3 = h * (v(i) + 0.5 * l2);
    l3 = h * g(t(i) + 0.5 * h, x(i) + 0.5 * k2, v(i) + 0.5 * l2);

    % Calculates k4 and l4
    k4 = h * (v(i) + 0.5 * l3);
    l4 = h * g(t(i) + h, x(i) + k3, v(i) + 0.5 * l3);

    % Updates x dan v value
    x(i+1) = x(i) + (k1 + 2*k2 + 2*k3 + k4) / 6;
    v(i+1) = v(i) + (l1 + 2*l2 + 2*l3 + l4) / 6;

    % Updates t value
    t(i+1) = t(i) + h;
  end
  x_exact = (exp(-
3.940828402*t)).*(0.02*cos(10.91428636*t) +
0.0118*sin(10.91428636*t));

% Calculates error
error = abs(x - x_exact)

figure;
plot(t, error);
xlabel('t');
ylabel('Error');
title('Error with exact solution');
end

Command Window to run the script
k = 9102.46;
m = 67.6;
c = 532.8;
g = @(t, x, v) -k/m*x - c/m*v;
t0 = 0;
x0 = 0.02;
v0 = 0.05;
h = 0.05; % steps
n = 100; % iterations number
[t, x, v] = rk4(g, t0, x0, v0, h, n);
% Displaying the result
disp('Numerik result:')
disp('---------------------------------')
disp('     t        x         v    ')
disp('---------------------------------')
disp([t, x, v])
figure;
subplot(2, 1, 1);
plot(t, x);
xlabel('Time(t)');
ylabel('Position(x)');
title('Position (x) Against Time (v)');
subplot(2, 1, 2);
plot(t, v);
xlabel('Time(t)');
ylabel('Speed(v)');
title('speed (v) Against Time (v)');
```

ISSN No:-2456-2165 https://doi.org/10.38124/ijisrt/25apr256

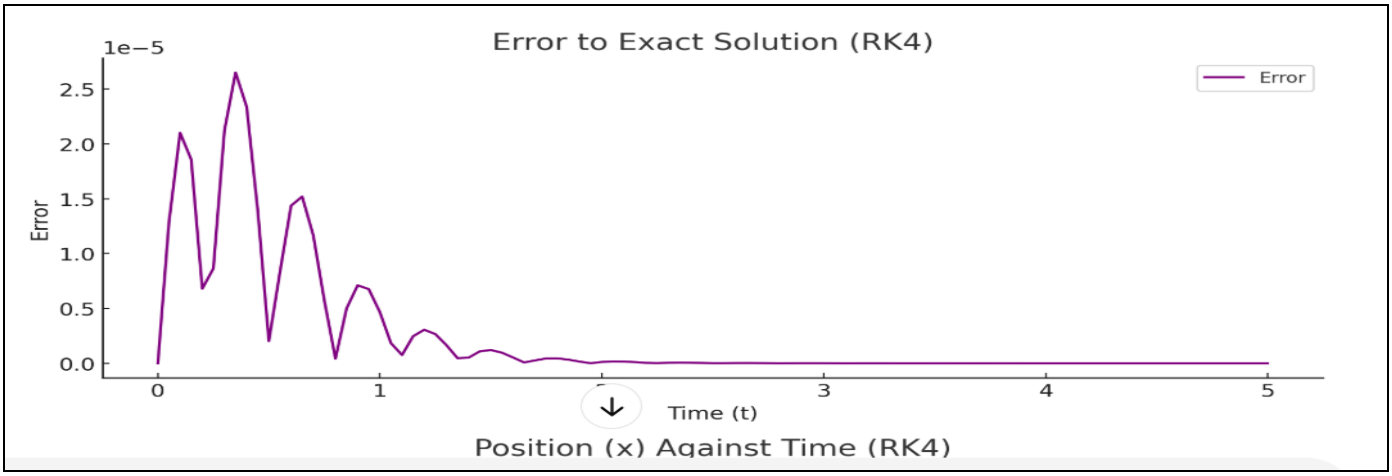

Fig 7: Graph of Error to Exact Solution of RK4

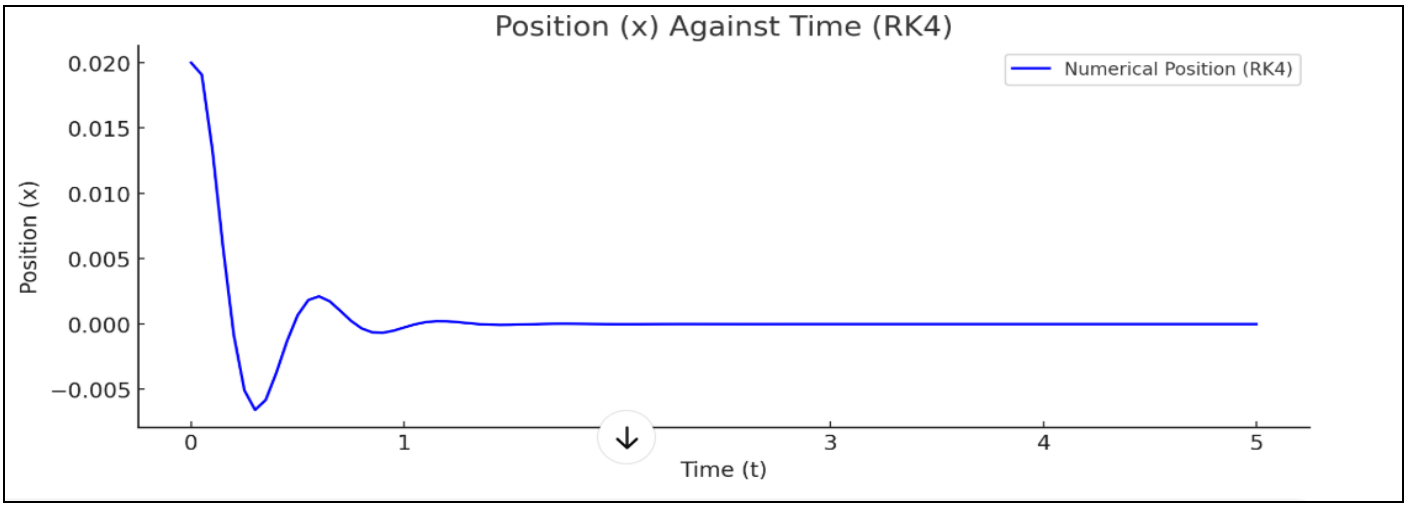

Fig 8: Graph of Position againt Time of RK4

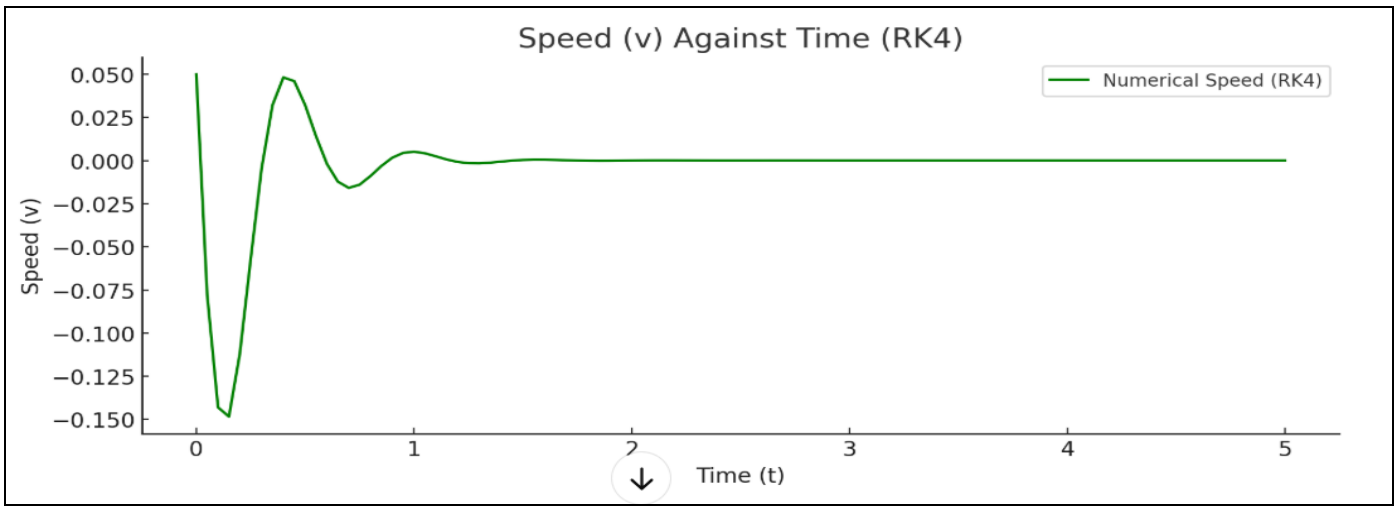

Fig 9: Graph of Speed againt Time of RK4

Based on Figure 7, Figure 8, and Figure 9, it can be seen that the spring oscillates and reaches equilibrium at time $t = 1.65$. It stretches maximally at $x = 0.02$ for position x versus time and at 0.0557 for velocity v versus time and has the most significant error of 0.7011. Compared to the Runge-Kutta-Gill method, RK4 is more accurate for this system because it has more local minor mistakes than RK-Gill. RK4 is more stable for time steps h = 0.05, which allows long-term system analysis without significant error accumulation. In general, RK4 successfully solves the differential equations of the damped oscillatory system with consistent and accurate results. The position and velocity graphs show how the system energy is dissipated due to damping. The error in the exact solution is small, which shows the reliability of the RK4 numerical method for this physical system.

ISSN No:-2456-2165 https://doi.org/10.38124/ijisrt/25apr256

C. Solusi Model Getaran Pegas dengan Metode Milne

The Milne method is a predictor-corrector method that combines several previous steps to calculate the following values. This method is usually used in ordinary differential equations because of its efficiency and accuracy in smooth solutions. The Milne method is a continuation of the Runge Kutta Gill method, namely as a prediction stage for equation (16) as in equation (14)

$$x_{i+1} = x_{i-3} + \frac{4h}{3}(2f_{i-2} - f_{i-1} + 2f_i)$$

$$v_{i+1} = v_{i-3} + \frac{4h}{3}(2g_{i-2} - g_{i-1} + 2g_i)$$

The next stage is to correct equation (16) as in equation (15). So that we obtain

$$x_{i+1} = x_{i-1} + \frac{h}{3}(f_{i-1} + 4f_i + f_{i+1})$$

$$v_{i+1} = v_{i-1} + \frac{h}{3}(g_{i-1} + 4g_i + g_{i+1})$$

Based on the provisions of the Milne method that have been described, a Matlab program will be created from the Milne method, which aims to obtain the results of each iteration in the form of a table so that further analysis can be carried out and produce a graph that can provide a clearer picture. This program will create a graph in the form of (1) Error against the exact solution, (2) Position x(t) against time, and (3) Velocity v(t) against time.

The simulation results on the unique equation (18) obtained three result graphs from the Milne method simulation: (1) Error Graph against Time: This shows the absolute difference between the numerical solution and the exact solution. This graph provides an overview of how accurate the Milne method is in solving ordinary differential equations: (2) Position Graph against Time: Shows the change in system position (x) over time. The position oscillates with damping, as expected for a damped spring system; and (3) Velocity Graph against Time: This shows the change in system velocity (v) over time, which also oscillates with reduced amplitude due to damping. The graphs are presented in Figure 10, Figure 11, and Figure 12. The MATLAB script for the Milne method is as follows.

➢ *Script MATLAB Metode Milne*

```
function [t, x, v] = milneMethod(g, t0, x0, v0, h, n)
% Initializes vectors t, x, dan v
t = zeros(n+1, 1);
x = zeros(n+1, 1);
v = zeros(n+1, 1);

% Set the initial condition
t(1) = t0;
x(1) = x0;
v(1) = v0;

for i = 1:3
    % Calculates k1 and l1
    k1 = h * v(i);
    l1 = h * g(t(i), x(i), v(i));

    % Calculates k2 and l2
    k2 = h * (v(i) + 0.5 * l1);
    l2 = h * g(t(i) + 0.5 * h, x(i) + 0.5 * k1, v(i) + 0.5 *l1);

    % Calculates k3 and l3
    k3 = h * (v(i) + 0.5 * l2);
    l3 = h * g(t(i) + 0.5 * h, x(i) + 0.5 * k2, v(i) + 0.5 *l2);

    % Calculates k4 and l4
    k4 = h * (v(i) + 0.5 * l3);
    l4 = h * g(t(i) + h, x(i) + k3, v(i) + 0.5 * l3);

    % Updates x dan v values
    x(i+1) = x(i) + (k1 + 2*k2 + 2*k3 + k4) / 6;
    v(i+1) = v(i) + (l1 + 2*l2 + 2*l3 + l4) / 6;

    % Updates t value
    t(i+1) = t(i) + h;
end
for i = 4:n
    % Calculates approximate values using the Milne method
    x_pred = x(i-3) + 4*h/3 * (2*v(i-2) - v(i-1) + 2*v(i));
    v_pred = v(i-3) + 4*h/3 * (2*g(t(i-2), x(i-2), v(i-2)) - (g(t(i-1), x(i-1), v(i-1))) + (2*g(t(i), x(i), v(i))));

    % Calculates the correction value using the Milne method
    x_corr = x(i-1) + h/3 * (v(i-1) + 4*v(i) + v_pred);
    v_corr = v(i-1) + h/3 * (g(t(i-1), x(i-1), v(i-1)) + 4*g(t(i), x(i), v(i)) + g(t(i+1), x_pred, v_pred));

    % Updates x and v values
    x(i+1) = x_corr;
    v(i+1) = v_corr;

    % Updates t value
    t(i+1) = t(i) + h;
end
x_exact = (exp(-3.940828402*t)).*(0.02*cos(10.91428636*t) + 0.0118*sin(10.91428636*t));

% Calculating error
error = abs(x - x_exact)

figure;
plot(t, error);
xlabel('Time(t)');
ylabel('Error');
title('Error Towards Exact Solution');
end
```

Here is the Command Window to run the script

```
k = 9102.46;
m = 67.6;
```

ISSN No:-2456-2165 https://doi.org/10.38124/ijisrt/25apr256

```
c = 532.8;
g = @(t, x, v) -k/m*x - c/m*v;
t0 = 0;
x0 = 0.02;
v0 = 0.05;
h = 0.005; % steps
n = 400; % number of iteration
[t, x, v] = milneMethod(g, t0, x0, v0, h, n);
% Displaying results
disp('Numeric result:')
disp('--------------------------------')
disp('     t        x        v    ')
disp('--------------------------------')
disp([t, x, v])
figure;
subplot(2, 1, 1);
plot(t, x);
xlabel('Time(t)');
ylabel('Position(x)');
title('Position (x) Against Time (t)');
subplot(2, 1, 2);
plot(t, v);
xlabel('Time(t)');
ylabel('Speed(v)');
title('speed (v) Against Time (t)');
```

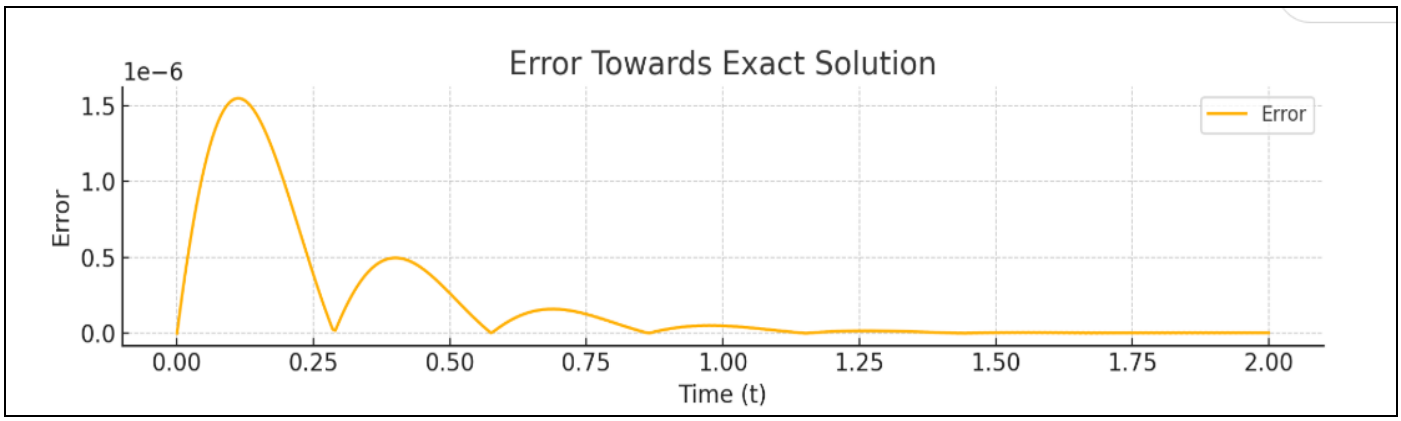

Fig 10: Graph of Error against Exact Solutions of Milne Method

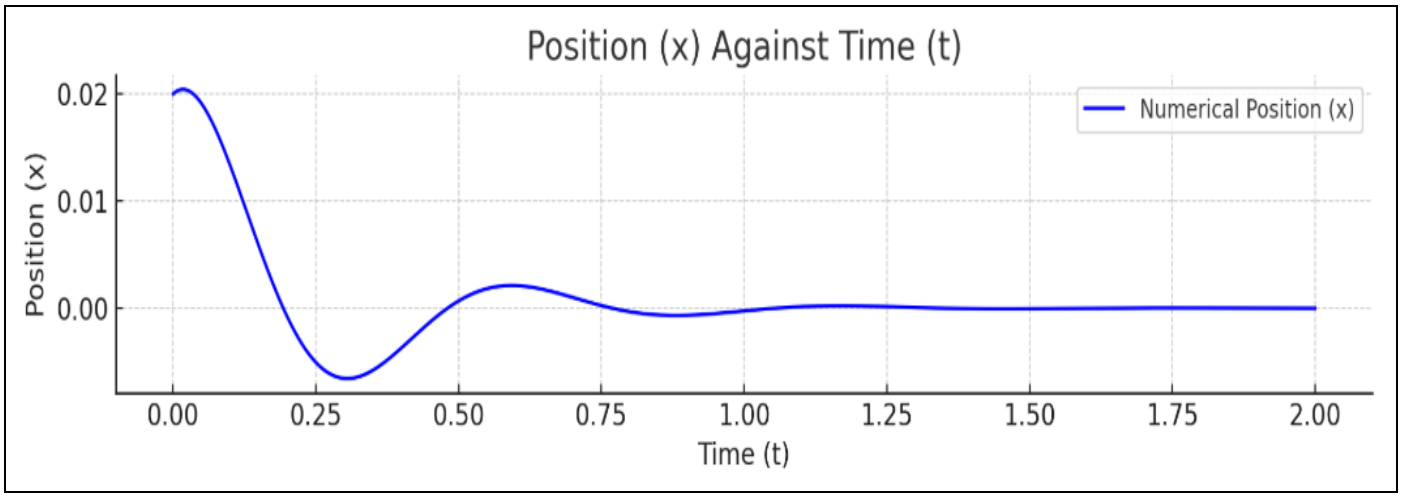

Fig 11: Graph of Position Against Time of Milne Method

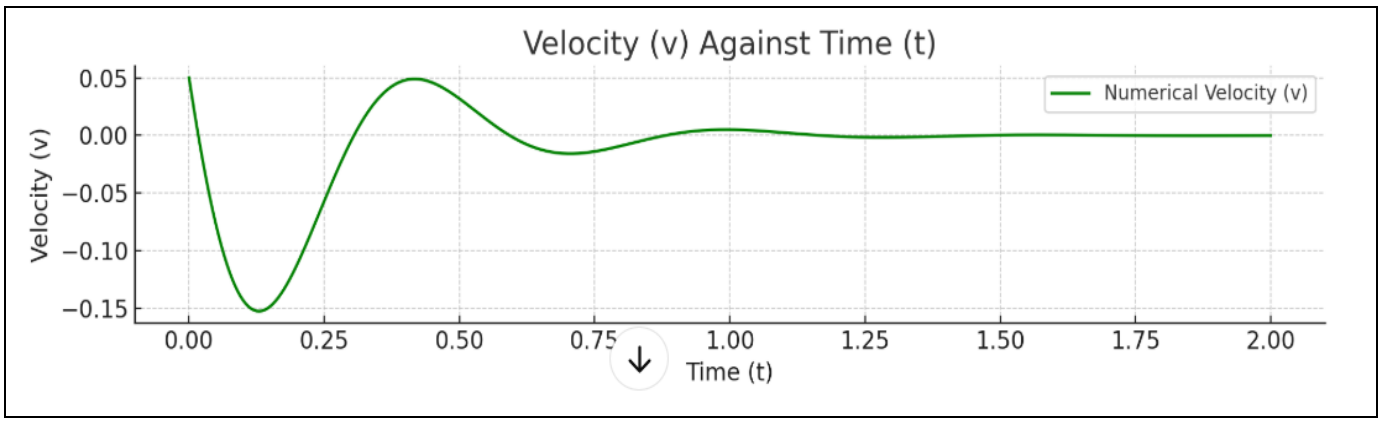

Fig 12: Graph of Velocity Versus Time Milne Method

ISSN No:-2456-2165 https://doi.org/10.38124/ijisrt/25apr256

Based on Figure 10, Figure 11, and Figure 12, it can be seen that the spring oscillates and reaches an equilibrium point at around $t = 2$ and stretches maximally by $x = 0.0204$ for position x versus time and stretches maximally at a value of 0.0500 for velocity versus time with the most significant error of 0.1095. Based on the results above, the data processing results that can be analyzed are (1) Accuracy. The Milne method requires initial values calculated from other methods, such as the RK4. This is a general and practical approach, but it depends on the step h. If h is too large, the initial results may not be accurate enough, causing error accumulation in subsequent iterations; (2) Stability. The Milne method is an explicit method that has a stability limit. For systems with damped oscillations, numerical stability depends on the step h. If h is too large, this method can become unstable; (3) Efficiency. The prediction and correction processes are calculated for each iteration. You can improve efficiency by checking the convergence of predictions and corrections (e.g., stopping the correction iteration if the relative changes are small enough); (4) Error Calculation. The error graph is quite informative, but it presents the relative errors $error\ relative = \frac{[x - x_{exact}]}{x_{exact}}$.

It may be more relevant to evaluate the solution in cases where the scale of the xx positions changes significantly; (5) Modularity. The code can be more modular by separating key functions, such as Runge-Kutta and Milne calculations, into separate functions to improve readability and maintainability.

IV. CONCLUSION

Based on the three methods above, it can be concluded that (1) Accuracy and stability. The Runge-Kutta of Order 4 (RK4): (a) It shows higher accuracy and minor errors and is more stable than Milne. (b) It suits complex dynamics or rapid change systems more suitably. Runge-Kutta-Gill: (a) Slightly more accurate than Milne but less accurate than RK4, and (b) Milne can be an efficient choice for significant time steps with moderate accuracy requirements. (2) Computational Efficiency. Milne's method is more efficient regarding the number of function evaluations per step than RK4. However, it requires smaller time steps to achieve equivalent accuracy, which can offset this efficiency. RK4 requires four function evaluations per step, making it more computationally intensive. However, its higher accuracy can reduce the need for small time steps. (3) Implementation. Milne's method requires an initial step using another method (e.g., RK4) to start the Milne iteration, and Implementation can be more complex due to the nature of the predictor-corrector. The RK4 method shows a more direct implementation and does not require additional initial steps, making it generally more straightforward to use. Milne can be a good choice for simple damped oscillatory systems requiring computational efficiency. For applications that require high accuracy and better stability, especially in long-term simulations, the RK4 method is more recommended. Suggestions for improvement are: (1) Reducing the time step (h) Reducing hh can improve the accuracy of the Milne method, but with an increase in the number of iterations and computational time. (2) Use of Hybrid Methods: Combining Milne with other predictor-corrector methods or dynamic time step adaptation to improve stability and accuracy. (3) To ensure consistency and accuracy, further validation is needed to compare the Milne results with other numerical methods, such as the Adams-Bashforth method.

REFERENCES

[1]. D. Giancoli, R. Resnick dan J. Walker, Fisika Dasar, Jakarta: Erlangga, 2005.
[2]. J. Clerk Maxwell, A Treatise on Electricity and Magnetism, 3rd ed., vol. 2. Oxford: Clarendon, 1892, pp.68-73.
[3]. I.S. Jacobs and C.P. Bean, “Fine particles, thin films and exchange anisotropy,” in Magnetism, vol. III, G.T. Rado and H. Suhl, Eds. New York: Academic, 1963, pp. 271-350.
[4]. K. Elissa, “Title of paper if known,” unpublished.
[5]. R. Nicole, “Title of paper with only first word capitalized,” J. Name Stand. Abbrev., in press.
[6]. Y. Yorozu, M. Hirano, K. Oka, and Y. Tagawa, “Electron spectroscopy studies on magneto-optical media and plastic substrate interface,” IEEE Trans. J. Magn. Japan, vol. 2, pp. 740-741, August 1987 [Digests 9th Annual Conf. Magnetics Japan, p. 301, 1982].
[7]. M. Young, The Technical Writer’s Handbook. Mill Valley, CA: University Science, 1989.
[8]. Kuang, J., Wang, M., Han, J., & Sheng, Y. (2023). Improved Milne-Hamming Method for Resolving High-Order Uncertain Differential Equations. *Applied Mathematics and Computation*, *457*, 128199.
[9]. Ixaru, L. G., & Rizea, M. (1980). A Numerov-like scheme for the numerical solution of the Schrödinger equation in the deep continuum spectrum of energies. *Computer Physics Communications*, *19*(1), 23-27.
[10]. Van de Vyver, H. (2005). Frequency evaluation for exponentially fitted Runge–Kutta methods. *Journal of computational and applied mathematics*, *184*(2), 442-463.
[11]. Steyer, A. J., & Van Vleck, E. S. (2016). A step-size selection strategy for explicit Runge–Kutta methods based on Lyapunov exponent theory. *Journal of computational and applied mathematics*, *292*, 703-719.