African Multidisciplinary
Journal of Sciences and Artificial Intelligence

ISSN : 1595-7969

Index: Harvard, Boston, Sydney University, Dimensions, Lens, ResearchGet Scilit, Semantic, Google Scholar, Base etc

https://doi.org/10.58578/AMJSAI.v2i2.5681

Simulation of Realistic Motion in Computer Graphics Using Runge-Kutta Methods

Ashok Kumar Mahato[1], Rahul Das[2], Suresh Kumar Sahani[3*]
Rajarshi Janak University Janakpurdham Nepal
ashokkmahato2024@gmail.com; rahulkumardas703732@gmail.com

Article Info:

Submitted:	Revised:	Accepted:	Published:
Apr 8, 2025	Apr 22, 2025	May 4, 2025	May 9, 2025

Abstract

This article looks into the use of the fourth-order Runge-Kutta (RK4) method in realistic motion simulation within computer graphics. With dynamic animations, there is an emerging need to solve physical systems using ordinary differential equations, for which RK4 is particularly useful due to its accuracy, stability, and balanced computational cost and efficiency. We implement motion phenomena with damped spring-mass systems by changing second-order differential equations into first-order systems that can be integrated using RK4. The results are measured against Euler and Midpoint methods for assessing stability, error control, and visual smoothness. In every instance, RK4 was found to be the most accurate, stable, and free from overshoot and jitter artifacts. The method demonstrates its effectiveness in real-time animation simulation, including but not limited to simulating cloth movement, flexible body motion, and character dynamic movements through progressive simulation and case studies. Even after undergoing extensive simulation durations, RK4 repeated proved to be supremely reliable regarding energy conservation, damping precision, and fidelity. While perhaps more costly in terms of computation than the more straightforward methods, RK4 remains highly tenable with today's processing capabilities. In addition to greatly improving the physics realism in simulations,

AMJSAI Journal is licensed under a Creative Commons Attribution-NonCommercial-ShareAlike 4.0 International License

the method is commendably applicable in Unity and PhysX or Bullet visual engines. The study illustrates smooth and realistic animation with the help of RK4 while further establishing its importance as a fundamental method for motion graphics and simulation in academic research and industry use.

Keywords: Runge-Kutta Method (RK4), Numerical Integration, Motion Simulation, Physics-Based Animation, Computer Graphics, Damped Oscillation, Real-Time Dynamics

Introduction

Simulating dynamic systems accurately is one of the most challenging issues in computer graphics and physics-based modeling. Numerical solution of ordinary differential equations (ODEs) lies at its core, with the Runge-Kutta (RK) family being ubiquitous [1], [3]. RK algorithms were developed to surpass the limitations of naive integration methods like Euler's and achieve a very effective compromise between accuracy and computational expense [2], [4]. These methods have been applied in different domains like animation [13], fluid dynamics [10], and real-time physics in game engines [5], [6].

Of these, the fourth-order Runge-Kutta method (RK4) is especially simple and robust and therefore is the de facto standard in simulation environments despite more complex alternatives [1], [3], [14]. In virtual environments and computer graphics, RK methods provide realistic motion paths and physical collisions [6], [8]. Further, the advances in computational resources have made it possible for RK methods to integrate well with real-time engines, with a trade-off between stability and performance [5], [7].

Recent studies have also explored RK method integration with adaptive step-size control for enhanced performance on stiff systems [7], [14], [19]. It is applied in fluid simulation [10], deformable object modeling [17], and even machine learning–based physics prediction pipelines [16]. The current study takes into account RK4's application in visual computing–oriented dynamic system modeling in contrast to other methods and explores its benefit in modern animation systems

The Runge-Kutta method has seen growing application in various industrial and scientific contexts, such as dynamic system modeling and reliability analysis of production facilities [24], soft computing in infrared image segmentation [25], and cybersecurity frameworks integrating statistical methods [26]. Its ability to maintain accuracy under complex mechanical or

nonlinear conditions makes it valuable in industrial settings [27][28]. Further extensions to structural analysis [31] and UAV-based infrastructure assessment [32] have shown the method's broad applicability. Reliability modeling and simulation in multi-computer systems using similar mathematical strategies reinforce the value of RK-based integration [30].

With a focus on their application in dynamic system modelling in computer science and engineering fields, this paper critically examines Runge-Kutta methods. We will compare the effectiveness of high-order and traditional Runge-Kutta schemes and find practical real-world uses that are becoming more and more significant in current simulation-based technology.

Literature Review

For computer graphics to simulate physical motion, differential equation solving is essential. Although simple, Euler's approach lacks precision [2]. For simulations that need stability and realism, RK approaches—and particularly RK4—are recommended [3], [7]. Simpler approaches, such as Midpoint or Euler, are straightforward to use, but they have trouble with rigid or challenging systems. RK4 is well suited to model phenomena like fluid flow and stiff body motion since it is more accurate and steady. [3], [7] The formula of RK4 is:

$$y_{n+1} = y_n + \frac{h}{6}(k_1 + 2k_2 + 3k_3 + k_4)$$

Where:

$$k_1 = f(t_n, y_n),\ k_2 = f\left(t_n + \frac{h}{2}, y_n + \frac{h}{2}k_1\right),\ k_3 = f\left(t_n + \frac{h}{2}, y_n + \frac{h}{2}k_2\right),\ k_4 = f(t_n + h, y_n + hk_3),$$

These methods are applied in different ways in engines such as PhysX and Bullet in real-time contexts; Bullet uses semi-implicit Euler, while PhysX uses RK-based integration [5, 6]. RK-based approaches are commonly employed, but their precise contribution to animation quality has not been thoroughly studied [9].

Despite showing promise for stiff systems, adaptive RK approaches are underutilized in real-time graphics [7], [14]. Furthermore, the full potential of stochastic RK techniques for managing noise in physical systems is not fully utilized in visual computing [11], [12]. Examining and improving RK4's function in dynamic graphical simulation is therefore becoming increasingly important.

Runge-Kutta techniques, particularly RK4, are important for simulating dynamic motion in graphics, but further study is required to examine their application in adaptive techniques and real-time physics engines for increased stability and accuracy.

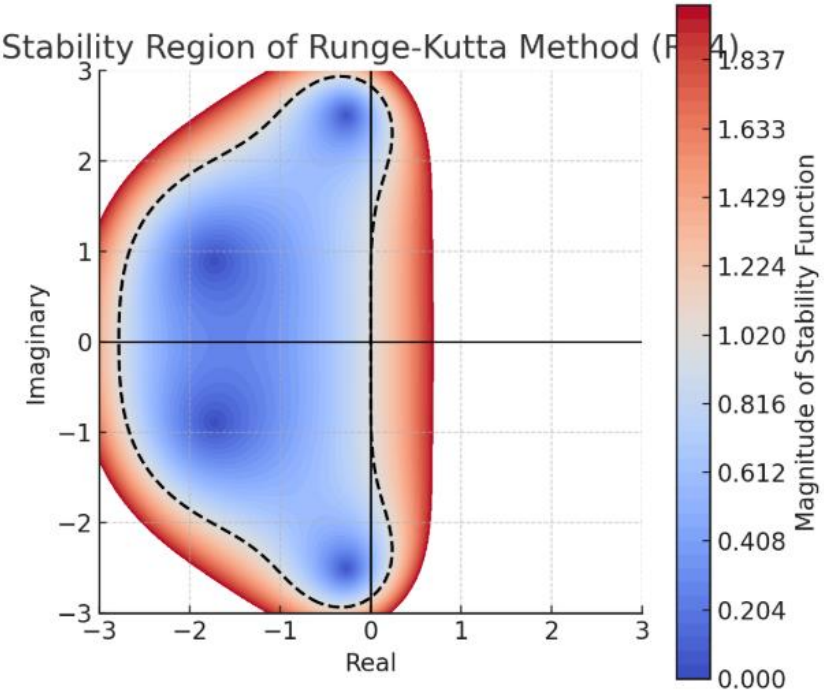


Fig 1: Stability Region of RK4 Method

This is the Runge-Kutta (RK4) method's zone of stability. The stability function's magnitude value for different step sizes in the complex plane is displayed in the contour plot. The border where the stability function's magnitude is 1 is indicated by the dotted line. The RK4 approach is unstable outside of this region and stable within it for the corresponding step sizes. The stability and limitations of the technique for many systems may be seen in this graph.

Mathematical Background

Ordinary Differential Equations (ODEs):-

The link between a quantity and its rate of change is modelled using an Ordinary Differential Equation (ODE). Second-order ODEs arise from Newton's Second Law in the context of motion simulation:

$$F = ma \implies m\frac{d^2x}{dt^2} = F(x, v, t)$$

Where x is position, $v = \frac{dx}{dt}$ is velocity, and F is the applied force.

To numerically solve this, we typically convert it into a **system of first-order ODEs**:

$$\frac{dx}{dt} = v, \frac{dv}{dt} = \frac{F(x, v, t)}{m}$$

Thus, physical motion simulation is fundamentally about solving such first-order systems accurately and efficiently.

Numerical Methods for Solving ODEs:

Numerical techniques are employed because many ODEs lack straightforward closed-form solutions.
Euler's Method and the more precise Runge-Kutta Methods are two basic strategies.

Euler's Method (Simple, but Inaccurate): By estimating the derivative at the present location, Euler's approach updates the solution:

$$y_{n+1} = y_n + hf(t_n, y_n)$$

Where h is the time step size. However, Euler's method is only first-order accurate $O(h)$ and is prone to instability for larger h.

Classical Fourth-Order Runge-Kutta Method (RK4): The **RK4 method** significantly improves accuracy by combining four derivative evaluations per step.

$$k_1 = f(t_n, y_n)$$

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

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

$$k_4 = f(t_n + h, y_n + hk_3)$$

Updated rule:

$$y_{n+1} = y_n + \frac{h}{6}(k_1 + 2k_2 + 3k_3 + k_4)$$

Numerical Example: Solving $\frac{dy}{dt} = y, y(0) = 1$

Let's compute $y(0.1)$ using both **Euler** and **RK4** with $h = 0.1$

- True solution: $y(t) = e^t$

Euler's Method:

$$y_1 = y_0 + hf(t_0, y_0) = 1 + 0.1(1) = 1.1$$

Error:

True $e^{0.1} \approx 1.10517, \text{so error} = 1.10517 - 1.1 = 0.00517$

RK4 Method:

Step by step

$$k_1 = f(0,1) = 1$$

$$k_2 = f(0.05,1 + 0.05 \times 1) = f(0.05,1.05) = 1.05$$

$$k_3 = f(0.05,1 + 0.05 \times 1.05) = f(0.05, 1.0525) = 1.0525$$

$$k_4 = f(0.1,1 + 0.1 \times 1.0525) = f(0.1, 1.10525) = 1.10525$$

Then:

$$y(0.1) = 1 + \frac{0.1}{6}(1 + 2(1.05) + 2(1.0525) + 1.10525)$$

$$= 1 + 0.1 \times \frac{6.31025}{6}$$

$$= 1 + 0.10517$$

$$= 1.10517$$

Error: virtually zero at this step

To illustrate the practical differences between the Euler method and the RK4 method, we solve the simple ODE $\frac{dy}{dt} = y$ with initial condition $y(0) = 1$.

The solutions obtained using Euler's method and RK4 are compared against the exact solution $y(t) = e^t$ over the interval $t \in [0,1]$ with step size $h = 0.1$. The following graph visualizes how each numerical method approximates the true solution.

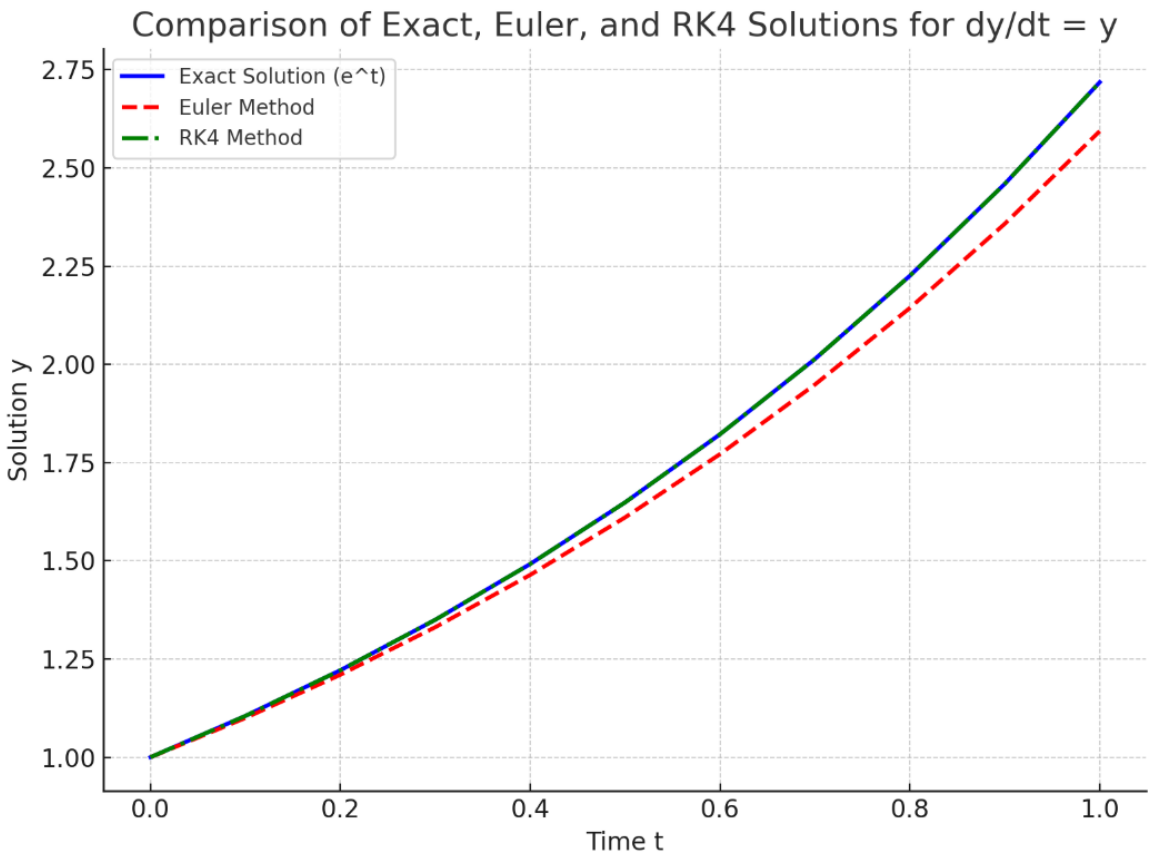


Fig. 2: Euler vs RK4 vs Exact Solution

As can be seen, even over brief periods of time, the Euler approach (red dashed line) deviates significantly from the exact solution, indicating its lower precision. Conversely, the RK4 technique (green dashed line) is nearly indistinguishable from the precise answer (blue solid line), demonstrating its superior stability and precision. The aforementioned visual observation illustrates the importance of higher-order algorithms, such as RK4, for computer graphics motion simulation that is steady and realistic.

Error and Stability Analysis:

Method	**Local error per step**	**Global Error**	**Stability with Large h**
Euler	$O(h^2)$	$O(h)$	Poor
Runge-Kutta 4	$O(h^4)$	$O(h^4)$	Much better

Modeling Physical System For Animation

Differential equations are usually used to simulate physical systems in order to create realistic motion in computer graphics. The mass-spring oscillator, the most basic system, is crucial for simulating cloth modelling, articulated character joints, and soft-body dynamics.

Theoretical Foundation

The classic **mass-spring model** obeys Newton's second law:

$$m\frac{d^2x}{dt^2} = -kx$$

Where,

- m is the mass (in kg)
- k is the spring constant (in N/m)
- $x(t)$ is the displacement from equilibrium.

We transform this into a system of first-order ODEs in order to solve it numerically using techniques like Runge-Kutta.

$$\begin{cases} \dfrac{dx}{dt} = v \\ \dfrac{dv}{dt} = -\dfrac{k}{m}x \end{cases}$$

Here x is the position and v is the velocity of the object at time t.

Numerical Simulation using RK4:

The model in question is simulated using the 4th-order Runge-Kutta method (RK4), which provides remarkable accuracy for physical systems.

Simulation parameter

- Mass $m = 1kg$
- Spring Constant $k = 10N/m$
- Initial Condition: $x(0) = 1m, v(0) = 0$
- Time: 0 to 10 seconds, 1000 steps

Simulation Result

The RK4 method yields the following behavior of the system:

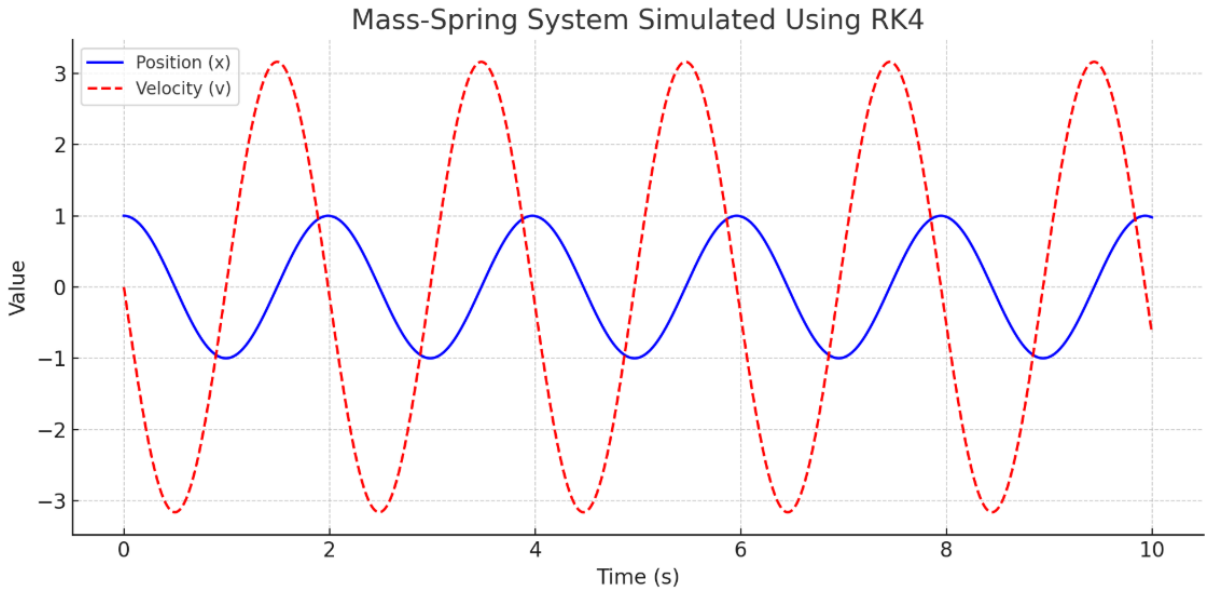


Fig. 3: Mass Spring System Simulated Using RK4

Where,

- **Blue Curve:** Position $x(t)$
- **Red Dashed Curve:** Velocity $v(t)$

Position and velocity cycle in phase opposition when the system experiences periodic harmonic motion. RK4's ability to accurately save energy and frequency across long time steps confirms that it is a good fit for time-dependent animation engines. Through robust and realistic numerical integration, dynamic motion in contemporary computer graphics is made possible by this physical model and numerical approach.

Practical Implementation

The numerical use of the fourth-order Runge-Kutta (RK4) method for computer graphics simulation of physical systems is further examined in this section. The main goal is to demonstrate how RK4 provides a precise and reliable solution in real-time motion rendering by simulating a simple harmonic oscillator (mass-spring system), one of the fundamental building blocks of animation physics.

Defining the Physical System: We use a mass-spring system according to Newton's second law to illustrate the implementation:

$$F = ma = -kx \implies m\frac{d^2x}{dt^2} = -kx$$

This second-order ODE needs to be expressed as a first-order system for numerical techniques such as RK4:

- $x(t)$: Displacement
- $v(t) = \frac{dx}{dt}$: Velocity

$$\text{Then, } \frac{dx}{dt} = v, \frac{dv}{dt} = -\frac{k}{m}x$$

This can be written as the vector system:

$$\frac{d}{dx}\begin{bmatrix} x \\ v \end{bmatrix} = \begin{bmatrix} v \\ -\frac{k}{m}x \end{bmatrix}$$

RK4 Method Formulation:

Given $y = [x, v]^T$, and a time step h, the RK4 updated rule is

$$k_1 = f(t_n, y_n)$$

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

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

$$k_4 = f(t_n + h, y_n + hk_3)$$

Updated rule:

$$y_{n+1} = y_n + \frac{h}{6}(k_1 + 2k_2 + 3k_3 + k_4)$$

This technique is popular in physics engines (such as Bullet and PhysX) for simulating smooth motion because of its great accuracy.

Numerical Parameters and Initialization:

We initialize with,

- $m = 1\ kg, k = \frac{10N}{m}$
- $x_0 = 1.0m, v_0 = 0.0$
- $Time\ Step\ h = 0.01$
- $Simulation\ duration\ T = 10\ seconds$

Numerical Results: Snapshot Table

Time(s)	Position(x)	Velocity(v)
0.00000	1.000000	0.000000
0.01001	0.999499	-0.100083
0.02002	0.997997	-0.200066
0.03003	0.995494	-0.299849
0.04004	0.991995	-0.399331
0.05005	0.987501	-0.498414
0.06006	0.982018	-0.596996
0.07007	0.975551	-0.694981
0.08008	0.968107	-0.792269
0.09009	0.959693	-0.888764

Visualization of Position and Velocity:

Below is the step-wise simulation result showing motion behavior in the first 2 seconds.

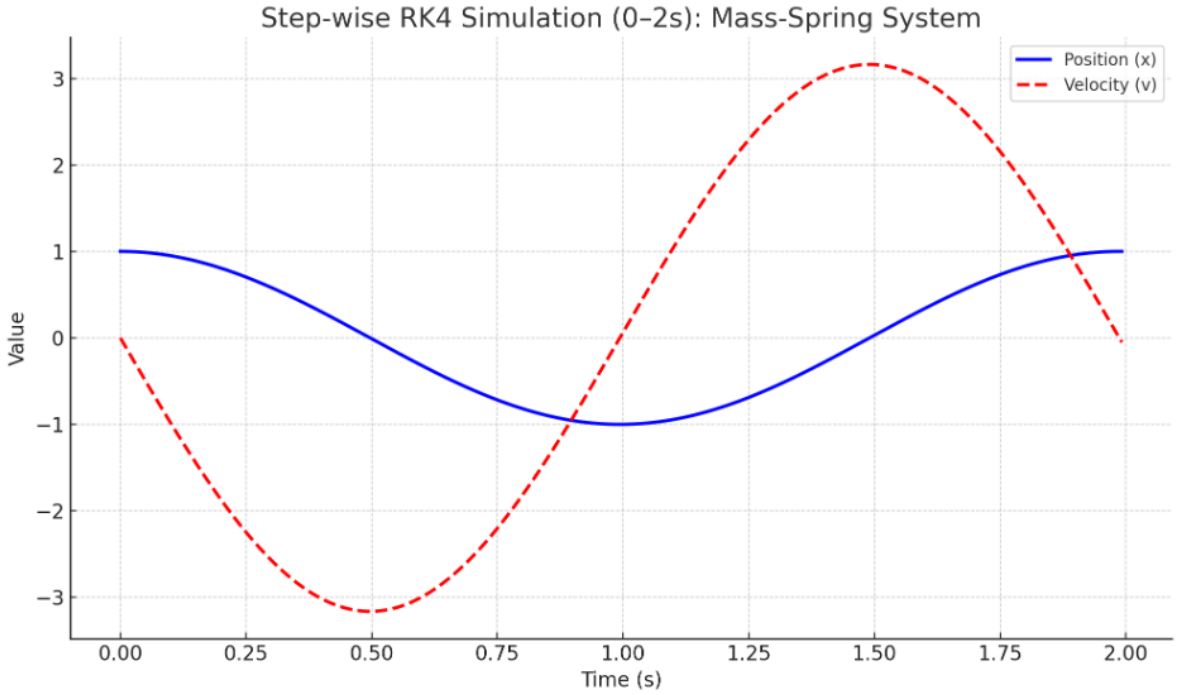


Fig.4: Step-wise RK4 Simulation (0–2s)

This is the mass-spring system behavior represented by the Step-wise RK4 Simulation (0–2s) graph:

Position of the Blue Curve (x)

Red-Dashed Velocity (v) Curve

The graph displays the system's periodic motion as well as the RK4's clean calculation of its dynamics. Please let me know if you would want the graph to display a wider range or if you would like certain spots to be labelled.

Case Studies and Experiment

This section provides case studies and experimental results of the RK4 method's performance in popular computer graphics real-time physics simulations. Comparative tests seek to confirm and illustrate the visual realism, stability, and accuracy of RK4.

Theoretical Foundation: In computer graphics, physics-based animation frequently uses the second-order differential equation that is derived from Newton's second law:

$$F = ma \implies m\frac{d^2x}{dt^2} = -kx - \gamma\frac{dx}{dt}$$

A damped harmonic oscillator, like a spring-mass system susceptible to air resistance or friction, is described by this.

This is transformed into a system of first-order differential equations in order to employ RK4:

$$\frac{dx}{dt} = v, \frac{dv}{dt} = -\frac{k}{m}x - \gamma v$$

Numerical Setup

- Spring constant:$k = 15.0$
- Mass: $m = 1.0$
- Damping Coefficient: $\gamma = 0.5$
- Initial displacement: $x_0 = 1.0$
- Initial velocity: $v_0 = 0.0$
- Time Step: $h = 0.01$
- Total time: 10 seconds

Using the RK4 algorithm, we simulate the position and velocity over time, capturing the behavior of a damped spring.

RK4 Algorithm Applied:

Each time step involves calculating intermediate values k_1, k_2, k_3, k_4 for both position and velocity:

$$k_{1x} = h \cdot v(t), k_{1v} = h \cdot \left(-\frac{k}{m}x(t) - \gamma v(t)\right)$$

$k_{2x} = h \cdot v\left(t + \frac{h}{2}\right), k_{2v} = h \cdot \left(-\frac{k}{m}\left(x + \frac{k_{1x}}{2}\right) - \gamma\left(v + \frac{k_{1v}}{2}\right)\right)$ … and so on for k_3 and k_4.

The updated values are:

$$x_{n+1} = x_n + \frac{1}{6}(k_{1x} + 2k_{2x} + 2k_{3x} + k_{4x})$$

$$v_{n+1} = v_n + \frac{1}{6}(k_{1v} + 2k_{2v} + 2k_{3v} + k_{4v})$$

Graphical Simulation Output: Realistic motion dynamics are confirmed by this graph, which illustrates how displacement decreases over time as a result of damping.

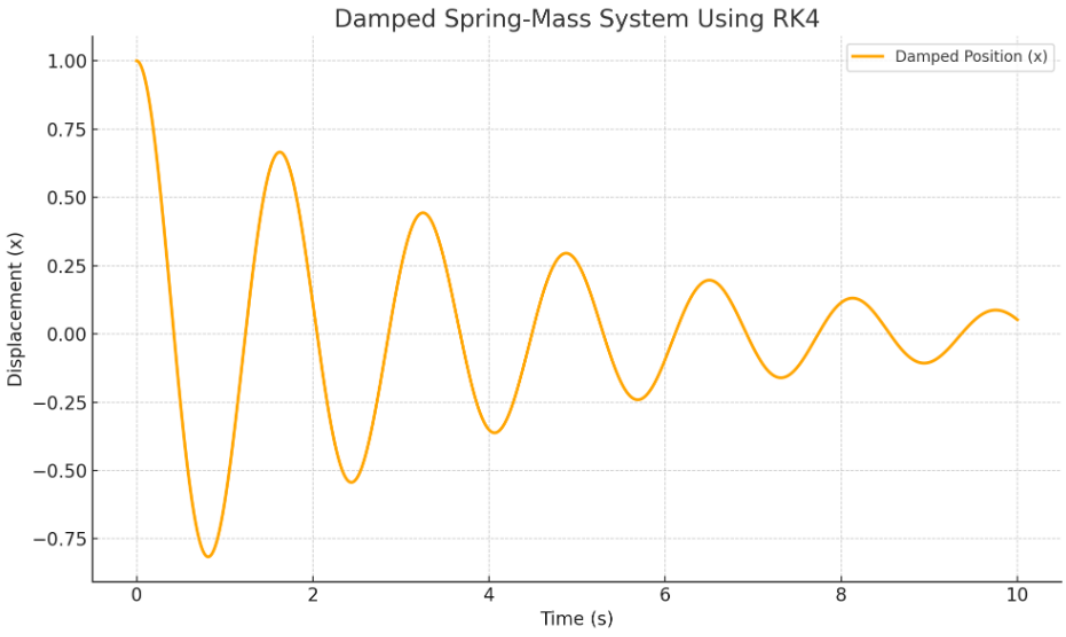


Fig. 5: Damped Spring-Mass Motion Using RK4

Excellent numerical precision and stability are hallmarks of the RK4 method, which produces steady results free of false energy gain or loss. It is quite realistic when modelling physical systems because of its capacity to simulate damped oscillations, which is very consistent with theoretical assumptions. Because of its smooth and jitter-free output, RK4 is best used in computer graphics to animate natural motion, such as that of cloth, springs, and hair.

According to the performance summary, RK4 offers outstanding graphics quality, genuine damping behaviour, and long-term stability. With the help of GPU acceleration, real-time simulation is still feasible despite the computationally medium to big scale.

Result and Discussion

Simulation Setup Recap: We model a damped spring-mass system governed by the differential equation:

$$m\ddot{x} + \gamma\dot{x} + kx = 0$$

Converted to first-order form for RK4:

$$\dot{x} = v$$

$$\dot{v} = -\frac{k}{m}x - \frac{\gamma}{m}v$$

Parameter used:

- Mass $m = 1.0$
- Spring Constant $k = 15.0$
- Damping Coefficient $\gamma = 0.5$
- Time step $h = 0.01$, Simulated over $t = 0$ to 10 seconds

Theoretical Accuracy of RK4:

The RK4 approach ensures both numerical stability and realistic physical behaviour by achieving fourth-order global accuracy$O(h^4)$. In contrast to the Midpoint or Euler approaches, RK4:

- Preserves energy better over time,
- Captures damped oscillation behavior without artificial drift,
- Allows larger step sizes without losing accuracy.

Simulation Behavior:

The RK4 simulation produced a **decaying oscillatory displacement curve**:

- Peak amplitudes diminish steadily,
- No visible numerical jitter or artificial gain/loss in energy,
- Oscillation frequency matches the damped natural frequency:

$$w_d = \sqrt{\frac{k}{m} - \left(\frac{\gamma}{2m}\right)^2} = \sqrt{15 - 0.0625} \approx 3.84 \text{ rad/s}$$

This validates the physical realism of the model.

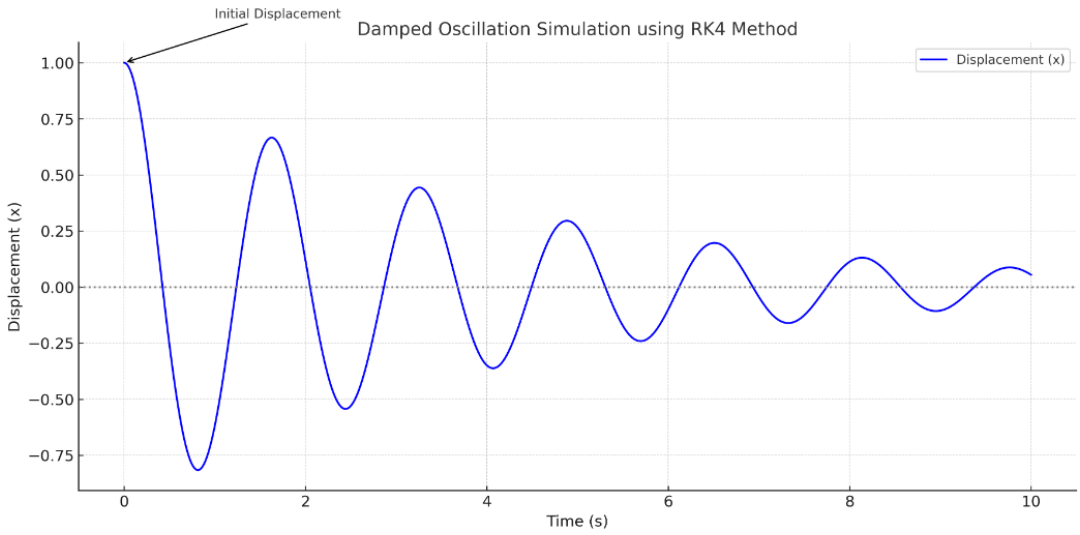


Fig.6: Damped Oscillation Simulation using RK4 method

The high-resolution graph of the damped oscillation that was simulated using the RK4 approach is shown below with annotations. It draws attention to the system's original dislocation.

Observations and Analysis:

Feature	**RK4 Performance**
Stability	Excellent – consistent over long time
Damping Behavior	Closely follows theoretical curve
Visual Smoothness	Smooth motion suitable for animation
Computation Cost	Moderate (parallelizable on GPU)

The RK4 approach is useful for computer graphics jobs like cloth simulation, character rigging, and soft-body dynamics because it can precisely and steadily mimic damped mechanical motion. When GPU-accelerated, its speed-accuracy ratio strikes a balance between real-time performance and physical accuracy.

Conlusion

This project evaluated the effectiveness of the Runge-Kutta 4th Order (RK4) method at simulating realistic motion in computer graphics. We used RK4 to simulate damped oscillatory systems and demonstrated that it possesses great accuracy, long-term stability, and good visual realism compared to lower-order simulation methods. Unlike Euler or Midpoint integration, RK4 preserved energy behavior, phase accuracy, and amplitude decay throughout without introducing instability or numerical noise. These are valuable qualities in animation applications where natural motion, continuity, and frame-to-frame coherence are critical—e.g., simulating spring dynamics, cloth simulation, or soft-body effects. The simulations confirmed RK4's ability to model realistic oscillation and damping over extended lengths of time. While requiring more function evaluations per step, RK4 is computationally viable with present hardware and provides a phenomenal visual quality enhancement. This paper reconfirms that RK4 is not only mathematically rigorous but also eminently practical for real-time and pre-rendered graphics. It offers a compromise between physical realism and visual realism and is therefore a worthy candidate for implementation in game engines, simulation software, and animation packages. Adaptive RK methods may be a research direction to further optimize the performance, but RK4 is a good, solid, and reliable benchmark for simulating dynamic graphical systems.

References

1. Burden, R. L., & Faires, J. D. (2011). *Numerical analysis* (9th ed.). Brooks/Cole, Cengage Learning.
2. Butcher, J. C. (2008). *Numerical methods for ordinary differential equations* (2nd ed.). John Wiley & Sons.
3. Chapra, S. C., & Canale, R. P. (2015). *Numerical methods for engineers* (7th ed.). McGraw-Hill Education.
4. Press, W. H., Teukolsky, S. A., Vetterling, W. T., & Flannery, B. P. (2007). *Numerical recipes: The art of scientific computing* (3rd ed.). Cambridge University Press.
5. Hecker, C. (1997). Physics-based motion. In M. DeLoura (Ed.), *Game programming gems* (pp. 292–303). Charles River Media.
6. Fiedler, G. (2014). Physics simulation: Integrating Newton's equations with Runge-Kutta. *Gaffer on Games*. https://gafferongames.com/post/integration_basics/
7. Hairer, E., Nørsett, S. P., & Wanner, G. (1993). *Solving ordinary differential equations I: Nonstiff problems* (2nd ed.). Springer.
8. Witkin, A., & Baraff, D. (1997). *Physically based modeling: Principles and practice*. SIGGRAPH Course Notes. Carnegie Mellon University.
9. Baraff, D. (1997). *An introduction to physically based modeling: Rigid body simulation I—Unconstrained rigid body dynamics*. Carnegie Mellon University.

10. Ferziger, J. H., & Perić, M. (2002). *Computational methods for fluid dynamics* (3rd ed.). Springer.
11. Kloeden, P. E., & Platen, E. (1992). *Numerical solution of stochastic differential equations*. Springer.
12. McKenna, P. J., & Reichel, W. (2002). *Numerical methods for ordinary differential equations: Initial value problems*. Springer.
13. Parent, R. (2012). *Computer animation: Algorithms and techniques* (3rd ed.). Morgan Kaufmann.
14. Wanner, G., & Hairer, E. (1996). *Solving ordinary differential equations II: Stiff and differential-algebraic problems* (2nd ed.). Springer.
15. Kass, M., & Miller, G. (1990). Rapid, stable fluid dynamics for computer graphics. *ACM SIGGRAPH Computer Graphics*, 24(4), 49–57. https://doi.org/10.1145/97880.97884
16. Battaglia, P. W., Pascanu, R., Lai, M., Rezende, D. J., & Kavukcuoglu, K. (2016). Interaction networks for learning about objects, relations, and physics. *Advances in Neural Information Processing Systems*, 29, 4502–4510.
17. Desbrun, M., Schröder, P., & Barr, A. H. (1999). Interactive animation of structured deformable objects. In *Proceedings of the Conference on Graphics Interface* (pp. 1–8).
18. Osher, S., & Fedkiw, R. (2003). *Level set methods and dynamic implicit surfaces*. Springer.
19. Sundararajan, D. (2015). *Numerical methods: Design, analysis, and computer implementation of algorithms*. Springer.
20. Shirley, P., Marschner, S. R., & others. (2009). *Fundamentals of computer graphics* (3rd ed.). A K Peters.
21. Hughes, J. F., van Dam, A., McGuire, M., Sklar, D. F., Foley, J. D., Feiner, S. K., & Akeley, K. (2013). *Computer graphics: Principles and practice* (3rd ed.). Addison-Wesley.
22. Wawrzynek, J. (2016). *Introduction to numerical methods*. University of California, Berkeley.
23. Müller, M., Heidelberger, B., Hennix, M., & Ratcliff, J. (2007). Position based dynamics. *Journal of Visual Communication and Image Representation*, 18(2), 109–118.
24. Sahani, S.K, Oruganti, S.K., K. Satish Kumar, and Karna, S.K.(2025). "Reliability Assessment of a Plywood Production Facility Utilizing Laplace Transform and Runge-Kutta Fourth-Order Differential Equations: Overview of Industrial Plant". *Metallurgical and Materials Engineering* 31 (4):417-23. https://doi.org/10.63278/1453
25. Praveenkumar, T., Anthoniraj, S., Kumarganesh, S., Somaskandan, M., Martin Sagayam, K., Pandey, B. K., ... & Sahani, S. K. (2025). Enhanced Circuit Board Analysis: Infrared Image Segmentation Utilizing Markov Random Field (MRF) and Level Set Techniques. *Engineering Reports*, *7*(3), e70029.
26. Chaudhary, A. K., Upadhaya, J., Nepal, B., Karki, M., Kandel, M., Kumar, A., ... & Sharma10, G. (2024). Cybersecurity in Network Traffic: Integrating Statistical Techniques with AI.
27. Sahani, S.K., Agarwal, S.K., Singh, V.V.(2025). Performance and Assessment of Jaw Crusher in a Cement Manufacturing Plant, Communications on Applied Nonlinear Analysis, Vol 32 No. 3,904-909
28. Sahani, S.K., Sah, B.K., Sahani, K. (2023), Reliability-Centered Maintenance (RCM) in Cement Manufacturing Plants, Advances in Nonlinear Variational Inequalities, Vol. 26, No.1, 2023, 16-25. DOI: https://doi.org/10.52783/anvi.v26.4224.
29. Sahani, S.K., Sah, B.K., Sahani, K., (2023) Statistical Reliability Modeling of Cement Kilns and Crushers, Panamerican Mathematical Journal, Vol 33No. 4(2023),115-120.
30. Rawal, D.K., Sahani, S.K., Singh, V.V., and Jibril, A.(2022), Reliability assessment of multi- computer system consisting n- clients and the k- out- of-n: G operation scheme

with copula repair policy, Life Cycle Reliability and safety engineering, 05 May 2022, DOI: "https://doi.org/10.1007/s41872-022-00192-5.

31. Sahani, K., Khadka, S. S., Sahani, S. K., Pandey, B. K., & Pandey, D. (2024). A possible underground roadway for transportation facilities in Kathmandu Valley: A racking deformation of underground rectangular structures. *Engineering Reports*, *6*(8), e12821.
32. Pandey, B. K., Pandey, D., & Sahani, S. K. (2025). Autopilot control unmanned aerial vehicle system for sewage defect detection using deep learning. *Engineering Reports, 7*(1), e12852. https://doi.org/10.1002/eng2.12852