Numerical Solution of Damped Harmonic Oscillator and Simulation using GNumeric Spreadsheets and Python Codes

Dr. Vedavathi Aluri

Department of Physics, Sri Sathya Sai Institute of Higher Learning
Email: *vedavathialuri[at]sssihl.edu.in*

Abstract: ***The damped harmonics oscillator problems have been solved numerically using classical Runge-Kutta method.The differential equations are solved in a Free Open Source Software (FOSS) GNumeric, a simple worksheet environment. The requisite algorithm is implemented in another FOSS python. Initial values for the simulation were taken from model parameters given in the book on C for Computer Simulation in Physics by R. C. Verma and Setul Verma, which were then optimized to obtain the best convergence for the particular problems.***

Keywords: Runge-Kutta method, FOSS, Python codes, Gnuemeric worksheets

1. Introduction

Every physics student studies classical mechanics and learns the analytical solutions of simple problems. But many interesting problems are difficult to solve analytically. Such problems can be solved using numerical techniques. The understanding of the theoritical model which is employed for solving the problems is the first step to any laboratory experiment, simulation or virtual experiment. In classical mechanics, a solution to a typical problem involves solving the equations of motion that could be obtained using Newton's second law or the Lagrangian approach [1]. This can be done by solving an initial value problem which is a second order differential equation for which initial position and initial velocity are given. In most cases, analytical techniques are available and we obtain the trajectory of the system with the capability to predict the position of the particle in the system at some future time, as the solution gives us r as r(t). Occasionally, we come across some cases for which the analytical solution is not available or is not easy to obtain, we approach the numerical techniques like Euler or Runge-Kutta methods [2] to solve the problem, which involve writing code in a programming language like python. In this paper, the numerical solutions of damped harmonic oscillator problem are obtained using Classical Runge-Kutta method and the the simulation has been done with GNumeric Spreadsheets, and python programming

2. Methodology

In GNuermic environment, the data can be accessed and manipulated using python programming. Since, both python and Gnumeric are free softwares, the source codes in GNumeric can be directly extended to the functions in python programming. Python offers a high-level abstraction through which to interact with the spreadsheet. Python and Gnumeric can be used in several ways (3). This paper describes how to implement GNumeric worksheet simulations to the Damped Harmonic Oscillator and python codes. The problem has been solved numerically using Classical Runge-Kutta method in this work.

The approach to modelling and simulation can be broadly divided into four steps:

- Modelling the Damped Harmonic problem.
- Implementation of the numerical method in a computer.
- Simulation of system by using FOSS.

The first step is elucidated by their application to the present study below. While the second and third steps are discussed under Results and Discussion.

2.1 Modelling of Damped Harmonic Oscillator

An ideal simple harmonic oscillator experiencing no dissipative force is difficult to encounter in the real world. The dissipative forces are always present which reduces the amplitude of the oscillationswith each cycl, dies our after some time, unless energy is supplied to sustain it. This is called damping, which is generally velocity dependent. Taking the force to be directly proportional to the velocity, the equation of motion for the damped oscillator becomes

$$m\frac{d^2x}{dt^2} = -kx - c\frac{dx}{dt} ------(1)$$

Where c demotes the damping coeficient and k denotes the force constamt. There are three different situations to the damped oscillator: Under damped ($\frac{c^2}{4m^2} < \frac{k}{m}$), Over damped ($\frac{c^2}{4m^2} > \frac{k}{m}$), Critically damped ($\frac{c^2}{4m^2} = \frac{k}{m}$) (4). Here the Classical Runge-Kutta method has been used for solving the differntial equation. It is considered that a mass m of 1.25 kg is attahced to a spring of force constant k=5 N/m is oscillating in a medium with damping coefficient c=0.2. The initial state of motion for the current situation is given by x(0)=0 m, v(0)=1.5 m/s. The time step is h= 0.05 s for the under damped oscillator and by changing the parameters, one can obtain the position and velocity of the oscillator in the three different cases.

3. Results and Discussion

3.1 Implementation of the numerical method in a computer:

Equation (1) is a II order Ordinary differential eqution. It can be resolved in to two first ordr ODEs as

$$\text{Let } \frac{dx}{dt} = v - - - - - - \qquad (2)$$

$$\frac{dv}{dt} = \frac{-k}{m}x - \frac{c}{m}v - - - - - - \qquad (3)$$

The classical Runge-Kutta method for the above two equations is given by

$$x_{i+1} = x_i + \frac{h}{6}[k_{11} + 2(k_{12} + k_{13}) + k_{14}]$$

$$v_{i+1} = v_i + \frac{h}{6}[k_{21} + 2(k_{22} + k_{23}) + k_{24}]$$

Here
$k_{11}, k_{12}, k_{13} k_{14,} k_{21,} k_{22,} k_{23}, k_{24}$ are constants to be calculated using GNumeric worksheets and Python codes.

3.2 Simulation of system

The damped harmonic oscillator problem after solving numerically using Runge-kutta method, was simulated with two FOSS. The constants were calculated using the GNumeric method and also python codes

3.2.1 Visualizing the position and velocity of Damped Harmoic Oscillator using GNumeric spreadsheets

The system parameters were chosen as force constant k=5 N/m, mass of the spring, m= 1.25000 kg, the initial position of the spring at the initial time t=0 was taken as 0 m/s and initial velocity as 1.5 m/s. The damping coefficient of the medium was considered as c=0.2 and the angular frequency was calculated as k/m. The system paramters were c Runge-Kutta fourth order method is a powerful numerical tool to solve ordinary differential equations numerically. The system parameters chosen for system are shown in Fig.1. After calculating the constants and substituting in the R-K formulae for the position and velocity, the value obtained at different times are given Fig.2. The graphs between posiiton and time and velocity and time resembled the expected graph which is shown in Fig.3 to visualize the position, velocity of the system at different times. This is most impressive when displayed by computer animation as the system oscillates by varying aplitudes.

The initial conditions are			
Force cons	K=	5.00000	N/m
mass	m=	1.25000	kg
initialtime	ti=	0.00000	s
final time	tf=	15.00000	s
no of steps	n=	300.00000	
initial posit	xi=	0.00000	m/s
initial velo	vi=	1.50000	m/s
step size	vi=	0.05000	
omega^2	vi=	4.00000	
damping c	vi=	0.20000	
mu=	vi=	1.25000	
	c/m=	0.16000	

Figure 1: The system paramters to implement R-K Method

	c/m=	0.16000										
Time	xi	vi	k11	k12	k13	k14	k21	k22	k23	k24	a
0	0.000000	1.50000	1.50000	1.494	1.490274	1.4806	−0.24	−0.38904	−0.38784384	−0.53495204928	−0.24000
0.05	0.074576	1.48059	1.48059	1.467214	1.463566	1.4466	−0.5352	−0.681119	−0.6791971583	−0.822479862979757	−0.25181
0.1	0.147816	1.44661	1.44661	1.42604	1.422506	1.3985	−0.82272	−0.964092	−0.9614697248	−1.099531487067097	−0.41623
0.15	0.219001	1.39850	1.39850	1.371002	1.367616	1.33	−1.09977	−1.235216	−1.2319247776	−1.363433236191647	−0.25880
0.2	0.287440	1.33685	1.33685	1.30276	1.299554	1.2624	−1.36366	−1.491887	−1.4879649668	−1.611663593933064	−0.64506
0.25	0.352473	1.26239	1.26239	1.222096	1.219101	1.1760	−1.61187	−1.731666	−1.7271570921	−1.841877093917578	−0.73069
0.3	0.413480	1.17596	1.17596	1.129912	1.127157	1.0786	−1.84207	−1.952301	−1.9472550095	−2.051926258634681	−0.80035
0.35	0.469886	1.07852	1.07852	1.027219	1.024728	0.971	−2.05211	−2.151749	−2.1462204941	−2.239881393099747	−0.85230
0.4	0.521166	0.97112	0.97112	0.915121	0.912917	0.855	−2.24004	−2.328195	−2.3222418672	−2.404048053472915	−0.88423
0.45	0.566851	0.85491	0.85491	0.794809	0.792912	0.7312	−2.40419	−2.480064	−2.4737502209	−2.542982035316247	−0.89458
0.5	0.606531	0.73112	0.73112	0.667546	0.665973	0.6011	−2.5431	−2.606043	−2.5994330968	−2.655501752092824	−0.88266
0.55	0.639858	0.60104	0.60104	0.534654	0.533417	0.4661	−2.6556	−2.705082	−2.698245504	−2.740697901514242	−0.84862
0.6	0.666553	0.46602	0.46602	0.3975	0.396609	0.3275	−2.74077	−2.776413	−2.7694181876	−2.797940345149637	−0.79345
0.65	0.686401	0.32743	0.32743	0.257484	0.256945	0.186	−2.79799	−2.819544	−2.8124630871	−2.826882155036694	−0.71897
0.7	0.699260	0.18669	0.18669	0.11602	0.115836	0.0453	−2.82691	−2.834273	−2.8271759524	−2.827460809597146	−0.62768
0.75	0.705058	0.04522	0.04522	−0.02547	−0.0253	−0.095	−2.82747	−2.820678	−2.8136361158	−2.799896549669755	−0.52272
0.8	0.703793	−0.09558	−0.09558	−0.16558	−0.16506	−0.234	−2.79988	−2.77912	−2.7722034422	−2.7446879336496	−0.40772
0.85	0.695534	−0.23431	−0.23431	−0.30293	−0.30207	−0.369	−2.74465	−2.710236	−2.703512513	−2.662604658285692	−0.28664

Figure 2: The calculated values of the constants

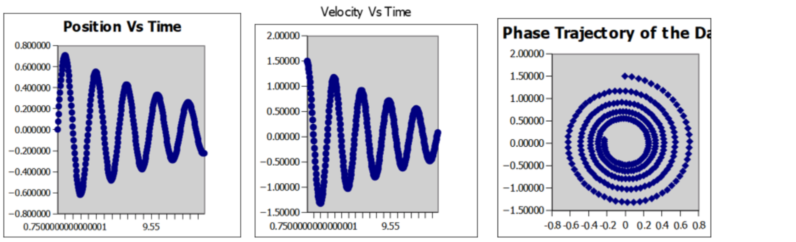

Fig.1. The graphs of Position Vs Time, Velocity Vs Time and Phase Trajectory (Under damped)

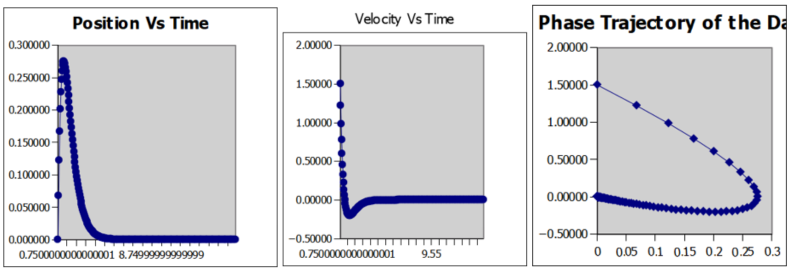

Figure 3: Plots obtained from calcuated values using GNuemeric worksheets

3.2.2 Python Simulation

Python is an interpreted, interactive, object-oriented high-level language. Its syntax resembles pseudo-code, especially because of the fact that indentation is used to indentify blocks. Python is a dynamcally typed language, and does not require variables to be declared before they are used. Variables "appear" when they are first used and "disappear" when they are no longer needed. Python is a scripting language like Tcl and Perl. Because of its interpreted nature, it is also often compared to Java. Unlike Java, Python does not require all instructions to reside inside classes. Python is also a multi-platform language, since the Python interpreter is available for a large number of standard operating systems, including MacOS, UNIX, and Microsoft Windows. Python interpreters are usually written in C, and thus can be ported to almost any platform which has a C compiler. To make the understanding more clear, the python codes, have been written using different modules available, for the damped oscillator with different conditions and visualized the position, velocity and amplitude of the system at different times. The plots are shown in Fig .4 which resemble the behavoiur of the damped oscillator.

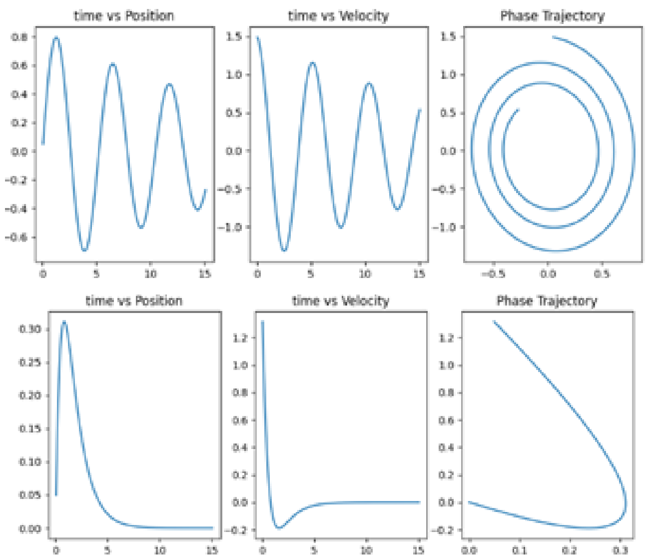

Figure 4: Plots obtained from python code

4. Conclusion

The numerical methods have been implanted to the damped harmonic oscillator to enhance the understanding of the students. By visualizing different damping conditions, one can understand that in over damped and critical damped conditions, the system will return to the equilibrium position most quickly. In conclusion, it is proposed that in an introductory mechanics course, numerical solutions can be shown either by computer animation or in graphs like figures showed in this paper.

References

[1] L. Verlet, 1967 Physics Review 159, 98
[2] R. W. Stanley 1984 Am. J. Phys. 52, 499 (1984)
[3] Aditi Sharma and O S K S Sastri, 2020 European J. Phys. 1, 41
[4] C for Computer Simulations in Physics, R.C. Vema, Setul Verma, ISBN: 978-81-89927-20-2