UNIVERSITAT POLITÈCNICA DE CATALUNYA
BARCELONATECH
Escola d'Enginyeria de Barcelona Est

Numerical simulation of the Damped Harmonic Oscillator ODEs

Marcel Fuertes Clavero
Universitat Politècnica de Catalunya
Escola d'Enginyeria de Barcelona Est (EEBE)
Numerical Simulation Applied to Engineering
marcel.fuertes@estudiantat.upc.edu
Course 2022/23

Abstract

We present a numerical analysis of the differential equations related to the Damped Harmonic Oscillator problems, distinguishing between the different cases depending on the value of the parameters. We apply the 4-th order Runge-Kutta algorithm for performing the analysis, comparing the obtained results with the theoretical solutions, which we mathematically detail how are obtained too.

Contents

1 Introduction **3**

2 Problem description **3**

2.1 Application of the theory of the damping oscillator to the real world 5

2.2 Analytical solution to the ODEs 6

3 Fourth Order Runge-Kutta method **8**

3.1 Runge-Kutta for our particular problem 9

4 Experiments **9**

4.1 Computation of the constants using the initial conditions 10

4.2 Simple harmonic oscillator 10

4.3 Damped Oscillatory motion 11

4.4 Forced Damped Oscillator 12

5 Conclusions and future work **13**

A Appendix **15**

1 Introduction

In classical mechanics, the damped harmonic oscillator is a classic problem that describes the movement of a mechanical oscillator under the influence of a restoring force and friction. Depending on several factors, this problem is separated into different cases, which can all be modeled with ordinary differential equations. The most general way of writing all cases is

$$m\frac{d^2x(t)}{dt^2} + b\frac{dx(t)}{dt} + m\omega_0^2 x(t) = F_0\cos(\omega t),$$

where $x(t)$ denotes the position of the object at time t, $m > 0$ the mass of the object, $b \geq 0$ the damping term, $\omega_0 \in \mathbb{R}$ the undamped angular frequency of the oscillator, F_0 the amplitude of the force and $\omega \in \mathbb{R}$ the driving angular frequency. Depending on whether b or F_0 are different from zero or not we can separate several cases. In this work, we will mathematically study all cases applying a numerical method.

Section 2 describes the theoretical approach of the problem, giving a physical explanation of the phenomenon and a mathematical view of how the differential equations are obtained. We will also present an application of the theory to the real world, studying in particular the case of RLC circuits. After that, we will present the mathematical procedure for obtaining the analytical solutions of the differential equations for each case.

Once all the theory related to the Damped Harmonic Oscillator has been seen, we present in Section 3 an overview of the numerical method that we will apply for solving the ODEs, the 4-th order Runge-Kutta algorithm.

Section 4 presents the results of the experiments, including many of the plots for the different cases as well as a study of the errors compared to the analytical solutions. A comment we should highlight is that, although it is more common to use Matlab in the engineering community for these type of studies, we have chosen to use C. We have taken this path for several reasons, but the main ones are that C is faster and open-source, unlike Matlab, which is expensive. If the reader wants to see how thw code is, what it does and how to compile it, see Appendix A. All the material that has been used for the experiments can be found in the following Github repository: `https://github.com/MarcelFuertes/Forced_mechanical_oscillator`.

2 Problem description

In this section, we will see a theoretical description of the problem we are working with, both mathematically and from a physical point of view. For each of the cases that we will study, we will present the ordinary differential equation that are associated with, as well as the procedure to obtain them. This section is inspired by [1] , in case the reader wants to read more about the subject.

We start with the simplest case that is the *simple harmonic oscillator*: Let's consider the block-spring system where the block is initially stretched an amount $x_0 > 0$ from the equilibrium position and is released from rest, $v_{x,0} = 0$. There are three states that we shall study: state 1, the initial state; state 2, at an arbitrary time in which the position and velocity are different from zero; and state 3, when the object first comes back to the equilibrium position. We shall show that the mechanical energy is equal for each of these states and has no variance throughout the motion. Choose the equilibrium position for the zero point of the potential energy.

- State 1: all the energy is stored in the object-spring potential energy E_p. The object is released from rest so the kinetic energy E_k is zero, the total mechanical energy E_1 is then

  $$E_1 = E_p = \frac{1}{2}kx_0^2,$$

  where k is a positive real number, characteristic of the spring.
- State 2: at some time t, the position and x-component of the velocity of the object of mass m are given by

  $$x(t) = x_0\cos\left(\sqrt{\frac{k}{m}}t\right), \qquad v_x(t) = -\sqrt{\frac{k}{m}}x_0\sin\left(\sqrt{\frac{k}{m}}t\right).$$

The mechanical energy is the sum of the kinetic and potential energies, so

$$E_2 = E_k + E_p = \frac{1}{2}mv_x^2 + \frac{1}{2}kx^2 = \frac{1}{2}kx_0^2 \left(\underbrace{\cos^2\left(\sqrt{\frac{k}{m}}t\right) + \sin^2\left(\sqrt{\frac{k}{m}}t\right)}_{=1} \right)$$
$$= \frac{1}{2}kx_0^2.$$

The mechanical energy in state 2 is equal to the initial potential energy in state 1, so the mechanical energy is constant. We isolated the object spring system so that there is no external work performed on the system and no internal non-conservative forces doing work, so it is what we expected.

- State 3: suppose that the object is at the equilibrium position, then the potential energy is zero, and the mechanical energy is in the form of kinetic energy (see Figure 1):

$$E_3 = E_k = \frac{1}{2}mv_{eq}^2.$$

The system is closed, so the mechanical energy is constant, i.e. $E_1 = E_3$. Therefore the initial stored potential energy is released as kinetic energy, i.e. $\frac{1}{2}kx_0^2 = \frac{1}{2}mv_{eq}^2$.

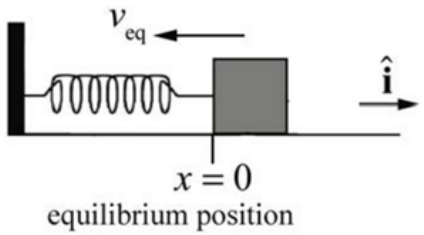


Figure 1: State 3 at equilibrium and in motion.

We have shown that the mechanical energy is constant at all times. The mechanical energy at an arbitrary time is given by

$$E = K + U = \frac{1}{2}mv_x(t)^2 + \frac{1}{2}kx(t)^2.$$

If we differentiate this equation, the result is zero (since E is constant), giving us the differential equation:

$$\frac{dE(t)}{dt} = mv_x(t)\frac{dv_x(t)}{dt} + kx(t)\frac{dx(t)}{dt} = v_x(t)\left(m\frac{d^2x(t)}{dt^2} + kx(t)\right) = 0$$
$$\iff m\frac{d^2x(t)}{dt^2} + kx(t) = 0. \tag{1}$$

Now let's study the second case: the *damped oscillatory motion*. Consider our spring-block system moving on a horizontal frictionless surface, but now attaching the block to a damper that resists its motion due to viscous friction (see Figure 2). The viscous force arises when objects move through fluids at speeds slow enough so that there is no turbulence. The viscous force opposes the motion and is proportional to the velocity, so that the damper is referred to as a linear damper. The constant of proportionality $b > 0$ depends on the properties of the damper.

$$\overrightarrow{\mathbf{f}_{vis}} = -b\overrightarrow{\mathbf{v}}.$$

Set the origin at the equilibrium position and choose the positive x-direction to the right in Figure 2. Let $x = x(t)$ be the position of the object with respect to the equilibrium position. The x-component

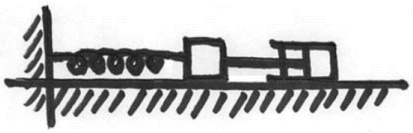


Figure 2: Spring-block system connected to a linear damper.

of the total force acting on the spring is then the sum of the linear restoring spring force, and the viscous friction force (see Figure 3).

$$F_x = -kx - b\frac{dx}{dt}.$$

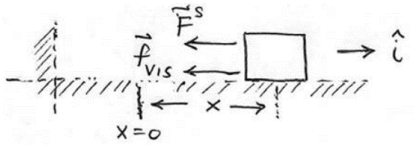


Figure 3: Free-body force diagram for spring-object system with linear damper.

Newton's second law states that any change in the motion of a body is proportional to the force acting on it and that this change in movement always takes place in the same direction in which the force acts. So, Newton's Second law in the x-direction becomes

$$\frac{d^2x}{dt^2} + \frac{b}{m}\frac{dx}{dt} + \frac{k}{m}x = 0. \tag{2}$$

The third and last case is the *forced damped oscillator*:

Let's drive our damped spring-object system by a sinusoidal force. Suppose that the x-component of the driving force is given by

$$F_x(t) = F_0 \cos(\omega t),$$

where $F_0 \in \mathbb{R}$ and $\omega \in \mathbb{R}$ is the driving angular frequency. The force varies between F_0 and $-F_0$ because $\cos(\omega t) \in [-1, 1]$. Let $x = x(t)$ be the position of the object with respect to the equilibrium position. The x-component of the force acting on the object is now the sum

$$F_x = F_0 \cos(\omega t) - kx - b\frac{dx}{dt}.$$

Newton's Second law in the x-direction becomes

$$m\frac{d^2x(t)}{dt^2} + b\frac{dx(t)}{dt} + kx(t) = F_0 \cos(\omega t). \tag{3}$$

2.1 Application of the theory of the damping oscillator to the real world

The damped harmonic oscillator equations, even if they seem to only work in a theoretical world, have applications too, which are indirectly part of our daily life. They model objects literally oscillating while immersed in a fluid as well as more abstract systems in which quantities oscillate while losing energy. In physics, there is an empirical law which states that the force F needed to extend or compress a spring by some distance x scales linearly with respect to that distance. Mathematically, that is

$$F = -kx,$$

where k is a constant factor characteristic of the spring. This law is known as *Hooke's law*. Our models are great for many physical systems because most systems both obey Hooke's law when

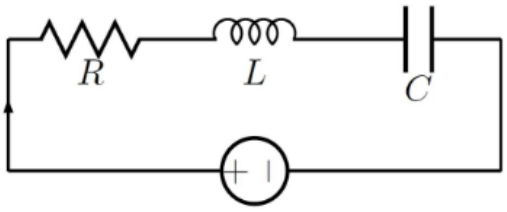


Figure 4: RLC circuit.

perturbed about an equilibrium point and also lose energy as they decay back to equilibrium. These two conditions are sufficient to obey the equation of motion of the damped harmonic oscillator.

A classical example is a circuit with an inductor, capacitor, and resistor [2].

Kirchoff's laws states that the sum of voltages in a closed loop must be zero. In order to avoid external affectations (and for convenience) we will assume that the capacitor is charged and no external voltage source is connected to the circuit. The voltages for each component of the RLC circuit depend on the charge (name it $Q = Q(t)$) on the capacitor and I, the current admitted through the circuit. When the capacitor is discharging, $I = -\frac{dQ}{dt}$. The equations are

$$\begin{cases} V_R &= IR \\ V_C &= -\frac{Q}{C} \\ V_I &= L\frac{dI}{dt}, \end{cases}$$

where the voltage across the capacitor opposes the flow of conventional current (see Figure 4). Applying Kirchoff's law, $V_R + V_C + V_I = 0$, so if we substitute, we have that

$$L\frac{d^2Q}{dt^2} + R\frac{dQ}{dt} + \frac{1}{C}Q = 0.$$

The time-dependence of the charge on the capacitor thus behaves like a damped harmonic oscillator.

2.2 Analytical solution to the ODEs

We present an analytical solution for (3), inspired by [3–5], which covers all the possible cases for the different values of the parameters. We should remark that $m, b, k > 0$ and that $k := m\omega_0^2$, where ω_0 is called the "undamped angular frequency of the oscillator",

First of all, in order to lighten the notation, we set $x = x(t)$, so our differential equation is now written as

$$m\frac{d^2x}{dt^2} + b\frac{dx}{dt} + kx = F_0\cos(\omega t).$$

We are in front of a second order differential equation, whose general solution will be the sum of the homogeneous equation (name it $x_h(t)$) and a particular solution (name it $x_p(t)$). This solution is not unique: there are two constants of integration, which are determined by specifying the initial position and velocity. However, as we will prove below, the equation does have a unique steady state solution with $x_p(t)$ oscillating at the same frequency as the external drive. Those "undriven solutions" die away as time goes on. So, we can add such a solution to fit the specified initial conditions, and after a while the system will lose memory of those conditions and settle into the steady driven solution.

Let's determine the particular solution $x_p(t)$. We begin by writing

$$\text{external driving force} = F_0 e^{i\omega t},$$

with $F_0 \in \mathbb{R}$, so $\text{Re}\left(F_0 e^{i\omega t}\right) = F_0\cos(\omega t)$. We will solve the complex differential equation

$$m\frac{d^2z}{dt^2} + b\frac{dz}{dt} + kz = F_0 e^{i\omega t} \tag{4}$$

and then take the real part of the solution, which is precisely the solution of (3). We start by trying the complex solution $z(t) = Ae^{i(\omega t+\varphi)}$, imposing $A, \varphi \in \mathbb{R}$. We can always take the amplitude A to be

real: that is not a restriction, since we've added the adjustable phase factor $e^{i\varphi}$. If $z(t) = Ae^{i(\omega t+\varphi)}$ is a solution of (4), then its real part $A\cos(\omega t + \varphi)$ will be a solution of (3). Substituting $z(t)$ in (4), we obtain

$$-m\omega^2 Ae^{i(\omega t+\varphi)} + ib\omega Ae^{i(\omega t+\varphi)} + kAe^{i(\omega t+\varphi)} = F_0 e^{i\omega t},$$

which gives us

$$A = \frac{F_0 e^{-i\varphi}}{k - m\omega^2 + ib\omega}.$$

The denominator of A, in polar form, is written as $re^{i\theta}$, where $r = \sqrt{\left(k - m\omega^2\right)^2 + (b\omega)^2}$ and $\theta = \arctan\left(\frac{b\omega}{k-m\omega^2}\right)$, so now A can be written as $A = \frac{F_0}{r}e^{-i(\varphi+\theta)}$. We imposed A to be real, so this implies that $\varphi = -\theta$, and consequently $A = \frac{F_0}{r}$. Our particular solution is now written as $z(t) = \frac{F_0}{r}e^{i(\omega t-\theta)}$. Taking real parts and undoing all variable changes, we obtain that a particular solution of (3) is

$$x_p(t) = \frac{F_0}{\sqrt{m^2\left(\omega_0^2 - \omega^2\right)^2 + (b\omega)^2}} \cos\left(\omega t - \arctan\left(\frac{b\omega}{m\left(\omega_0^2 - \omega^2\right)}\right)\right).$$

Let's now calculate the general solution $x_h(t)$ for the homogeneous equation [1][2]. That is,

$$m\frac{d^2x}{dt^2} + b\frac{dx}{dt} + m\omega_0^2 x = 0. \tag{5}$$

We assume that a solution will be proportional to $e^{\lambda t}$ for some constant λ and substitute it in (5). After some basic algebraic manipulations, we obtain:

$$\left(m\lambda^2 + b\lambda + m\omega_0^2\right)e^{\lambda t} = 0.$$

Since $e^{\lambda t} \neq 0$, the zeros must come from the polynomial, so we solve for λ and we obtain

$$\begin{cases} \lambda_1 = -\dfrac{b}{2m} - \dfrac{\sqrt{b^2 - 4m^2\omega_0^2}}{2m} \\ \lambda_2 = -\dfrac{b}{2m} + \dfrac{\sqrt{b^2 - 4m^2\omega_0^2}}{2m}. \end{cases}$$

This general solution forces us to distinguish between three cases, which are $b^2 > 4m^2\omega_0^2$, the *overdamped case*; $b^2 = 4m^2\omega_0^2$, the *critically damped case*; and $b^2 < 4m^2\omega_0^2$, the *underdamped case*[3]. For the first two cases, observe that, since $b, m > 0$, we clearly see that $\lambda_1, \lambda_2 \leq 0$. For the underdamped case, $\lambda_1, \lambda_2 \in \mathbb{C}$.

We start with the overdamped case. The root λ_1 gives $x_1(t) = c_1 e^{-t\left(\frac{b}{2m} + \frac{\sqrt{b^2-4m^2\omega_0^2}}{2m}\right)}$ as a solution, where c_1 is an arbitrary constant. Analogously, $x_2(t) = c_2 e^{-t\left(\frac{b}{2m} - \frac{\sqrt{b^2-4m^2\omega_0^2}}{2m}\right)}$ is another solution. The general solution of the homogeneous equation is the sum of $x_1(t)$ and $x_2(t)$, i.e. $x_h(t) = x_1(t) + x_2(t)$. Observe that, as we mentioned before, $x_h(t)$ dies away as time goes on.

Summarizing all, we have obtained that the general solution of (3), when $b^2 > 4m^2\omega_0^2$ is

$$\begin{aligned} x(t) =& c_1 e^{-t\left(\frac{b}{2m} + \frac{\sqrt{b^2-4m^2\omega_0^2}}{2m}\right)} + c_2 e^{-t\left(\frac{b}{2m} - \frac{\sqrt{b^2-4m^2\omega_0^2}}{2m}\right)} \\ &+ \frac{F_0}{\sqrt{m^2\left(\omega_0^2 - \omega^2\right)^2 + (b\omega)^2}} \cos\left(\omega t - \arctan\left(\frac{b\omega}{m\left(\omega_0^2 - \omega^2\right)}\right)\right). \end{aligned} \tag{6}$$

[1] Observe that, dividing (5) by m, $x_h(t)$ is also the solution of the *damped oscillatory motion* (2). In general, the solutions of the damped oscillatory motion are the same than the forced harmonic oscillator but setting $F_0 = 0$.

[2] In order to be consistent, from now on, we associate equation (5) to the damped oscillatory motion.

[3] A general schema on how the solutions are depending on the form of the roots of the characteristic polynomial can be found in `https://www.storyofmathematics.com/second-order-homogeneous-differential-equation/` .

In the second case, which happens when $b^2 = 4m^2\omega_0^2$, we have that $\lambda_1 = \lambda_2 = -\frac{b}{2m}$. In this case, the general solution of the homogeneous equation is given by $x_3(t) = e^{-\frac{b}{2m}t}(c_1 + c_2 t)$, and so the general solution of (3) is

$$x(t) = e^{-\frac{b}{2m}t}(c_1 + c_2 t) + \frac{F_0}{\sqrt{m^2\left(\omega_0^2 - \omega^2\right)^2 + (b\omega)^2}} \cos\left(\omega t - \arctan\left(\frac{b\omega}{m\left(\omega_0^2 - \omega^2\right)}\right)\right). \tag{7}$$

For the underdamped case, which happens when $b^2 < 4m^2\omega_0^2$, the roots are[4] $\lambda_1 = -\frac{b}{2m} - \frac{\sqrt{|b^2 - 4m^2\omega_0^2|}}{2m}i$, $\lambda_2 = -\frac{b}{2m} + \frac{\sqrt{|b^2 - 4m^2\omega_0^2|}}{2m}i$. The general solution of the homogeneous equation is given by $x_4(t) = e^{-\frac{b}{2m}t}\left(c_1 \cos\left(\frac{\sqrt{|b^2 - 4m^2\omega_0^2|}}{2m}t\right) + c_2 \sin\left(\frac{\sqrt{|b^2 - 4m^2\omega_0^2|}}{2m}t\right)\right)$, and summarizing all, the general solution of (3) for this case is

$$\begin{aligned} x(t) = {} & e^{-\frac{b}{2m}t}\left(c_1 \cos\left(\frac{\sqrt{|b^2 - 4m^2\omega_0^2|}}{2m}t\right) + c_2 \sin\left(\frac{\sqrt{|b^2 - 4m^2\omega_0^2|}}{2m}t\right)\right) \\ & + \frac{F_0}{\sqrt{m^2\left(\omega_0^2 - \omega^2\right)^2 + (b\omega)^2}} \cos\left(\omega t - \arctan\left(\frac{b\omega}{m\left(\omega_0^2 - \omega^2\right)}\right)\right). \end{aligned} \tag{8}$$

Another case that needs to be treated aside is the *simple harmonic oscillator* (1), that is, when $F_0, b = 0$. Observe that this case is much simpler than the previous ones. The characteristic equation for this case is

$$m\lambda^2 + k = 0,$$

and its roots are $\lambda = \pm\sqrt{\frac{k}{m}}i$. Then, the solution to the differential equation is as follows:

$$x(t) = c_1 \cos\left(\sqrt{\frac{k}{m}}t\right) + c_2 \sin\left(\sqrt{\frac{k}{m}}t\right) \overset{k=m\omega_0^2}{=} c_1 \cos(\omega_0 t) + c_2 \sin(\omega_0 t). \tag{9}$$

3 Fourth Order Runge-Kutta method

The 4th order Runge-Kutta method is an iterative method used in temporal discretization for the approximate solutions of simultaneous nonlinear equations. It was developed around 1900 by Carl Runge and Wilhelm Kutta [6, 7].

Suppose we have the following initial value problem:

$$\begin{cases} \dfrac{dx(t)}{dt} & = f(t, x) \\ x(t_0) & = x_0 \end{cases}$$

where $x(t)$ is an unknown function of time t, which we would like to approximate. We set a step-size $h > 0$ and we define

$$\begin{aligned} x_{n+1} &= x_n + \frac{h}{6}\left(k_1 + 2k_2 + 2k_3 + k_4\right), \\ t_{n+1} &= t_0 + (n+1)h, \end{aligned}$$

[4] Knowing that $b^2 - 4m^2\omega_0^2 < 0$, observe that $b^2 - 4m^2\omega_0^2 = -\left|b^2 - 4m^2\omega_0^2\right|$, and so $\sqrt{b^2 - 4m^2\omega_0^2} = \sqrt{-\left|b^2 - 4m^2\omega_0^2\right|} = i\sqrt{|b^2 - 4m^2\omega_0^2|}$.

where

$$
\begin{aligned}
k_1 &= f\left(t_n, x_n\right), \\
k_2 &= f\left(t_n + \frac{h}{2}, x_n + h\frac{k_1}{2}\right), \\
k_3 &= f\left(t_n + \frac{h}{2}, x_n + h\frac{k_2}{2}\right), \\
k_4 &= f\left(t_n + h, x_n + hk_3\right).
\end{aligned}
$$

Here, k_1 is the slope at the beginning of the interval, using x. k_2 is the slope at the midpoint of the interval, using x and k_1. k_3 is again the slope at the midpoint, but using x and k_2. k_4 is the slope at the end of the interval, using x and k_3. When averaging the four slopes, greater weight is given to the slopes at the midpoint. The pseudocode of this procedure can be found in Algorithm 1.

Algorithm 1: 4-th order Runge-Kutta method

Input: h, $t^{(0)}$, $x^{(0)}$, n, pre-defined **f(t,x)**.
Do
$\forall k = 0, 1, 2, \ldots, n-1$:

Compute k_1 , k_2 , k_3 and k_4:
$k_1 = hf(t^{(k)}, x^{(k)})$
$k_2 = hf(t^{(k)} + \frac{h}{2}, x^{(k)} + \frac{k_1}{2})$
$k_3 = hf(t^{(k)} + \frac{h}{2}, x^{(k)} + \frac{k_2}{h})$
$k_4 = hf(t^{(k)} + h, x^{(k)} + k_3)$

Compute $\phi^k(h, f)$:
$\phi^k = \frac{1}{6}(k_1 + 2k_2 + 2k_3 + k_4)$

Update $\mathbf{x}$:
$x^{(k+1)} := x^{(k)} + \phi^{(k)}$

Update $\mathbf{t}$:
$t^{(k+1)} := t^{(0)} + (k+1)h$
Stop
Output: $k+1$, $t^{(k+1)}$, $x^{(k+1)}$ (in case of success)

3.1 Runge-Kutta for our particular problem

For our particular problem, we are dealing with a second order differential equation, so we need to apply a previous step in order to apply Runge-Kutta. We apply a change of variable, $v = x'(t)$, so now equation (3) is written as the following system:

$$
\begin{cases}
x'(t) &= v \\
v'(t) &= -\dfrac{b}{m}v - \omega_0^2 x + \dfrac{F_0}{m}\cos(\omega t).
\end{cases}
$$

Now, two equations are coupled and we must solve them at the same time, so, when we code Runge-Kutta, we will have to introduce other k_i's. Also, we name $g(v) = v$ and $f(x, v, t) = -\frac{b}{m}v - \omega_0^2 x + \frac{F_0}{m}\cos(\omega t)$.

4 Experiments

In this section, we present the results of all experiments for each case, distinguishing between the simple harmonic oscillator, the damped oscillatory motion and the forced damped oscillator. The C code that we have written allows to separate cases based on the definition of the values for each parameter.

4.1 Computation of the constants using the initial conditions

When we compare our obtained results with the theoretical ones, there is a previous step that needs to be done in order to determine the exact form of the analytical solution. As we have seen in subsection 2.2, the analytical solutions depend on two constants that can be calculated using the initial conditions of the differential equation. In our case, these are $x(0) = 0$ and $x'(0) = 1$. Even if the solutions look huge, the computation and evaluation in $t = 0$ of the derivatives are straightforward. Using the even property of $\sin(x)$ and the odd property of $\cos(x)$, we can simplify the computation of $x(0)$ and $x'(0)$.

We start with the overdamped case, when $b^2 > 4m^2\omega_0^2$. Setting $\lambda_1 = \frac{b}{2m} + \frac{\sqrt{b^2-4m^2\omega_0^2}}{2m}$, $\lambda_2 = \frac{b}{2m} - \frac{\sqrt{b^2-4m^2\omega_0^2}}{2m}$, $\alpha = \frac{F_0}{\sqrt{m^2\left(\omega_0^2-\omega^2\right)^2+(b\omega)^2}}$ and $\theta = \arctan\left(\frac{b\omega}{m\left(\omega_0^2-\omega^2\right)}\right)$, we have that

$$\begin{cases} x(0) &= c_1 + c_2 + \alpha\cos(-\theta) = c_1 + c_2 + \alpha\cos(\theta) = 0 \\ x'(0) &= -\lambda_1 c_1 - \lambda_2 c_2 - \alpha\omega\sin(-\theta) = -\lambda_1 c_1 - \lambda_2 c_2 + \alpha\omega\sin(\theta) = 1. \end{cases}$$

The computation of c_1 and c_2 is just a simple linear equation system, and its solution is

$$\begin{aligned} c_1 &= \frac{\alpha\lambda_2\cos(\theta) + \alpha\omega\sin(\theta) - 1}{\lambda_1 - \lambda_2}, \\ c_2 &= \frac{\alpha\lambda_1\cos(\theta) + \alpha\omega\sin(\theta) - 1}{\lambda_2 - \lambda_1}. \end{aligned}$$

Now let's study the case $\lambda_1 = \lambda_2 = \lambda$ (i.e. $b^2 = 4m^2\omega_0^2$). We have that

$$\begin{cases} x(0) &= c_1 + \alpha\cos(\theta) = 0 \\ x'(0) &= \lambda c_1 + c_2 + \omega\alpha\sin(\theta) = 1. \end{cases}$$

The solution to this system is

$$\begin{aligned} c_1 &= -\alpha\cos(\theta), \\ c_2 &= \alpha\lambda\cos(\theta) - \alpha\omega\sin(\theta) + 1. \end{aligned}$$

For the underdamped case (i.e. when $b^2 < 4m^2\omega_0^2$) we have that

$$\begin{cases} x(0) &= c_1 + \alpha\cos(\theta) = 0 \\ x'(0) &= -\frac{b}{2m}c_1 + \frac{\sqrt{|b^2 - 4m^2\omega_0^2|}}{2m}c_2 + \omega\alpha\sin(\theta) = 1. \end{cases}$$

The solution to this system is

$$\begin{aligned} c_1 &= -\alpha\cos(\theta), \\ c_2 &= -\frac{\alpha b\cos(\theta) + 2\alpha m\omega\sin(\theta) - 2m}{\sqrt{|b^2 - 4m^2\omega_0^2|}}. \end{aligned}$$

For the *simple harmonic oscillator* (1), the initial condition $x(0) = 0$ gives us that $c_1 = 0$, and so, using $x'(0) = 1$, we end up with $c_2 = \frac{1}{\omega_0}$.

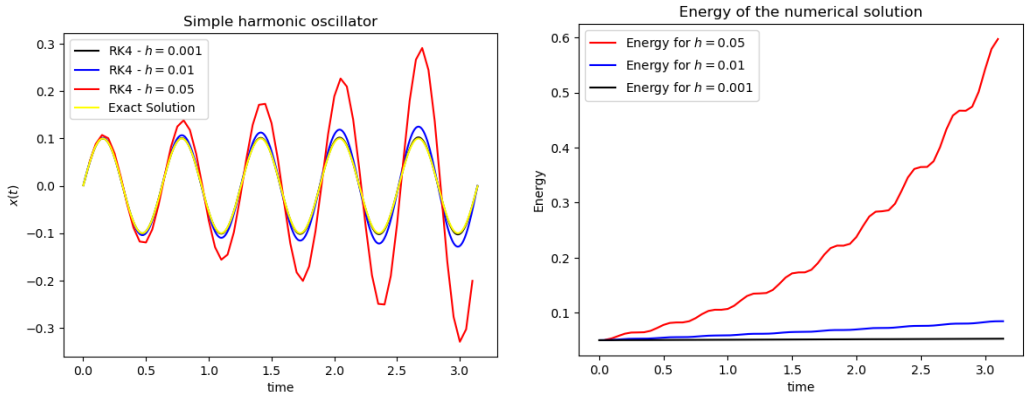
4.2 Simple harmonic oscillator

We start the experiments with the simple harmonic oscillator case (1). That is, when $F_0 = 0$ (i.e. no external force) and $b = 0$ (i.e. no damping). For the following experiments, we have set $\omega_0 = 10$, $m = 0.1$, $\omega = 0.4\omega_0$ and an integration range of $0 < t < 5T_0$, where $T_0 = \frac{2\pi}{\omega_0}$ is the natural period of the oscillator. The code returns, for each step, the value of the Runge-Kutta algorithm (RK4) as well as the value of the analytical solution (9), which allows us to make a precise comparison of the errors (see Appendix A for the details of the code). We have considered the absolute error and the relative error; the first one is defined as $|x_{\text{exact}} - x_{\text{approx}}|$, and the second one is defined as

$\left|\frac{x_{\text{exact}} - x_{\text{approx}}}{x_{\text{exact}}}\right|$ when $x_{\text{exact}} \neq 0$. For each value of h, we have saved the mean value of the errors for all steps.

For this subsection, we have compared the performance of RK4 for different values of h against the values of the analytical solution. We have also studied whether energy is conserved over time. In order to obtain a nearly perfect solution, a small step-size h was needed. As we can see in Table 1, the relative errors have been significant when $h = 0.1$ and $h = 0.01$. These high errors can be appreciated too in Figure $5(a)$, where we can observe that for larger values of h, the numerical solution gets away from the analytical one as time increases. In particular, the algorithm loses precision when the derivative of the analytical solution changes sign, that is, when there is a maximum or a minimum. Even though, the results with $h = 0.001$ are so precise compared to the real solution that it is difficult to appreciate in Figure $5(a)$ the points of RK4.

In our numerical solution, the energy is conserved quite well over time for small values of h. In fact, when we plot the function $E(t) = \frac{1}{2}mv(t)^2 + \frac{1}{2}kx(t)^2$, we can observe in Figure $5(b)$ that the value is nearly constant for $h = 0.01$ and $h = 0.001$. Since we are dealing with a numerical solution and not an analytical one, there appear precision errors that make the conservation not to be perfect, but the difference in the variation is negligible.



(a) Exact solution vs RK4

(b) Mechanical energy

Figure 5: Figure (a) shows the comparison of the obtained results of the algorithm for different values of h against the plot of the exact result (9). Figure (b) plots the values of the mechanical energy over time for the same values of h.

Table 1: Results of the computation of the 4-th order Runge-Kutta method for the simple harmonic oscillator case varying the value of h.

h	Mean absolute error	Maximum error	Minimum error	Mean relative error
0.05	0.066205	0.230419	0.001046	1.70762
0.01	0.009205	0.028434	6.468988×10^{-6}	0.152985
0.001	0.000845	0.002532	9.257258×10^{-8}	0.017124

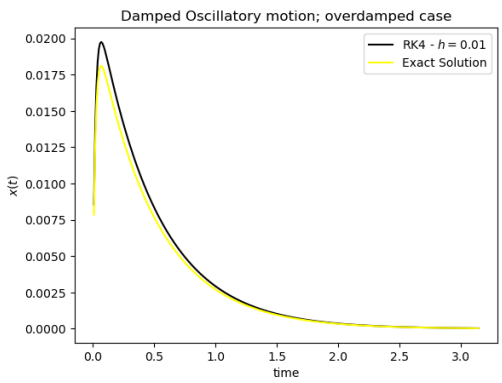
4.3 Damped Oscillatory motion

The second kind of experiments are related to the *damped oscillatory motion* case (5). That is, when $b > 0$ and $F_0 = 0$. We have set $m = 0.1$, $\omega_0 = 10$ and $\omega = 0.4\omega_0$, varying the value of b for distinguishing between the three principal subcases: overdamping, critically damping and underdamping. The integration range remains as $0 < t < 5T_0$, where $T_0 = \frac{2\pi}{\omega_0}$. For this subsection, we have studied these three principal subcases comparing the results of RK4 with the analytical solutions for each subcase, whose plots can be seen in Figure 6. The precision of the algorithm for $h = 0.01$ has been so good that the plots of the analytical solutions and the RK4 points are practically the same, with some slight differences in the extrema. Table 2 shows the exact values of the errors for each subcase.

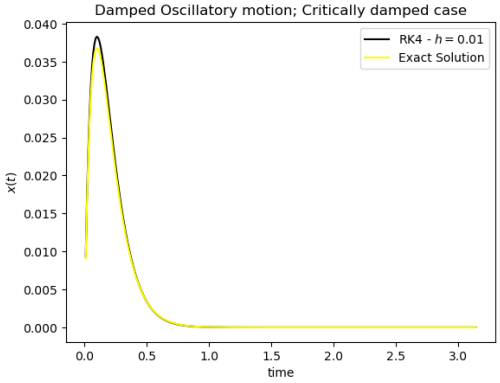
Table 2: Results of the computation of the 4-th order Runge-Kutta method for the three subcases, setting $h = 0.01$.

Case	Mean absolute error	Maximum error	Minimum error	Mean relative error
Overdamped	0.000275	0.001654	1.704795×10^{-6}	0.073789
Critically Damped	0.000118	0.001522	9.402785×10^{-14}	0.588876
Underdamped	0.00160	0.003424	1.96123×10^{-6}	0.22613

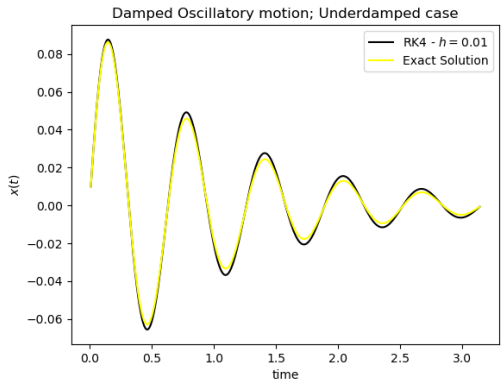
For the overdamped and critically damped cases, it can be seen that the solution decays without oscillating. The critically damped solution decays faster than the overdamped one. For the underdamped case, it can be seen that the solution oscillates at real frequency $\frac{\sqrt{|b^2-4m^2\omega_0^2|}}{2m}$, but also decays exponentially in time at a rate proportional to the damping coefficient, in particular, proportional to $\frac{b}{2m}$.



(a) Overdamped case for $b = 5$



(b) Critically damped case for $b = 2$



(c) Underdamped case for $b = 0.2$

Figure 6: Comparison of the obtained results of the algorithm for $h = 0.01$ against the plots of the analytical results.

4.4 Forced Damped Oscillator

For the Forced Damped Oscillator, we set $F_0 = 1$. As we have explained before, now the solutions of the ODE are the sum of the homogeneous equation and a particular one. The homogeneous solutions rapidly die away as time goes on, so there is no necessity of studying separately the three subcases as before, since they are essentially equal in practice.

Table 3: Results of the computation of the 4-th order Runge-Kutta method for different values of h.

h	Mean absolute error	Maximum error	Minimum error	Mean relative error
0.05	0.001909	0.006606	3.41567×10^{-5}	0.052241
0.01	0.000348	0.00117	1.723197×10^{-6}	0.011263

In Figure 7, we can compare the results of the algorithm with $h = 0.05$ and $h = 0.01$ with the analytical solution (7). Both are pretty accurate, in fact, the points of RK4 for $h = 0.01$ are practically the same than the analytical solution and are difficult to see in the plot. Table 3 shows the values of the errors for each step-size h.

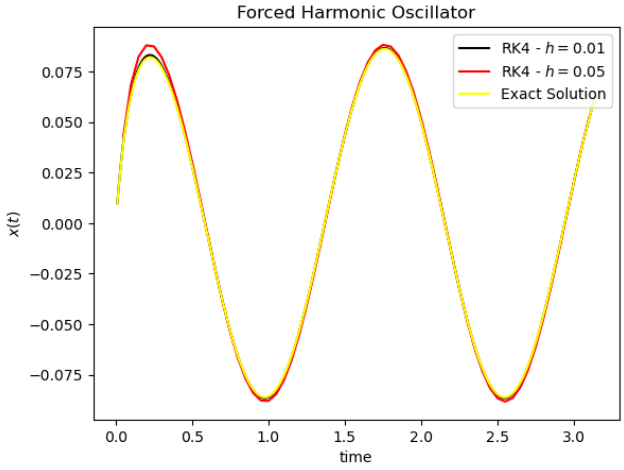


Figure 7: Comparison of the obtained results of the algorithm for different values of h against the plot of the analytical results, setting $b = 2$.

There is phenomenon known as *resonance*, which occurs when the driving frequency equals the natural frequency and the damping is small. If we recap up to subsection 2.2, we calculated a parameter A, which is called the amplitude. We can think of this parameter as a function of ω, since we have fixed the value of the rest of the parameters and

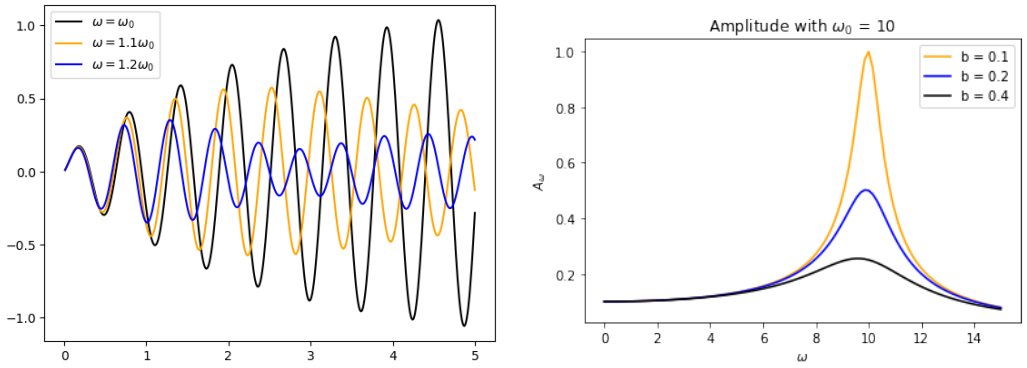
$$A_\omega := A = \frac{F_0}{\sqrt{m^2\left(\omega_0^2 - \omega^2\right)^2 + (b\omega)^2}}.$$

As the driving frequency gets progressively close to the resonant or natural frequency, the amplitude of the oscillations achieve their maximum value. If the damping b is small, A_ω gets larger when $\omega \approx \omega_0$, as we can see below in Figure 8(b). This amplitude can be seen in the numerical solutions. Fixing a small damping $b = 0.1$, Figure $8(a)$ shows the points of RK4 varying the value of ω. In the resonance case, the amplitude of the solutions $x(t)$ get larger over time. Out of resonance, the amplitudes are bounded.

5 Conclusions and future work

The 4-th order Runge-Kutta algorithm has been a suitable option for solving the Damped Harmonic Oscillator ODEs. In fact, this method has demonstrated to be highly accurate for adequate values of h, and it should come as no surprise, since the local truncation error is on the order of $\mathcal{O}(h^5)$.

For the simple harmonic case (i.e. $F_0 = 0$ and $b = 0$), the algorithm has needed a step-size h of the order of 10^{-3} for achieving remarkable results, compared to the analytical solution. When the value of h was large, the obtained results diverged from the ones that should obtain. As expected, the conservation of energy is numerically achieved for these small values of h.



(a) Numerical solutions near the resonance

(b) Amplitude

Figure 8: Figure (a) contains the points of RK4 near the resonance. That is, for a small value of b (0.1 in particular) and for different values of ω. Figure (b) contains the plot of A_ω for different values of b.

For the damped oscillatory cases, we have needed step-sizes of the order of 10^{-2}. The results for this cases have been outstanding, practically obtaining the same plot for the numerical solution as for the analytical one. For the overdamped and critically damped cases, the solution decays without oscillating. The critically damped solution decays faster than the overdamped one. For the underdamped case, the solution oscillates at real frequency $\frac{\sqrt{|b^2-4m^2\omega_0^2|}}{2m}$, but also decays exponentially in time at a rate proportional to the damping coefficient, in particular, proportional to $\frac{b}{2m}$.

For the last case, the Forced Damped Oscillator, step-sizes of the order of 10^{-2} have been enough to achieve great results. We have seen that, when fixing a small damping b, in the resonance case, the amplitudes of the solutions for different ω values get larger over time, while out of resonance, amplitudes are bounded.

The RK4 has proven to struggle a little (not so much for an adequate h) when there is a maximum or a minimum, since it is visible in the plots that the larger differences between RK4 and the analytical solutions occur there. It would be good to study in the future why this happens. We believe that this has happened due to the weights of the slopes k_1, k_2, k_3 and k_4, so it would be interesting to see if these differences also appear for a Runge-Kutta algorithm of lower or higher order.

References

[1] P. A. Dourmashkin, *Classical mechanics: Mit 8.01 course notes*. Wiley Custom Learning Solutions, 2014.

[2] M. DeCross, "Damped harmonic oscillators," [accessed 2022-12-15]. [Online]. Available: https://brilliant.org/wiki/damped-harmonic-oscillators/

[3] M. Fowler. Oscillations iii: Damped driven oscillator. [accessed 2022-11-04]. [Online]. Available: https://galileo.phys.virginia.edu/classes/152.mf1i.spring02/Oscillations4.htm

[4] R. Martínez Barchino and S. Cuadrado Gavilán, *Models amb equacions diferencials*, 2001-02. [Online]. Available: https://ddd.uab.cat/record/66103

[5] I. Berg, "Damped harmonic oscillator," [accessed 2022-11-29]. [Online]. Available: https://beltoforion.de/en/harmonic_oscillator/

[6] W. Kutta, "Beitrag zur naherungsweisen integration totaler differentialgleichungen," *Z. Math. Phys.*, vol. 46, pp. 435–453, 1901.

[7] C. Runge, "Über die numerische auflösung von differentialgleichungen," *Mathematische Annalen*, vol. 46, no. 2, pp. 167–178, 1895.

[8] D. M. Ritchie, S. C. Johnson, M. Lesk, B. Kernighan *et al.*, "The c programming language," *Bell Sys. Tech. J*, vol. 57, no. 6, pp. 1991–2019, 1978.

A Appendix

Below is the C [8] code with which all the results of the experiments have been obtained. The file's name is "RK4.c", and you can compile it typing:

```
gcc -Wall -o RK4 RK4.c -lm
```

```
./RK4
```

The code asks to type in the value of h, and returns 5 txt files.

1. The file "RK4-points_x.txt" contains the points $(t, x(t))$ of RK4.
2. The file "RK4-points_v.txt" contains the points $(t, v(t))$ of RK4.
3. The file "RK4-points_ES" contains the points $(t, \hat{x}(t))$ where now $\hat{x}(t)$ denotes the analytical solution at time t.
4. The file "RK4Errors.txt" contains the points $(t, \epsilon(t))$ where $\epsilon(t) = |\hat{x}(t) - x(t)|$.
5. The file "RK4-rel-errors.txt" contains the points $(t, \varepsilon(t))$ where $\varepsilon(t) = \left|\frac{\hat{x}(t)-x(t)}{\hat{x}}\right|$.

The first lines of the code define the values of the parameters. Users can change them manually for studying different cases of the problem.

```c
//code for RUNGA KUTTA 4-th ORDER
#include <stdio.h>
#include <stdlib.h>
#include<math.h>
#define PI 3.141592653589793
#define B 2.
#define Wo 10.
#define M 0.1
#define Fo 1
#define cte 0.4 // W is proportional to Wo, ( W = cte * Wo )
#define TOL 1e-10

double g(double v);
double f(double x, double v, double t);
double solExacta(double t);
void rk_4(double h,int N, double t[N], double x[N],double v[N], double tn);



int main(){
    FILE *fp_x;
    FILE *fp_v;
    FILE *fp_ES;
    FILE *fp2;
    FILE *fp3;
    fp_x=fopen("RK4-points_x.txt","w"); //we create the txt file that contains the points of x for the RK4 algorithm
    fp_v=fopen("RK4-points_v.txt","w"); //we create the txt file that contains the points of v for the RK4 algorithm
    fp_ES=fopen("RK4-points_ES.txt","w");//we create the txt file that contains the points of the analytical solution
    fp2=fopen("RK4Errors.txt","w"); //we create the txt file that contains the exact errors.
    fp3=fopen("RK4-rel-errors.txt","w"); //we create the txt file that contains the relative errors.
    printf("Introduce the value for h:\n");
    double h;
    scanf("%lf",&h);
    double tn=10.*PI/Wo;
```

```
    int N;
    double step=tn/h;
    N=(int)step+1;
    double t[N],x[N],v[N];
    x[0]=0.;
    v[0]=1.;
    t[0]=0.;


    rk_4(h,N,t,x,v,tn);
    int i;
    for(i=1;i<=(tn/h)+h;i++){//save points (t,x(t))
        fprintf(fp_x,"%.16G  \t %.16G \n",t[i],x[i]);
    }
    for(i=1;i<=(tn/h)+h;i++){//save points (t,v(t))
        fprintf(fp_v,"%.16G  \t %.16G \n",t[i],v[i]);
    }
    for(i=1;i<=(tn/h)+h;i++){//save points (t,solExacta(t))
        fprintf(fp_ES,"%.16G  \t %.16G \n",t[i],solExacta(t[i]));
    }
    for(i=1;i<=(tn/h)+h;i++){ //save errors
        fprintf(fp2,"%.16G  \t %.16G \n",t[i], fabs(solExacta(t[i])-x[
    i]));
    }
    for(i=1;i<=(tn/h)+h;i++){ //save relative errors
        fprintf(fp3,"%.16G  \t %.16G \n",t[i], fabs(solExacta(t[i])-x[
    i])/fabs(solExacta(t[i])));
    }
    fclose(fp_x);
    fclose(fp_v);
    fclose(fp_ES);
    fclose(fp2);
    fclose(fp3);
    return 0;
}


double g(double v){
    return v;
}

double f(double x, double v, double t){
    double w=cte*Wo;
    return -(B/M)*v-Wo*Wo*x+(Fo/M)*cos(w*t);
}

double solExacta(double t){
    if(B==0){ //simple harmonic oscillator
        return (1./Wo)*sin(Wo*t);

    }
    double alpha,theta;
    double w=cte*Wo;
    double inside_sqrt=B*B-4.*M*M*Wo*Wo;

    double inside_sqrt_2=M*M*(Wo*Wo-w*w)*(Wo*Wo-w*w)+B*B*w*w;
    if(inside_sqrt_2<=TOL){//should never enter here but just in case
        inside_sqrt_2=TOL;
    }
    alpha=Fo/sqrt(inside_sqrt_2);
    if(cte==1){ //case when w_0 = w
        theta=PI/2.; //because arctan(infty) = pi/2
    }
    else{
        theta=atan((B*w)/(M*(Wo*Wo-w*w)));
```

```
98      }
99
100         double c1,c2;
101     if(fabs(inside_sqrt)<=TOL){ //case when lambda1 and lambda2 are
        equal, critically damped case
102         double lambda;
103         lambda=-B/(2.*M);
104         c1=-alpha*cos(theta);
105         c2=alpha*lambda*cos(theta)-alpha*w*sin(theta)+1.;
106         return exp(lambda*t)*(c1+c2*t)+alpha*cos(w*t-theta);
107
108
109     }
110     else if (inside_sqrt>TOL){ //overdamped case
111         double lambda1,lambda2;
112         lambda1=(B+sqrt(inside_sqrt))/(2.*M);
113         lambda2=(B-sqrt(inside_sqrt))/(2.*M);
114         c1=(alpha*lambda2*cos(theta)+alpha*w*sin(theta)-1.)/(lambda1-
        lambda2);
115         c2=(alpha*lambda1*cos(theta)+alpha*w*sin(theta)-1.)/(lambda2-
        lambda1);
116
117         return c1*exp(-t*lambda1)+c2*exp(-t*lambda2)+alpha*cos(w*t-
        theta);
118
119     }
120     else{ //underdamped case
121         double real, imag;
122         real=-B/(2.*M);
123         imag=sqrt(fabs(B*B-4.*M*M*Wo*Wo))/(2.*M);
124         c1=-alpha*cos(theta);
125         c2=-(alpha*B*cos(theta)+2.*alpha*M*w*sin(theta)-2.*M)/(sqrt(
        fabs(B*B-4.*M*M*Wo*Wo)));
126         return exp(real*t)*(c1*cos(imag*t)+c2*sin(imag*t))+alpha*cos(w
        *t-theta);
127     }
128 }
129
130 void rk_4(double h,int N, double t[N], double x[N], double v[N],
        double tn){//runge-kutta
131     int i=0;
132     double k1,k2,k3,k4;
133     double l1,l2,l3,l4;
134     l1=h*f(x[0],v[0],t[0]);
135     k1=h*g(v[0]);
136     l2=h*f(x[0]+k1/2.,v[0]+l1/2.,t[0]+h/2.);
137     k2=h*g(v[0]+l1/2.);
138     l3=h*f(x[0]+k2/2.,v[0]+l2/2.,t[0]+h/2.);
139     k3=h*g(v[0]+l2/2.);
140     l4=h*f(x[0]+k3,v[0]+l3,t[0]+h);
141     k4=h*g(v[0]+k3);
142     double aux=0.;
143     while(t[i]<=tn+h){
144         i++;
145         aux++;
146         x[i]=x[i-1]+(k1+2*k2+2*k3+k4)/6.;
147         v[i]=v[i-1]+(l1+2*l2+2*l3+l4)/6.;
148         t[i]=t[0]+aux*h;
149         l1=h*f(x[i],v[i],t[i]);
150         k1=h*g(v[i]);
151         l2=h*f(x[i]+k1/2.,v[i]+l1/2.,t[i]+h/2.);
152         k2=h*g(v[i]+l1/2.);
153         l3=h*f(x[i]+k2/2.,v[i]+l2/2.,t[i]+h/2.);
154         k3=h*g(v[i]+l2/2.);
155         l4=h*f(x[i]+k3,v[i]+l3,t[i]+h);
```

```
156             k4=h*g(v[i]+k3);
157         }
158 }
```