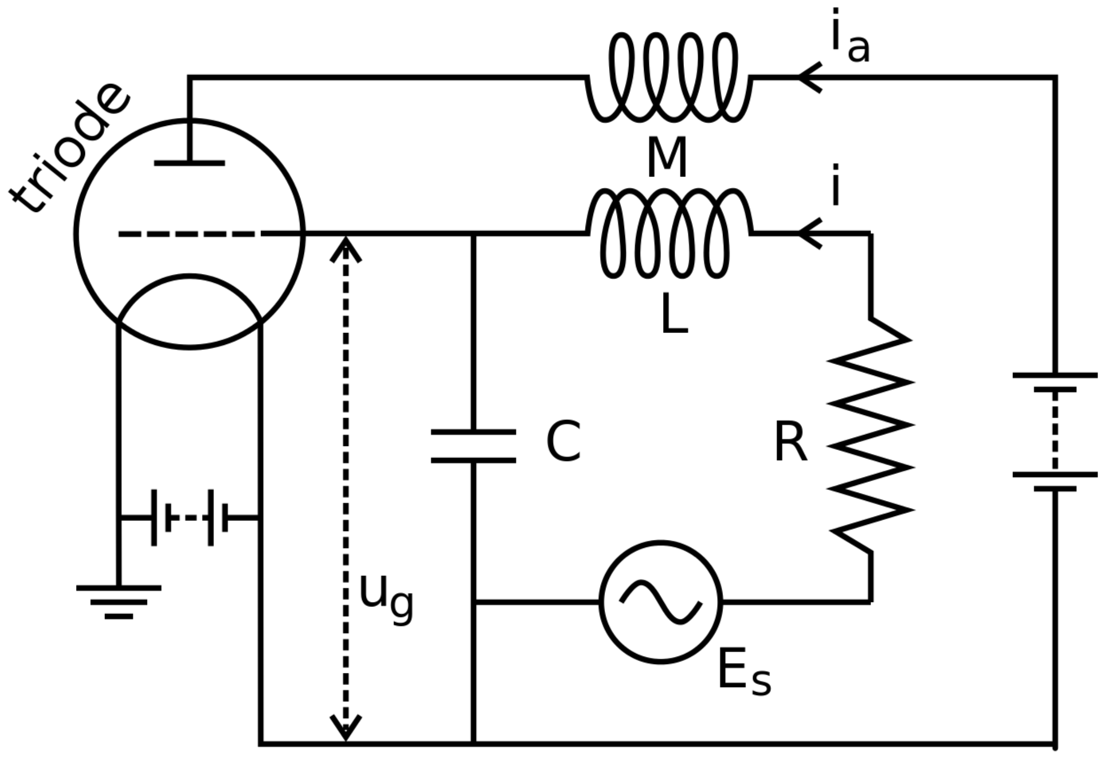
Integrated Master's Degree in Computational Engineering

1º Trabalho Prático de Avaliação Contínua

Non-Linear Oscillator

4th Runge-Kutta method and ODE45



COMPUTATIONAL PHYSICS

2019-2020

João Maria Machado, NMEC-89132, PL4

SUMMARY

In this report it was asked to analyse a non-linear oscillator using numerical methods, namely the Runge-Kutta method to determine various properties of the physical system, such as position over time, amplitude and period.

After explaining the code used to produce the solution, there will be an analysis over the obtained results, in order to verify that, in fact, **there is accordance in between the different methods used.**

INTRODUCTION

The studied physical system in question can be described by the following deferential equation:

$$\boldsymbol{m\frac{d^2y}{dt^2} + K(y + \alpha y^3) = \mu\left[\cos\left(\frac{dy}{dt}\right)\right]\frac{dy}{dt} + F_0\cos(w_0 t)} \quad (1)$$

The equation can be rearranged in order to make it easier to integrate:

$$\boldsymbol{\frac{d^2y}{dt^2} = \frac{\mu\left[\cos\left(\frac{dy}{dt}\right)\right]\frac{dy}{dt} + F_0\cos(w_0 t) - K(y + \alpha y^3)}{m}} \quad (2)$$

This ordinary differential equation can be integrated using the **Runge-Kutta** method. Firstly the problem will be solved using the ODE45 routine native to matlab, afterwards there will also be produced a solution using a 4^{th}order Runge-Kutta method. It is important to note that the ODE45 is a routine native to matlab that implements simultaneously a 4^{th} and 5^{th}order Runge-Kutta in order produce a method capable of adapting to functions that vary drastically in short intervals[1]

METHODS AND RESULTS

The first thing done in order to start writing the code was to write the equations that describe the physical system in a way that can be easily written into matlab. In the previous section there is already an equation (2) that describes the oscillator's acceleration. After there is only need to substitute the derivatives by their physical meaning. ($\boldsymbol{\frac{dy}{dt} = v,}$ and $\boldsymbol{\frac{dv}{dt} = \frac{d^2y}{dt^2}}$)

The system of first order differential equations is the following:

$$\begin{cases} \frac{dv}{dt} = \frac{\mu\left[\cos\left(\frac{dy}{dt}\right)\right]\frac{dy}{dt} + F_0\cos(w_0 t) - K(y + \alpha y^3)}{m} \\ \frac{dy}{dt} = v \end{cases} \quad (3)$$

After taking the necessary mathematical considerations the code can now be written.

The first method used is the ODE45, to implement this method two things must be defined. In first place we must define the tolerances (values of the absolute error and relative error) that this method will not exceed in absolute value, secondly the matlab function that contains information on **(3)** must also be defined. The tolerance settings can easily be noted after looking at the following snippet in the script **EX1B.m**:

```
reltol = 3*10^-14;
abstol_1=1*10^-13;
abstol_2=1*10^-13;
options = odeset('RelTol',reltol,'AbsTol',[abstol_1 abstol_2]);
```

Snippet 1: Tolerance settings for the Runge-Kutta Method.

Afterwards there was only one thing left to do, the integration. In order to integrate the function of derivatives **f** was defined and used in the main code.

```
Function dxdt = f(t,x,F0,w0,niu,alpha,K)
m=1;
dxdt = zeros(2, 1);
dxdt(1) = x(2);
dxdt(2) = (niu*cos(x(2))*x(2)+F0*cos(w0*t)-K*(x(1)+alpha*x(1)^3))/m;
…
[t,y] = ode45(@f,[ti tf],[y0 v0],options,F0,w0,niu,alpha,K);
```

Snippet 2: The function that contains the derivatives used for integration and the line in the main program where the function is summoned.

With that done we can now move on to studying the $\boldsymbol{\mu}$ parameter and how it effects the period and amplitude. In order to achieve this goal 2 things must be done: Integrate (like in the previous step) for the different values of

$\boldsymbol{\mu}$ and then computing a value of amplitude and period for each value of $\boldsymbol{\mu}$**.** This can be noted in the following snippet of script **EX1C.m**

```
for i=1:length(niu_vals)
    niu=niu_vals(i);
      . . .
count = 0;
    for n = 2:length(x) - 1
        if x(n - 1) <= x(n) && x(n) >= x(n + 1)  %maximos
            count = count + 1; %indices dos maximos
            ixd(count)=n;
        end
    end

          aux=lagr(t(ixd(1)-1:ixd(1)+1),x(ixd(1)-1:ixd(1)+1));
          x_max1 = aux(2);
          t_x_max1 = aux(1);
          aux=lagr(t(ixd(2)-1:ixd(2)+1),x(ixd(2)-1:ixd(2)+1));
          t_x_max2 = aux(1);
          T (i)= t_x_max2 - t_x_max1;
          A(i)=x_max1;
      end
```

Snippet 3: This code is responsible for firstly identifying all max values (and their indexes) of y and t then using the provided function lagr.m in order to find the most accurate values. Afterwards it uses the computed data in order to compute values of amplitude and period.

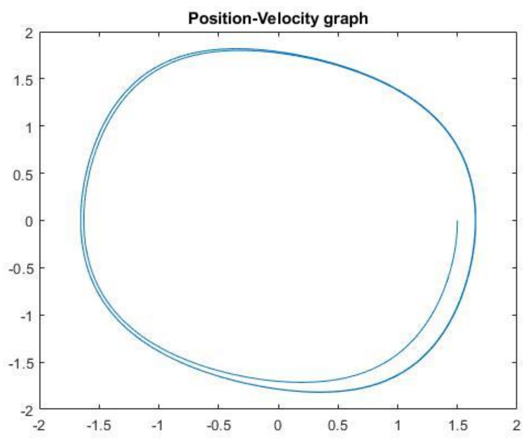
Finally, the last thing to do is to produce the code of the 4^{th} Runge-Kutta method. This task is rather trivial, when writing the Runge-Kutta method if implicit functions are implemented the coding process is always the same! This code can be observed in the file **EX1D.m**

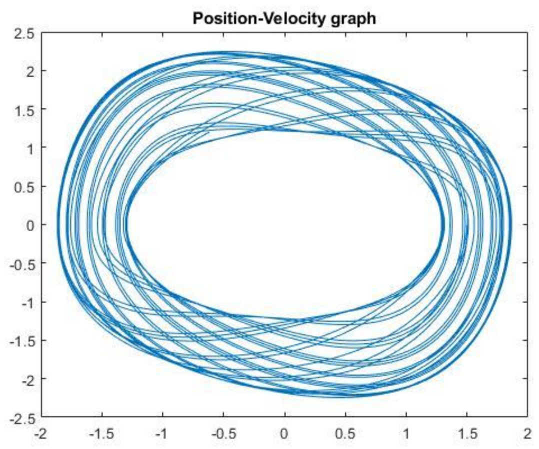
```
fy= @(v) v;

fv= @(y,v,t) (niu*cos(v)*v+F0*cos(w0*t)-K*(y+alpha*y^3))/m;
for i=1:length(t)-1
    r1v=fv(y(i),v(i),t(i));
    r1y=fy(v(i));
    r2v=fv(y(i)+r1y*(h/2),v(i)+r1y*(h/2),t(i));
    r2y=fy(v(i)+r1v*(h/2));
    r3v=fv(y(i)+r2y*(h/2),v(i)+r2v*(h/2),t(i));
    r3y=fy(v(i)+r2v*(h/2));
    r4v=fv(y(i)+r3y*h,v(i)+r3v*h,t(i));
    r4y=fy(v(i)+r3v*h);
    v(i+1)=v(i)+(h/6)*(r1v+2*r2v+2*r3v+r4v);
    y(i+1)=y(i)+(h/6)*(r1y+2*r2y+2*r3y+r4y);
    end
```

Snippet 4: Runge-Kutta Method. See annex for Butchers’ Table

DISCUSSION AND CONCLUSION

The final step of this report is to present the results and to discuss them. For the first part of the work we were able to obtain the following graphs.

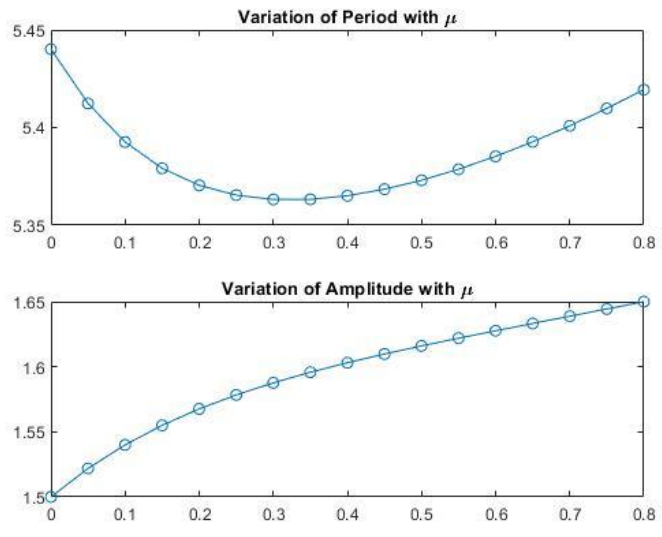




Graphs 1 (left) and 2 (right): On graph 1 we have w_0=1 and $F_0 = 0$ making the system a simple harmonic oscillator and on graph 2 we have w_0=2 and $F_0 = 0.8$, resembling a forced harmonic oscillator[2]

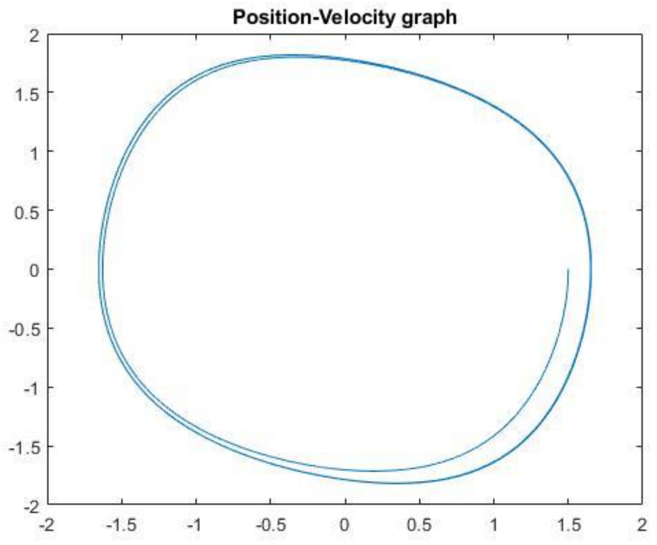
These results are very satisfactory because they are physically correct (with a global error of magnitude10^{-13}). The velocity goes from negative to positive values just as expected.

In respect to the $\boldsymbol{\mu}$ variation the following graphs were produced:



Graphs 3 (top) and 4 (bottom): We can observe that as $\boldsymbol{\mu}$ changes, both the period and the amplitude vary in an almost quadratic way. The results are plausible given that these quantities depend on speed and position.

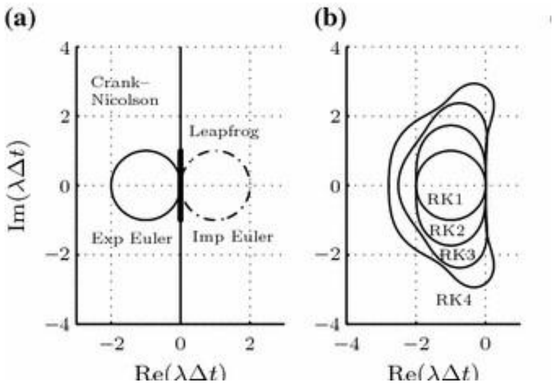
In regards to the 4^{th} order Runge-Kutta Method the following graph was obtained:



Graphs 5: 4^{th} order Runge-Kutta Method, one important thing to note is that the graph obtained in this part confirms the results produced previously (ODE45 analysis), still resembling the simple harmonic oscillator with damping.

One last thing to note is that both methods are stable for the given physical system however the method that ensures the minimal error is the ODE45 because it has the lowest tolerance for error when compared to the Runge-Kutta of fourth order (h^4, where h is the step of integration, and the h used is around 10^{-3} to avoid huge times of computation).

Finally when comparing with other method's this method is clearly superior. When compared to the Crank-Nicolson method, which is an acceptable method given that it usually conserves the energy of the physical system, the Runge-Kutta method still emerges as superior due to its efficiency and over all low error even though the regions of convergence are similar. If one were to look at the graph of the regions of convergence, the argument is clear:



Graphs 6: Regions of convergence of different numerical methods, taken from Kajishima T., Taira K. (2017) Finite-Difference Discretization of the Advection-Diffusion Equation. In: Computational Fluid Dynamics. Springer, Cham.

Anex 1

The butcher’s table used in the fourth order Runge-Kutta method is the following:

$$\begin{array}{c|cccc} 0 & & & & \\ \frac{1}{2} & \frac{1}{2} & & & \\ \frac{1}{2} & 0 & \frac{1}{2} & & \\ 1 & 0 & 0 & 1 & \\ \hline & \frac{1}{6} & \frac{1}{3} & \frac{1}{3} & \frac{1}{6} \end{array}$$

Anex 2

REFERENCES:

[1]- https://www.mathworks.com/help/matlab/ref/ode45.html

[2]- Serway, Raymond A., and Chris Vuille. College Physics. Volume 1. Volume 1. 2014.