Abstract

This report corresponds to the explanation of the code and evaluation of the results obtained in the third project of Simulation Methods. In order to solve the proposed questions, we have prepared a script in C, called *Continuation.c*, which is also attached in the delivery. We shall divide the report in two parts. The first one, will be destined to the explanation of the first two exercises (a computation of a periodic orbit near the origin) and, in the second one, we will study the last exercise, a continuation method.

1 Periodic orbits

We are asked to find a periodic orbit of period $T := 2\pi$ near the origin of the periodically perturbed pendulum,

$$\ddot{x} + \omega^2 \sin x = \epsilon \sin t$$

for $\omega = \sqrt{2}$ and $w = \frac{1}{\sqrt{2}}$ for a fixed value of ϵ, 10^{-2}. Let me explain how I have programmed this for a given value of ω. In order to find a periodic orbit, I will consider a temporal Poincaré section, i.e, $\Sigma = \{x \in \mathbb{R}^2 : t = 0 \ (\text{mod}\ T)\}$ and the Poincaré map, P, will be defined as the time T flow of the ODE. With this notation, it is easy to see that periodic orbits of period T appear as fixed points of P, $P(x^*) = x^*$. We can define $G(x) = P(x) - x$ and change the problem from finding a fixed point to finding a zero of a function. The idea is to apply the Newton's method to the function G and find a zero of the function. This is what has been implemented in the provided script, let me explain it.

The function *Newton_ex1_ex2* solve these two exercises. If the argument *choice* is set to 1, the function solves the problem for $\omega = \sqrt{2}$, otherwise, the function solves the problem for $w = \frac{1}{\sqrt{2}}$. The idea of the function is the following: given an initial point (x_0, y_0), compute an integration method to find $P(x_0, y_0)$ and perform a Newton method to finally found a fixed point. Let me explain this in detail.

Since we need the Jacobian of P to perform a Newton's method, we use variational equations, as explained in class to obtain it. Hence, by computing an integration method, we will obtain $P(x_0, y_0)$ and its derivative. The ODE to solve is

$$\begin{cases} \dot{x} = y \\ \dot{y} = -w^2 sin(x) + \epsilon sin(t) \\ \dot{v}_{11} = v_{21} \\ \dot{v}_{12} = v_{22} \\ \dot{v}_{21} = (-w^2 cos(x)) v_{11} \\ \dot{v}_{22} = (-w^2 cos(x)) v_{12} \\ (x(0), y(0)) = (x_0, y_0), V(0) = I \end{cases}$$

By solving this ODE, we obtain a solution $z \in \mathbb{R}^6$, where (z_1, z_2) is the image by the Poincaré map of (x_0, y_0) and the matrix $V = (z_{i,j})_{1 \leqslant i,j \leqslant 2}$ is the derivative of the Poincaré map at that point. Hence, I have created two functions called *ode_eq1* and *ode_eq2* that returns a pointer to the previous vector field with $\omega = \sqrt{2}$ and $w = \frac{1}{\sqrt{2}}$ respectively.

Back to the *Newton_ex1_ex2* function, the first thing I do, is to initialize the first two components of the solution vector to (x_0, y_0) and, to the identity matrix the next four components. Then,

I use the provided function *rkf45* to obtain the next point and the derivative V. Finally, we subtract the identity to V to obtain the Jacobain of G, J_G. At that point, we are ready to perform a Newton step. To compute the $k+1$ Newton step, we need to solve the following system in $\mathbf{x}$:

$$0 = G(\mathbf{x}^k) + J_G(\mathbf{x}^k)(\mathbf{x} - \mathbf{x}^k)$$

By calling $\mathbf{v} = \mathbf{x} - \mathbf{x}^k$, one can easily solve the system and the final result will be given by $\mathbf{v} + \mathbf{x}^k$. To solve the linear system, I have prepared a function called *gaussEliminationLS*, that solve the linear system using Gaussian elimination with partial pivoting.

To end this function, I have added code that, opens a file (*periodic_ex1.txt* if $w = \sqrt{2}$ and *periodic_ex2.txt* if $w = \frac{1}{\sqrt{2}}$) and prints all the points for the resulting fixed point of the Poincaré map using the integration method. We can observe in Figure 1 and 2, that, the resulting fixed points of the Poincaré map are actually periodic orbits! Finally, I have computed the eigenvalues of J_f, where f is the flow, at that fixed point to study the stability of the periodic orbit. As we have seen in class, to study the stability of a periodic orbit we need to study the ODE $\dot{x} = A(t)x$, where $A = J_f(\phi(t))$, being ϕ the periodic orbit. It can be shown that exist a change of variables such that the equation becomes $\dot{x} = Bx$ for a constant matrix B. Then, if $\text{Spec}(J_f) = \{\lambda_1, \lambda_2\}$, $\mu_j \in \text{Spec}(B) \iff \mu_j = \frac{1}{T}\log(|\lambda_j|)$. In this case, notice that in both cases ($\omega = \sqrt{2}$ and $\omega = \frac{1}{\sqrt{2}}$), the modulus of the eigenvalues are one, hence, we have two linearly stable periodic orbits. To compute the eigenvalues and calculate their modolus, we have prepared two little functions called *solver_complex* and *solver_real*, that, given the trace and the determinant of a 2×2 matrix, they compute the complex/real eigenvalues of the matrix. In our case, the matrix in which we want to compute its eigenvalues is V evaluated at the fixed point.

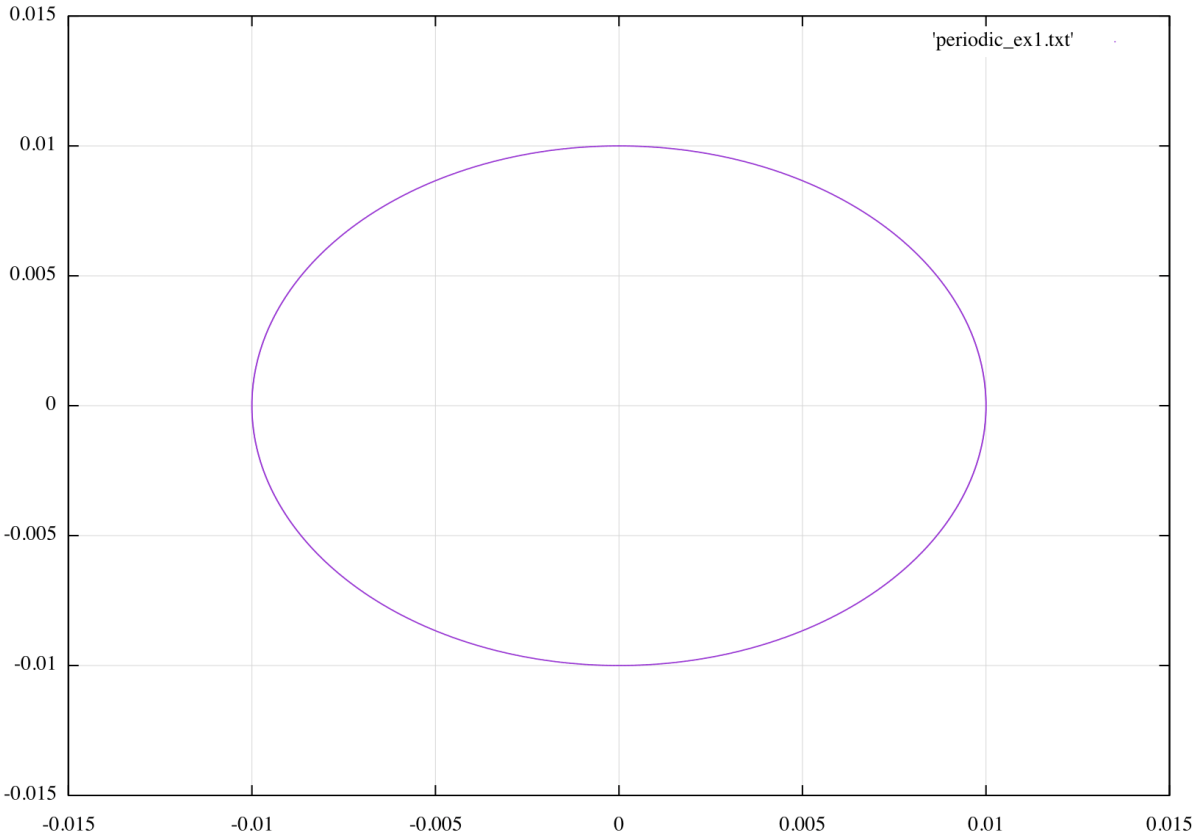

Figure 1: Periodic orbit for $\omega = \sqrt{2}$

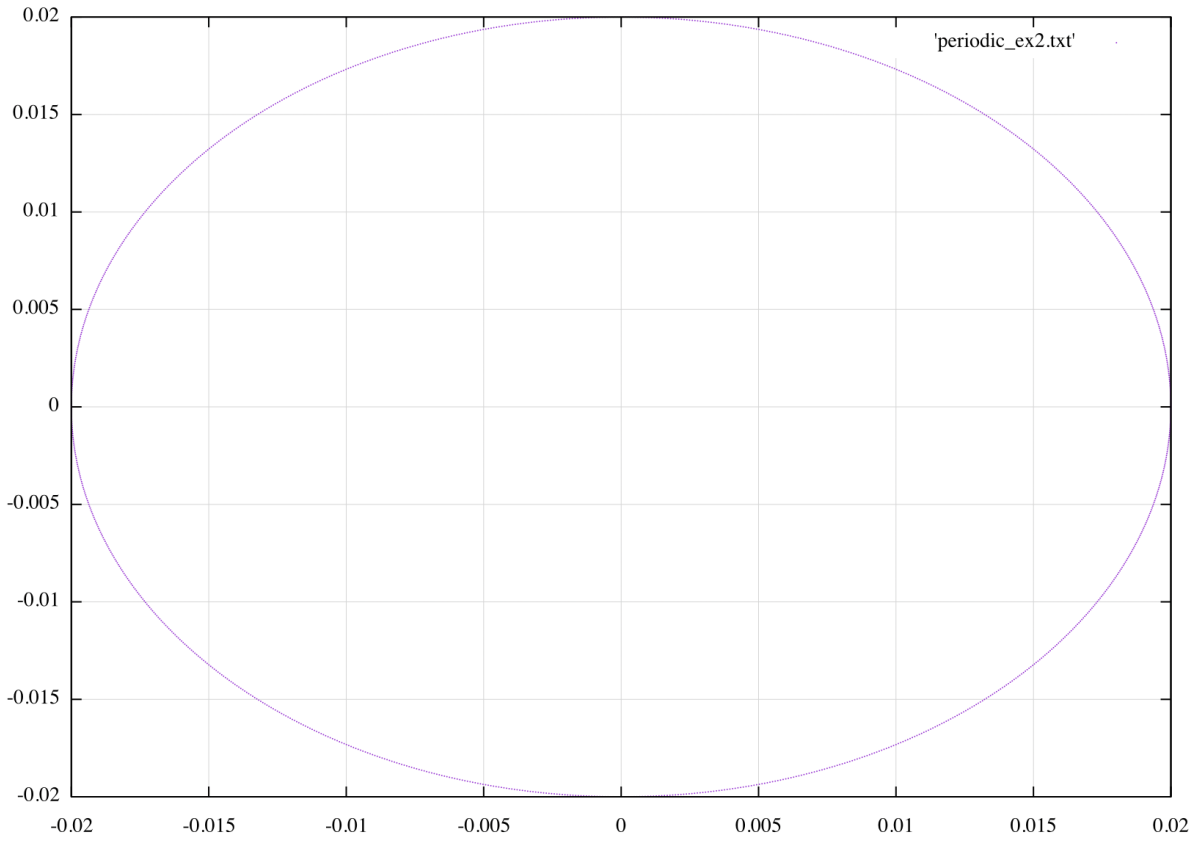

Figure 2: Periodic orbit for $w = \frac{1}{\sqrt{2}}$

2 Continuation method

Let us continue with the last part of the exercise. We are asked to perform a numerical continuation method with respect ω between $w = \frac{1}{\sqrt{2}}$ and $w = \sqrt{2}$, forwards and backwards. The function that implements this is called *continuation*. If the argument *choice* is set to 1, the function solves the forwards, otherwise, the function solves the backwards continuation. Let us explain the continuation method that we use to solve this exercise. Following the same notation as the previous exercise, we are interested in finding a zero of the function $G : \mathbb{R}^2 \to \mathbb{R}^2$, where $G(x, y) := (y - x, -\omega^2 sinx + \epsilon sint - y)$. At first, I tried to implement the easiest possible continuation method. The idea was the following: assume that we have a zero of G for a particular value of ω, then, we increase a little bit ω and we perform a Newton's method using as initial guess the previous point. Unfortunately, this does not work, near $\omega = 1$, the method starts doing many iterations, finding points that are not zeros of the function, leading to erroneous results.

Therefore, I decided to implement another continuation method, let us explain it. Instead of dealing with the ω as an external parameter, let us increase the dimension of the problem and add ω to an extra parameter of our function. Hence, we define now the function $G : \mathbb{R}^3 \to \mathbb{R}^2$ such that $G(x, y, \omega) = (y - x, -\omega^2 sinx + \epsilon sint - y)$. We are now interested in finding a zero of the following functions:

$$\begin{cases} G(z) = 0 \\ ||z - z_0||^2 - \delta^2 = 0 \end{cases}$$

Defining $H(z) := ||z - z_0||^2 - \delta^2 = 0$, and $F(z) := (G(z), H(z))$ we are interested in finding a zero of F using Newton's method. Notice that, for computing a Newton step, we need the

image of the Poincaré map at a point (x, y, ω) and the Jacobian of F at that point. Again, to compute the image of the Poincaré map, we use a numerical integration method, but now, notice that, since we have increase the dimension of the problem, the ODE to solve has changed:

$$\begin{cases} \dot{x} = y \\ \dot{y} = -\omega^2 sinx + \epsilon sint \\ \dot{\omega} = 0 \\ \dot{v}_{11} = v_{21} \\ \dot{v}_{12} = v_{22} \\ \dot{v}_{13} = v_{23} \\ \dot{v}_{21} = -\omega^2(cosx)v_{11} - 2\omega(sinx)v_{31} \\ \dot{v}_{22} = -\omega^2(cosx)v_{12} - 2\omega(sinx)v_{32} \\ \dot{v}_{23} = -\omega^2(cosx)v_{13} - 2\omega(sinx)v_{33} \\ \dot{v}_{31} = \dot{v}_{32} = \dot{v}_{33} = 0 \\ (x(0), y(0), \omega(0)) = (x_0, y_0, \omega_0), V(0) = I_2. \end{cases} \tag{2.1}$$

Using the provided integration method to solve the ODE, the first two components of the solution of the ODE are the image of the Poincaré map. Hence, since $G(z) = P(z) - z$ we can easily compute $G(z)$ and $H(z)$ is a quick calculation. At that point, we have computed the image of F, the function to which we want to compute a zero. To perform a Newton step, we need the Jacobian of F. Thanks to the ODE we have solved, we have "for free", the Jacobian of the Poincaré map (equivalently of G). Notice that, since the derivative of H is simple, the Jacobian of F has the following form:

$$J_F = \begin{bmatrix} v_{11} - 1 & v_{12} & v_{13} \\ v_{21} & v_{22} - 1 & v_{23} \\ 2(x - x_0) & 2(y - y_0) & 2(w - w_0) \end{bmatrix}$$

As before, we are ready to perform a Newton step by solving the following linear system:

$$0 = F(\mathbf{x}^k) + J_F(\mathbf{x}^k)(\mathbf{x} - \mathbf{x}^k). \tag{2.2}$$

Finally, let us explain which is the initial guess that we add to our Newton method. Assume that we have two zeros of the function F, $F(x_0, y_0, w_0) = F(z_0) = 0$ and $F(x_1, y_1, w_1) = F(z_1) = 0$. We can compute the vector $v_0 = \overrightarrow{z_0 z_1}/||\overrightarrow{z_0 z_1}||$ and set the initial guess of the Newton method to compute the next point to $z_0 + \delta v_0$. In order to keep track of δ and make it bigger if we are in front of a flat curve or make it small in the other case, we can observe the iterations of the Newton method. If the iterations are less than 10, we increase δ for the next step, otherwise, we decrease it. Notice that, in order to perform this method, we need two zeros of the function. Using the previous exercise, we already have one zero of the function (for $\omega = \sqrt{2}$ and $\omega = \frac{1}{\sqrt{2}}$), to find another, we increase or decrease a little bit ω (depending on if we are doing forwards or backwards continuation) and perform a "normal" Newton method, using as initial guess the previous point, in order to get two different points and start this method. This is all the theory needed to realize the code. Let us explain how the script works.

As already mentioned at the beginning of the chapter, the function *continuation* performs the continuation method forwards if *choice* is set to 1 or backwards otherwise. The first thing that this function does is to perform a "normal" step of Newton's method, increasing or decreasing a little bit ω, to find another point and start the continuation method as explained before. This is done in the function *Newton_ex3_firstStep*, that uses, as initial guess, the zero found in

$\omega = \sqrt{2}$ or $\omega = \frac{1}{\sqrt{2}}$, increasing or decreasing a little bit omega. Hence, we have defined two more vector fields to compute this step, *ode_eq_forwards* and *ode_eq_backrwards*, in which the ω is increased/decreased a little bit.

At this moment, we are ready to perform the mentioned continuation method. We can compute the vector v_0 and prepare the initial guess $z_0 + \delta v_0$ to our Newton method. We have implemented the function *Newton_ex3_continuation* that performs the continuation method mentioned before. We create the vector field (2.1) in the function *ode_eq3* to compute the numerical integration and solve the linear system (2.2) using Gaussian elimination with partial pivoting. Then, once we have the following point, we update the vector v and we compare it with the previous one, in order to check if we have to change the direction of the vector or not. Finally, in order to control the length of δ we check how many iteration the method has done, and we multiply by 1.5[1] or divide by 2 depending on if the method has done less or more than 10 iterations. Then, we are ready to compute the next initial guess to continue the algorithm.

If the reader performs this experiment, he or she will notice that this will not work. When we perform the forwards continuation method, δ starts becoming very small (10^{-7}) near $\omega = 1$ and the method makes very little progress. We have decided to add a parameter called *KB* that controls the number of KB that we will allow to print in the output file in which we are adding zeros of the function[2]. Once we reach 200 KB[3], we stop adding new points. The last ω found in the forwards continuation method is 0.905 and 1.09 in the backwards continuation. Again, we are stuck near $\omega = 1$. Nevertheless, now we can know the reason. In both cases, forwards and backwards, δ becomes very small, of the order of 10^{-7} and, as we have explained it before, this is because the curve we are computing has a big slope, thus, our method needs a really small δ to keep going. If we plot the points, using as axis x the omegas founds and using as axis y, the y founds[4], we can observe this slope we are talking about. See Figure 3:

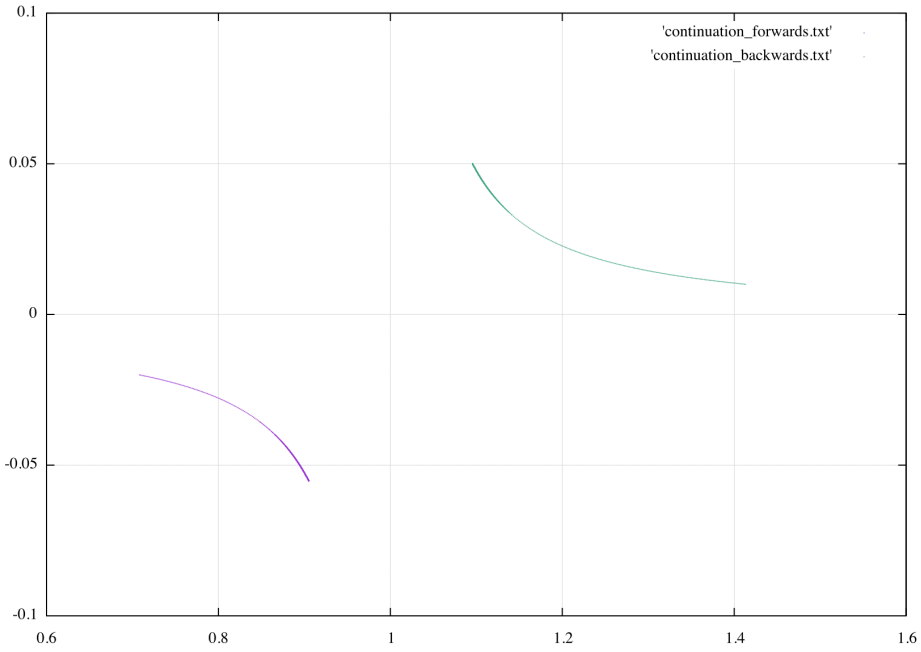

Figure 3: Periodic orbit for $w = \frac{1}{\sqrt{2}}$

[1] If delta is "too big", we do not multiply it and leave it as it is. Hence, we fix $\delta_{max} = 10^{-2}$.

[2] Two files are provided for the forwards and backwards continuation, *continuation_forwards.txt* and *continuation_backwards.txt*

[3] The reader can change this parameter in case the script takes a long time to compile

[4] x is always near 0, so we can bypass it.

In Figure 3, we can observe, in purple, the forward continuation and, in green, the backwards continuation. Notice that, the forwards continuation gets stuck near $\omega = 0.91$ as there seems to be some sort of "asymptote" in $\omega = 1$ that makes the continuation method goes to $-\infty$. In the same way, the backwards continuation gets stuck near $\omega = 1.1$ as there seems to be some sort of "asymptote" in $\omega = 1$ that makes the continuation method goes to ∞. It seems that there are no periodical orbits in $\omega = 1$ and makes the continuation method impossible between $\frac{1}{\sqrt{2}}$ and $\sqrt{2}$.

A How to compile

We are given a code with name *rkf45.c* that performs one step of Runge-Kutta-Fehlberg 4(5). To compile our code, it is necessary to compile both together, since our script uses the integration method implemented in *rkf45.c*. In order to do so, the reader needs to add the following lines to the terminal:

```
gcc Continuation.c rkf45.c -o executable.exe -Wall -lm
./executable.exe
```