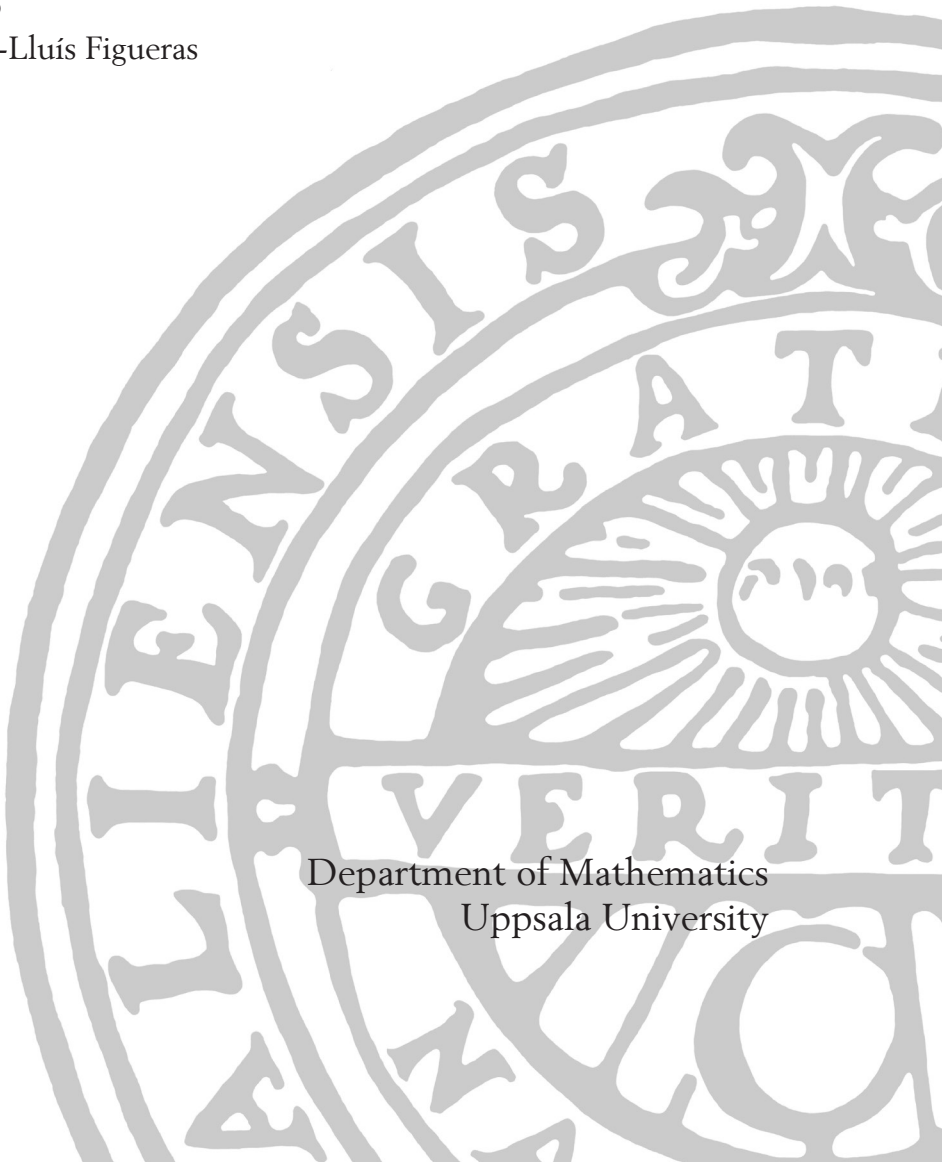
UPPSALA
UNIVERSITET

U.U.D.M. Project Report 2015:5

Analytical and Numerical Study of the Poincaré Map with Applications on the Computation of Periodic Orbits

Juan Carlos Albahaca

Examensarbete i matematik, 15 hp
Handledare och examinator: Jordi-Lluís Figueras
Maj 2015



Department of Mathematics
Uppsala University

Contents

Introduction

Dynamical system theory is an area of mathematics that studies models describing physical phenomena; events in nature such as population growth, environmental or financial forecasting, chemical reactions and a wide variety of other applications. The solutions of the models are represented in a state space as curves called trajectories that evolve over time, the states of flows are obtained in time steps t that can be calculated by numerical methods. The integration method used will be the 4th order Runge-Kutta since it is one of the most commonly used algorithms when very high accuracy is not required and we will be treating basic two-dimensional models. Depending on factors like linearity or non-linearity, there are different behaviors for the solutions, these flows of the vector fields can be drawn on a phase portrait.

A very useful tool to understand these behaviors is the Poincaré map which gives us a different way of analyzing the data. In this project we will first enunciate some important theorems about Dynamical system solutions and their dependence with parameters and initial conditions, then introduce the topic of Poincaré Maps to later give some examples and numerical calculations of periodic orbits. The goal of the project is to produce a program that recognizes every time a solution crosses a Poincaré Section, graph the Map and analyze the results for periodic orbits or limit cycles. This will be implemented on two-dimensional examples like the Lotka-Volterra "Predator vs. Pray" system and the Van der Pol system.

Chapter 1

Preliminaries

1.1 Existence and Uniqueness Theorem

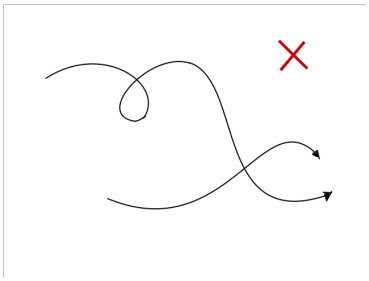
When you are solving an initial value problem of an ordinary differential equation you could ask yourself if this problem indeed has a solution and if this solution is unique, the next theorem gives certain properties that guarantee the problem has a unique solution, but this is not an if and only if situation which means that in the case of the problem not meeting these requirements there could still be a solution to the problem and moreover a unique one. Now we present a definition of the theorem taken from [1].

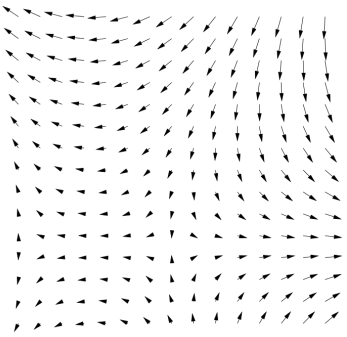
Consider the initial value problem:

$$\frac{dx}{dt} = f(t,x), \quad x(t_0) = x_0$$

Theorem 1.1.1. *Suppose that* $f(t,x)$ *and* $\frac{\partial f(t,x)}{\partial t}$ *are continuous functions defined on the region* $R = \{(t,x) : t_0 - \delta < t < t_0 + \delta, x_0 - \epsilon < x < x_0 + \epsilon\}$ *where* $\delta, \epsilon > 0$.
Then there exists a number δ_1 *such that the solution* $x = f(t)$ *exists and is the unique solution of the problem for* $t_0 - \delta_1 < t < t_0 + \delta_1$.

One important corollary of this theorem is that different trajectories never intersect, otherwise from any starting point there could be more than one solution, we know this is not possible from the uniqueness part of the theorem. This is why phase portraits have no crossing curves.





There is another important consequence of this theorem for two-dimensional phase spaces. Suppose there is a closed orbit C in the plane, then for any trajectory starting inside this orbit it cannot cross it, therefore it will stay confined. If there are no fixed points inside the orbit then the trajectory will eventually approach the curve C. This result is stated from the Poincaré-Bendixon theorem.

1.2 Dependence of solutions on parameters and initial conditions

The next theorems show that the solution's continuity and differentiability depend on the parameters (t, x_0, t_0).Taken from [10] "licoes de equacoes diferenciais ordinarias" by Sotomayor, Jorge.

Theorem 1.2.1 (Continuity Theorem)**.** *Suppose f is continuous in an open region Ω of $\mathbb{R}$ x $\mathbb{R}^n$ x Λ, for every $(t_0, x_0, \lambda) \in \Omega$ consider the IVP with a fixed λ:*

$$x' = f(t, x, \lambda), \;\; x(t_0) = x_0$$

with a unique solution $\varphi = \varphi(t, t_0, x_0, \lambda)$ defined in an open interval $((t_0, x_0, \lambda) - \omega, (t_0, x_0, \lambda) + \omega)$ then

$$D = \{(t, t_0, x_0, \lambda) : (t_0, x_0, \lambda) \in \Omega, t \in ((t_0, x_0, \lambda) - \omega, (t_0, x_0, \lambda) + \omega)\}$$

is an open region in $\mathbb{R}$ x Ω and φ is continuous in D.

Theorem 1.2.2 (Differentiability Theorem)**.** *Suppose f is continuous in an open region Ω of $\mathbb{R} \times \mathbb{R}^n \times \Lambda$ with $D_2 f$ continuous in Ω. Then for a fixed λ a solution $\varphi = \varphi(t, t_0, x_0, \lambda)$ of*

$$x' = f(t, x, \lambda), \;\; x(t_0) = x_0$$

is unique and $D_3\varphi$ exists with respect to x_0.
Even more, the application $(t, t_0, x_0, \lambda) \to D_3\varphi(t, t_0, x_0, \lambda)$ is continuous in

$$D = \{(t, t_0, x_0, \lambda) : (t_0, x_0, \lambda) \in \Omega, t \in ((t_0, x_0, \lambda) - \omega, (t_0, x_0, \lambda) + \omega)\}$$

and

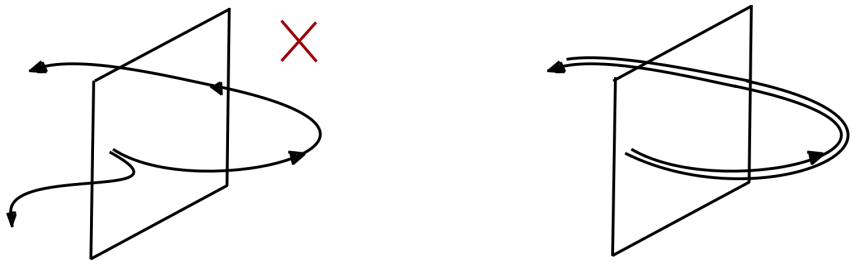
$$x(t) = D_3\varphi(t, t_0, x_0, \lambda).e_k = \frac{\partial \varphi}{\partial x_0^k}(t, t_0, x_0, \lambda)$$

for all $1 \leq k \leq dim\mathbb{R}^n$, is a solution of

$$x' = J(t)x, \quad x(t_0) = e_k$$

where $J(t) = J(t, t_0, x_0, \lambda) = D_2 f(t, \varphi(t, t_0, x_0, \lambda), \lambda)$

The main importance of this theorem is to show that for a neighborhood of the parameters chosen, the behavior of the trajectory will not differ after enough period of time.



The topological consequences of these theorems will have an impact for the Poincaré Maps defined later as the behavior of this will be dependent on these results.

Chapter 2

Poincaré Maps

Our main interest is to study dynamical systems that present periodic behavior, solutions such as spirals that can converge or diverge to closed orbits, or simply a continuous state space of closed curves. Sometimes we do not need to use the entire flow of the system to get the information we are interested in, we can find this information in a discrete system.

Poincaré Map is a discrete dynamical system that represents the continuous periodic flow of another system. [5]

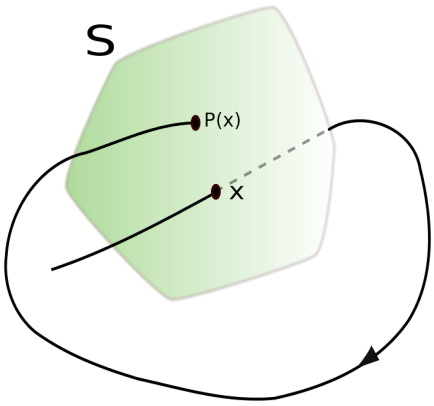


Figure taken from [6]
The next definition is taken from Strogatz book [11]. Consider an n-

dimensional system,

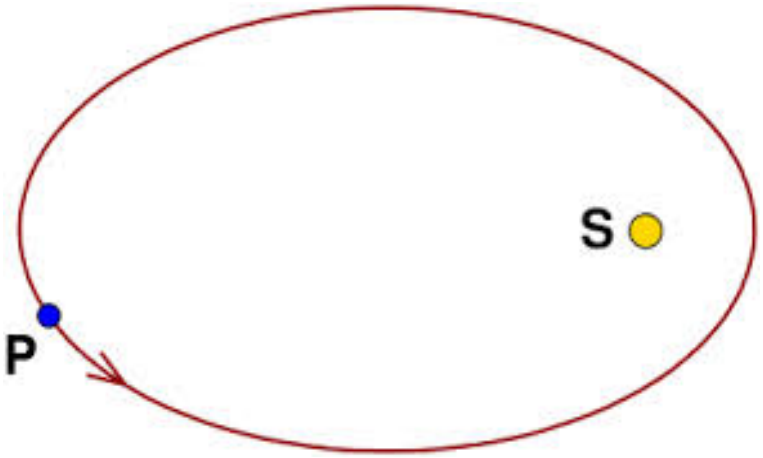
$$\dot{x} = f(x)$$

Let S be a $n-1$ dimensional surface of section. This surface must be transverse to the flow, meaning that all trajectories must go through it and not flow parallel to it. The Poincaré map is a mapping that goes from $S \to S$, this is obtained by taking every intersection from the trajectories one after the other. We will denote x_k as the kth intersection and define the Poincaré map as

$$x_{k+1} = P(x_k)$$

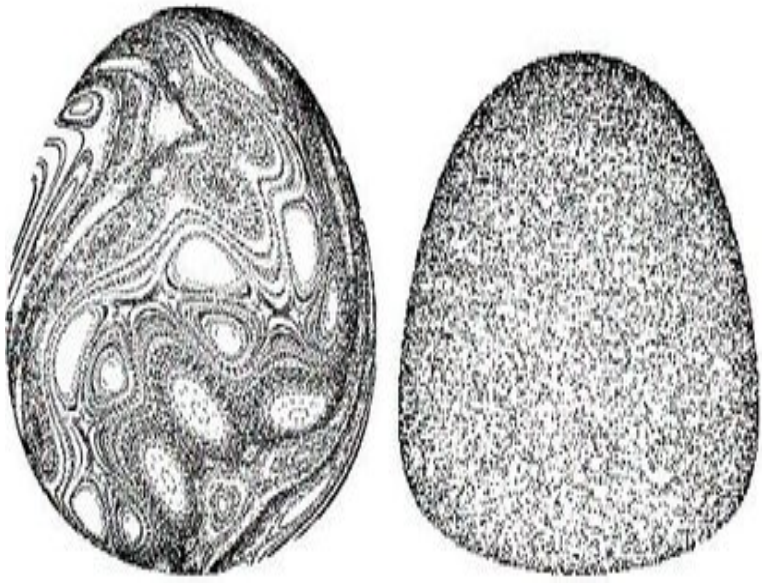
If $x* = P(x*)$ is a fixed point in the map, it means that a trajectory starting at this point comes back after some time T, and this is a closed orbit for the original system. The map P will contain information about the stability of closed orbits near the fixed points.

A classical example of the use of a Poincaré Map can be for analyzing planetary orbits. Suppose we have a planet that completes an orbit around a sun every year but slightly varies the orbit each time. In the long run we can have a better understanding of how this change evolves by using a Poincaré section and studying this mapping, instead of analyzing the complete flow. In this example we can clearly see the advantage that is taking a dimension off the problem.[7]



Taken from [9]

In any case the behavior of the original system will be adopted by the Poincaré Map, either if it is chaotic or an n-period orbit. chaotic maps can be very interesting and appealing to the eyes.



Poincaré sections of a passive tracer which visualize two different patterns; left: combination of many islands embedded in chaotic areas, right: a chaotic sea. Taken from [3]

Chapter 3

Numerical Considerations

3.1 Integration Method

The integration method used to solve the systems in this investigation is the Runge-Kutta 4. The fourth order Runge-Kutta method is an iterative method used for solving ordinary differential equations taking a step h on each iteration. This method was developed by the German mathematicians Carl Runge and Martin Wilhelm Kutta. It is familiar to the Euler Method,

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

,

But on each step the new y is determined by the previous y plus the weighted average of four increments, these increments are the product of the step size h times a slope specified by the right hand side function f of the differential equation.
The formulas for the increments and the new values of the variables x and y for the system taken from [2] are,

$$\frac{dx}{dt} = F(t, x, y), \quad \frac{dy}{dt} = G(t, x, y)$$

$$K_1 = F(t_n, x_n, y_n)h$$

$$L_1 = G(t_n, x_n, y_n)h$$
$$K_2 = F(t_n + \frac{h}{2}, x_n + \frac{K_1}{2}, y_n + \frac{L_1}{2})h$$
$$L_2 = G(t_n + \frac{h}{2}, x_n + \frac{K_1}{2}, y_n + \frac{L_1}{2})h$$
$$K_3 = F(t_n + \frac{h}{2}, x_n + \frac{K_2}{2}, y_n + \frac{L_2}{2})h$$
$$L_3 = G(t_n + \frac{h}{2}, x_n + \frac{K_2}{2}, y_n + \frac{L_2}{2})h$$
$$K_4 = F(t_n + h, x_n + K_3, y_n + L_3)h$$
$$L_4 = G(t_n + h, x_n + K_3, y_n + L_3)h$$

$$x_{n+1} = x_n + \frac{K_1 + 2K_2 + 2K_3 + K_4}{6}$$
$$y_{n+1} = y_n + \frac{L_1 + 2L_2 + 2L_3 + L_4}{6}$$

Scientists today consider RK4 to be very efficient and accurate for the integration of ordinary differential equations in comparison to others like the midpoint method, but if a very high accuracy is required there are other methods more suitable as the Bulirsch-Stoer. For our purposes in this investigation the Runge-Kutta method works well since we will use it to find the step where the trajectories of solutions go through a Poincaré section, from here we will use a bisection scheme to approach the point on the map.

3.2 Implementation

The technical computing language Matlab has been used to create a program that simulates a Poincaré section . Joining the integration method Runge-Kutta 4 with a bisection scheme we have been able to reproduce the concept of a Poincaré Map in a two dimensional system. The idea of the program is to take the section as a vertical line parallel to the y axis, then after obtaining a sequence of points with the RK4 we check whenever the line has been

crossed and start to bisect the step length used on the integration method until we reach a tolerance of `2e-10`, if the line has not been crossed in the the specific time flow then a message is shown that the flow does not cut the section. The inputs for this function are: "y" the vector field defined in another script,"tspan" a vector of two components representing the initial and final values of the desired time flow, "y0" as the initial values,"h" is the step size and "x" the Poincaré section or vertical line equal to a constant.

After this first idea we obtain a set of points whenever the line is crossed but this will only be useful for further analysis depending on the behavior of these points as the theory of the Poincaré Maps dictates.

The first thing to clarify is that there will always be an error to the estimations as it is impossible to have an exact precision, therefore we cannot find two equal points, but this does not stop us from finding clear results like a periodic orbit or convergence to a limit cycle.

One of the main uses of a Poincaré Section is to find closed orbits and for this we must find points that will map to themselves. It is our interest to find these points and test the program on vector fields such as the Lotka-Volterra and Van der Pol system.

After creating the program that obtains the points of the Poincaré Section, the next idea is to be able to obtain the n^{th} step of a Poincaré crossing and then graph the Poincaré map function $f(x) : \Sigma \to \Sigma$ with the points obtained.

With this graph we can compare to the function $y = x$ and the point where they meet will be a closed orbit. For this a new Matlab function was created where we find the zeros of the function $g(x) = f(x) - x$. In this case we use a bisection scheme once again where there is a change of signs in the function g until we reach a tolerance of `2e-10`.

Chapter 4

Computing Periodic Orbits via Poincaré Maps

On this section we will implement the Matlab functions on two-dimensional examples computing their periodic orbits with the use of a Poincaré Section $\Sigma = \{x \in \mathbb{R}^2 : x_1 = C\}$.

4.1 Lotka-Volterra Model

The Lotka-Volterra model also known as the Predator-Prey model was proposed by Alfred J. Lotka in 1910. It represents the behavior or interaction between a predator and its pray. Either the predator provokes the extinction of his prey or the prey will manage to survive in time, depending on how the model is constructed there are different scenarios. In this particular case, only the interaction between the two is involved in the model.

Now we introduce the equations:

$$x' = ax - bxy$$
$$y' = -ry + cxy$$

In this model the prey population at time t is represented by x and the predator population by y. "a" represents the growth rate of the prey and "b" the rate at which the predator eats the pray. "r" represents the death rate of the predator and "c" represents the rate of increase of the predator.

From the first equation we see that without the existence of a predator, the prey would grow exponentially $x' = ax$ and from the second equation, without a prey the predator would die exponentially $y' = ry$. We can also see the interaction term on the first equation $-bxy$ means that every time the predator interacts with a prey the later one decreases. From the second equation, the interaction term $+cxy$ represents an increase on the predator when this happens.[4]

To solve the system we multiply the first equation by $\frac{y'}{xy}$ and the second equation by $\frac{x'}{xy}$, then by subtracting both equations we obtain.

$$y'(\frac{a}{y} - b) + x'(\frac{r}{x} - c) = 0$$

Integrating both sides,

$$\int y' \frac{a}{y} dy - \int by' dy = -\int x' \frac{r}{x} dx + \int cx dx$$

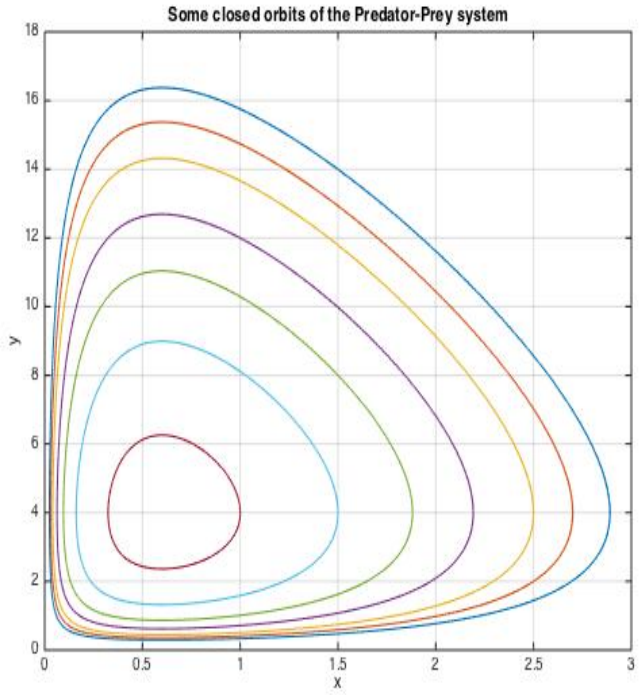
we obtain

$$r \log x - cx + a \log y - by = C$$

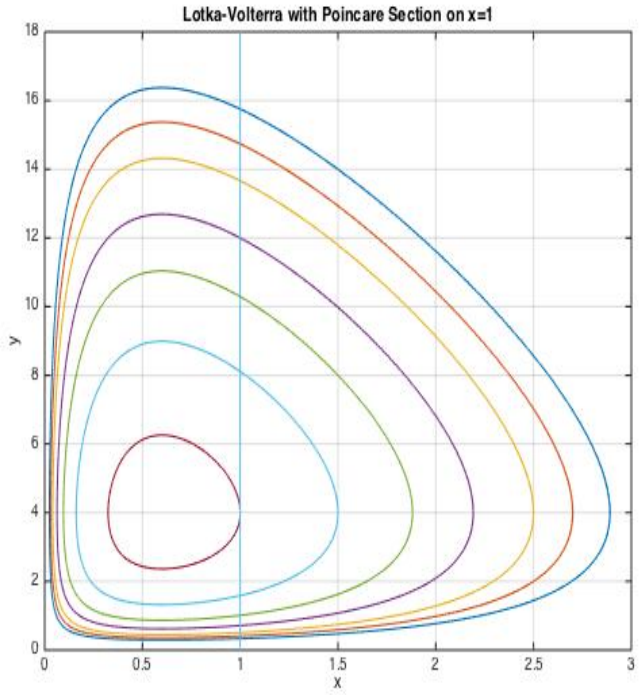
The solutions are the level curves of the function,

$$F(x, y) = |x|^r |y|^a e^{-(cx+by)}$$

All solutions on the first quadrant are closed. Here are some of them shown in the phase portrait taking $a = 0.4, b = 0.1, r = 0.3, c = 0.5$. Some ideas for this example were taken from Warwick Tucker's article for computing Poincaré Maps [12].



Now we want to know the behavior of the dynamical system with the use of a Poincaré section on $x = 1$.



We run the PoincareGraph program with $x = 1$ and y from 0.1 to 18, time flow from 0 to 1000 and step size $h = 0.1$ and we obtain the next result,

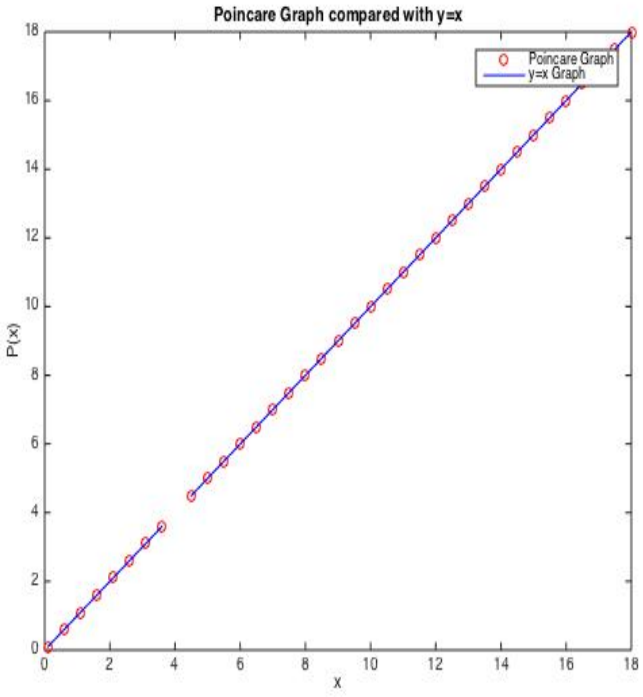
>> y=PoincareGraph(lv,[0.1:0.5:18],1,2,[0 1000],0.1)

The vector field does not intersect the Poincare section 2 times with the given initial value [1 4] and time flow 0 1000.

The program detects that the Poincaré Map is not well defined at the point $(1, 4)$, so we run the program again avoiding $y = 4$,

>> y=PoincareGraph(lv,[0.1:0.5:3.9],1,2,[0 1000],0.1) >> hold on >> y=PoincareGraph(lv,[4.5:0.5:18],1,2,[0 1000],0.1)

Obtaining the graph,



What this graph tells us is that the solutions of the Lotka-Volterra system are closed curves and the Poincaré Map is not well defined when $y = 4$ since the flow is tangent to the section in this point.

4.2 Van der Pol Oscillator

The Van der Pol Oscillator is a model that describes an oscillatory process with non-linear damping. It was discovered in the 1920s by Balthasar Van der Pol, a physicist and mathematician born in the Netherlands. The oscillations of the system converge towards a limit cycle so we will implement the Poincaré Map to find this result. A limit cycle is an isolated closed trajectory, this means that for a neighborhood S around this trajectory, all trajectories Γ_z for all initial values $z \in S$ are not closed. Our interest in this model is to investigate the behavior of the trajectories and their convergence.

The equation of the model is,

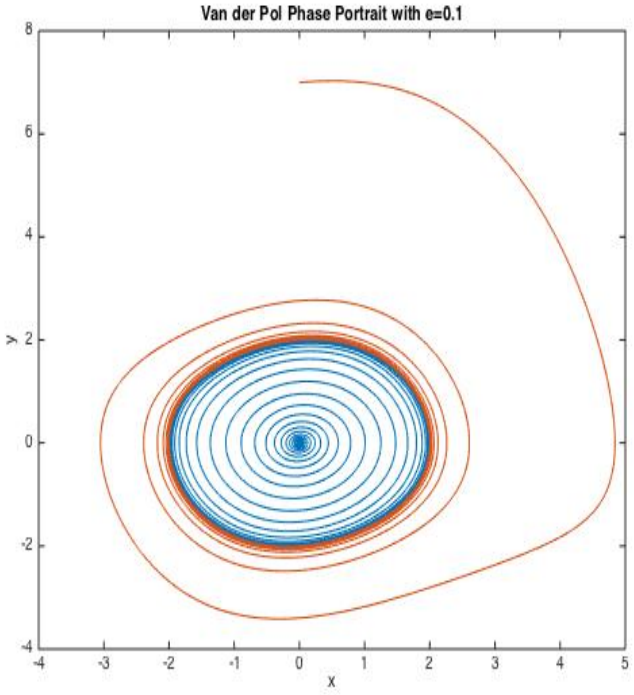
$$\ddot{x} - \epsilon(1 - x^2)\dot{x} + x = 0$$

We can rewrite it as,

$$\begin{cases} \dot{x} &= y \\ \dot{y} &= \epsilon(1 - x^2)y - x \end{cases}$$

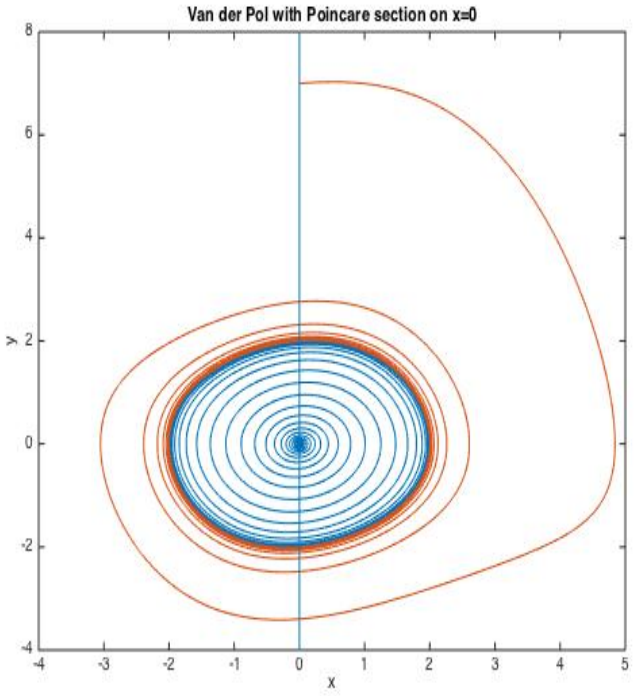
By Lienard's theorem the system has exactly one limit cycle, if $0 < \epsilon < 1$ then this limit cycle lies within a neighborhood of distance $O(\epsilon)$ from the circle centered at the origin with radius 2. The proof for the theorems can be found in Wesley Cao's paper [8].

Setting $\epsilon = 0.1$ we obtain the phase portrait.



In the phase portrait we can see how the two flows converge to the limit cycle centered at the origin and with radius ≈ 2 as we expected.

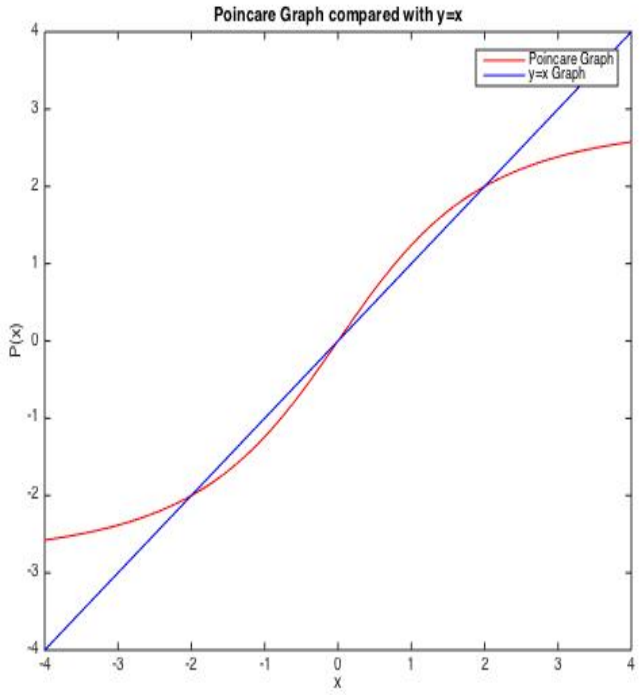
Now we will use a Poincaré Section on $x = 0$ to understand the behavior of the system from a different perspective.



We run the program PoincareGraph on the system with $x = 0$ and y values from -4 to 4, number of steps equals 2 so we return to the initial point and see if it converges, time flow from 0 to 100, and step size $h = 0.01$.

```
>> y=PoincareGraph(vdp,[-4:0.1:4],0,2,[0 100],0.01)
```

We obtain the next graph,



With the PoincareGraphBisec program we can find the points that intersect with $y = x$, this will represent either fixed points or periodic orbits.

>> y=PoincareGraphBisec(vdp,-4,-1,0,2,[0 100],0.01)

$y = -2.0018$

>> y=PoincareGraphBisec(vdp,1,4,0,2,[0 100],0.01)

$y = 2.0018$

>> y=PoincareGraphBisec(vdp,-1,1,0,2,[0 100],0.01)

$y = 1.0000$

From these results we can interpret that zero is an unstable fixed point and and there is a limit cycle of radius 2.0018 centered at zero.

Chapter 5

Conclusions and further work

Dynamical systems theory is a very important branch of Mathematics in which it is possible to model a wide variety of physical phenomena or abstract ideas we have not been able to understand. With the help of computers today we can predict many important things, not only in advanced topics in science but also in everyday life.

There are various ways of analyzing the solutions of the models and as we found in this investigation, the Poincaré Maps is a useful tool when it comes to understanding the behavior of periodic orbits and finding accurate approximations for limit cycles in two dimensional systems, it reduces the problem by one dimension and provides a deeper understanding. For three dimensional systems there is a whole new perspective to what this theory brings, by setting up plane sections to intersect flows in space you can visualize from within the changes in time.

Summary of contributions

In the project some useful programs were produced to represent Poincare Maps, they can be used for obtaining the points that cross a Poincare Section, to find periodic behaviors, to graph the mappings and to obtain closed orbits or limit cycles. By varying the inputs of the functions you can obtain more precise and faster results. Of course if there is a clear idea of the behavior of the solution, you might have a good initial guess of a Poincaré section that will provide interesting results, sometimes you already know the overall

behavior but need some information in a specific period of the flow. In any case there are many uses for them and they show a good performance for the kind of examples provided in the project.

Further work

It is of my interest to continue studying Dynamical System theory and develop the idea of Poincaré maps for three dimensional systems, producing a program for this and to make numerical analysis on important examples found in modern science. Mathematical modeling is a vast world we are working to unveil and with the help of this tool some new discoveries might be at grasp.

Bibliography

[1] Department of Mathematics, University of Illinois - Existence and Uniqueness theorem for First Order ODE's. http://http://www.math.uiuc.edu/~tyson/existence.pdf. Accessed: 2015-05-01.

[2] Department of Physics, Davidson College - Runge-Kutta method for solving two coupled first order differential equations or one second order differential equation. http://www.phy.davidson.edu/FacHome/dmb/py200/RungeKuttaMethod.htm. Accessed: 2015-04-26.

[3] Eindhoven University of Technology- Tracers take the tube: mixing in 3D viscous flows. http://www.tue.nl/en/university/departments/applied-physics/research/transport-physics/turbulence-and-vortex-dynamics/viscous-flow-and-mixing/tracers-take-the-tube/. Accessed: 2015-05-19.

[4] Lotka-Volterra model. http://en.wikipedia.org/wiki/Lotka

[5] Poincare Map, Advanced concept team, ESA. http://www.esa.int/gsp/ACT/mad/projects/inter_moon.html. Accessed: 2015-05-18.

[6] Poincare Map, Wikipedia. http://en.wikipedia.org/wiki/Poincar%C3%A9_map#/media/File:Poincare_map.svg. Accessed: 2015-05-18.

[7] Poincaré Maps and interpretation. http://physics.stackexchange.com/questions/141493/poincar%C3%A9-maps-and-interpretation. Accessed: 2015-05-09.

[8] Wesley Cao, Cornell University - Van der Pol Oscillator. http://www.math.cornell.edu/~templier/junior/final_paper/Wesley_Cao-vanderpol.pdf. Accessed: 2015-05-01.

[9] What figure skaters, orbiting planets and neutron stars have in common. www.einstein-online.info.

[10] Jorge Sotomayor. lições di equações diferenciais ordinárias. Livros Técnicos e Cientificos Editors S.A., Rio de Janeiro, Brasil, 1979.

[11] Steven H. Strogatz. Nonlinear dynamics and chaos: With applications to physics, biology, chemistry and engineering. Perseus Books Publishing, Reading, Massachusetts, 1994.

[12] Warwick Tucker. Computing accurate poincaré maps. *Physica D*, 2002.

Chapter 6

Appendix

6.1 Matlab codes

6.1.1 Poincaré function

```
function PM = Poincare(y,tspan,y0,h,x)
if nargin < 5, error('at least 5 input arguments required'), end
n = length(tspan);
ti = tspan(1); tf = tspan(n);
tp = ti:h:tf;
s = length(tp);
yp = zeros(s,3);
yp(1,2:3)= [y0(1) y0(2)];
yp(:,1)=tp;
poin=zeros(s,2);
z=zeros(s,2);
ff=1;
m=1;  %counter of yp matrix
cc=1; %counter of kl matrix
kl=zeros(s,10);
%To make the program faster we store the values of the ks and ls in a matrix
for n = ti:h:tf
  kl(cc,1) = y{1}(n,[yp(m,2) yp(m,3)])*h;
  kl(cc,2) = y{2}(n,[yp(m,2) yp(m,3)])*h;
  kl(cc,3) = y{1}(n+(h/2),[yp(m,2)+(kl(cc,1)/2) yp(m,3)+(kl(cc,2)/2)])*h;
  kl(cc,4) = y{2}(n+(h/2),[yp(m,2)+(kl(cc,1)/2) yp(m,3)+(kl(cc,2)/2)])*h;
  kl(cc,5) = y{1}(n+(h/2),[yp(m,2)+(kl(cc,3)/2) yp(m,3)+(kl(cc,4)/2)])*h;
```

```
  kl(cc,6) = y{2}(n+(h/2),[yp(m,2)+(kl(cc,3)/2) yp(m,3)+(kl(cc,4)/2)])*h;
  kl(cc,7) = y{1}(n+(h/2),[yp(m,2)+kl(cc,5) yp(m,3)+kl(cc,6)])*h;
  kl(cc,8) = y{2}(n+(h/2),[yp(m,2)+kl(cc,5) yp(m,3)+kl(cc,6)])*h;
  kl(cc,9) = yp(m,2) + ((1/6)*(kl(cc,1) + (2*kl(cc,3)) + (2*kl(cc,5)) + kl(cc,7)));
  kl(cc,10) = yp(m,3) + ((1/6)*(kl(cc,2) + (2*kl(cc,4)) + (2*kl(cc,6)) + kl(cc,8)));
    m=m+1;
    yp(m,2:3)= [kl(cc,9) kl(cc,10)]; %yp(m,2:3)= [p q];


    %bisection method

    if (m>10) && (kl(cc,9)==x)
        poin(ff,1:2) = [yp(m,2) yp(m,3)];
        ff=ff+1;
    end

    if ((m>10) && (x<yp(m,2)) &&  (x > yp(m-1,2))) || ((m>10) && (x>yp(m,2)) &&
(x<yp(m-1,2)))
    d=n;
    s=h;
     v=m-1; z=d-s;

    r=yp(v,2); w=yp(v,3); g=yp(v+1,2);

    while (abs(r-x)>2e-10)
    if ((x < r) &&  (x > g)) || ((x > r) &&  (x < g))

        s=s/2;

            z=z+s;
            k1 = y{1}(z,[r w])*s;
    l1 = y{2}(z,[r w])*s;
    k2 = y{1}(z+(s/2),[r+(k1/2) w+(l1/2)])*s;
    l2 = y{2}(z+(s/2),[r+(k1/2) w+(l1/2)])*s;
    k3 = y{1}(z+(s/2),[r+(k2/2) w+(l2/2)])*s;
    l3 = y{2}(z+(s/2),[r+(k2/2) w+(l2/2)])*s;
    k4 = y{1}(z+(s/2),[r+k3 w+l3])*s;
    l4 = y{2}(z+(s/2),[r+k3 w+l3])*s;
    uu = ((1/6)*(k1 + (2*k2) + (2*k3) + k4));
    pp = ((1/6)*(l1 + (2*l2) + (2*l3) + l4));
    r = r + ((1/6)*(k1 + (2*k2) + (2*k3) + k4));
    w = w + ((1/6)*(l1 + (2*l2) + (2*l3) + l4));


    else
```

```
            r=r-uu;
            w=w-pp;
            z=z-s;


    s=s/2;
    z=z+s;

    k1 = y{1}(z,[r w])*s;
    l1 = y{2}(z,[r w])*s;
    k2 = y{1}(z+(s/2),[r+(k1/2) w+(l1/2)])*s;
    l2 = y{2}(z+(s/2),[r+(k1/2) w+(l1/2)])*s;
    k3 = y{1}(z+(s/2),[r+(k2/2) w+(l2/2)])*s;
    l3 = y{2}(z+(s/2),[r+(k2/2) w+(l2/2)])*s;
    k4 = y{1}(z+(s/2),[r+k3 w+l3])*s;
    l4 = y{2}(z+(s/2),[r+k3 w+l3])*s;
    uu = ((1/6)*(k1 + (2*k2) + (2*k3) + k4));
    pp = ((1/6)*(l1 + (2*l2) + (2*l3) + l4));
    r = r + ((1/6)*(k1 + (2*k2) + (2*k3) + k4));
    w = w + ((1/6)*(l1 + (2*l2) + (2*l3) + l4));


    end
    end
    poin(ff,1:2) = [r w];
    ff=ff+1;
    end
    cc=cc+1;
end
    PM = poin(1:2:ff-1,1:2);
    if poin==z
    disp(['The vector field does not intersect the Poincare',...
    ' section with the given initial values and time flow']);
    PM=[];
    else
    PM = poin(1:2:ff-1,1:2);
    end
```

6.1.2 Poincaré One Step

```
function PM = PoincareStep(y,tspan,y0,h,x)
if nargin < 5, error('at least 5 input arguments required'), end
```

```
ti = tspan(1); tf = tspan(2);
tp = ti:h:tf;
s = length(tp);
yp = zeros(s,3);
yp(1,2:3)= [y0(1) y0(2)];
yp(:,1)=tp;
m=1;  %counter for the yp matrix
cc=1; %counter of kl matrix
kl=zeros(s,10);
ff=1;
poin = [];
for n = ti:h:tf
  kl(cc,1) = y{1}(n,[yp(m,2) yp(m,3)])*h;
  kl(cc,2) = y{2}(n,[yp(m,2) yp(m,3)])*h;
  kl(cc,3) = y{1}(n+(h/2),[yp(m,2)+(kl(cc,1)/2) yp(m,3)+(kl(cc,2)/2)])*h;
  kl(cc,4) = y{2}(n+(h/2),[yp(m,2)+(kl(cc,1)/2) yp(m,3)+(kl(cc,2)/2)])*h;
  kl(cc,5) = y{1}(n+(h/2),[yp(m,2)+(kl(cc,3)/2) yp(m,3)+(kl(cc,4)/2)])*h;
  kl(cc,6) = y{2}(n+(h/2),[yp(m,2)+(kl(cc,3)/2) yp(m,3)+(kl(cc,4)/2)])*h;
  kl(cc,7) = y{1}(n+(h/2),[yp(m,2)+kl(cc,5) yp(m,3)+kl(cc,6)])*h;
  kl(cc,8) = y{2}(n+(h/2),[yp(m,2)+kl(cc,5) yp(m,3)+kl(cc,6)])*h;
  kl(cc,9) = yp(m,2) + ((1/6)*(kl(cc,1) + ...
  (2*kl(cc,3)) + (2*kl(cc,5)) + kl(cc,7)));
  kl(cc,10) = yp(m,3) + ((1/6)*(kl(cc,2) +...
  (2*kl(cc,4)) + (2*kl(cc,6)) + kl(cc,8)));
    m=m+1;
    yp(m,2:3)= [kl(cc,9) kl(cc,10)];


    %bisection method

    if (m>10) && (kl(cc,9)==x)
        poin(ff,1:2) = [yp(m,2) yp(m,3)];
        ff=ff+1;
        break
    end

    if ((m>10) && (x<yp(m,2)) &&  (x > yp(m-1,2))) || ((m>10)...
        && (x>yp(m,2)) &&  (x<yp(m-1,2)))
    d=n;
    s=h;
     v=m-1; z=d-s;

    r=yp(v,2); w=yp(v,3); g=yp(v+1,2);

    while (abs(r-x)>2e-10)
```

```
if ((x < r) &&  (x > g)) || ((x > r) &&  (x < g))

    s=s/2;

        z=z+s;
        k1 = y{1}(z,[r w])*s;
l1 = y{2}(z,[r w])*s;
k2 = y{1}(z+(s/2),[r+(k1/2) w+(l1/2)])*s;
l2 = y{2}(z+(s/2),[r+(k1/2) w+(l1/2)])*s;
k3 = y{1}(z+(s/2),[r+(k2/2) w+(l2/2)])*s;
l3 = y{2}(z+(s/2),[r+(k2/2) w+(l2/2)])*s;
k4 = y{1}(z+(s/2),[r+k3 w+l3])*s;
l4 = y{2}(z+(s/2),[r+k3 w+l3])*s;
uu = ((1/6)*(k1 + (2*k2) + (2*k3) + k4));
pp = ((1/6)*(l1 + (2*l2) + (2*l3) + l4));
r = r + ((1/6)*(k1 + (2*k2) + (2*k3) + k4));
w = w + ((1/6)*(l1 + (2*l2) + (2*l3) + l4));


else

       r=r-uu;
       w=w-pp;
       z=z-s;


s=s/2;
z=z+s;

k1 = y{1}(z,[r w])*s;
l1 = y{2}(z,[r w])*s;
k2 = y{1}(z+(s/2),[r+(k1/2) w+(l1/2)])*s;
l2 = y{2}(z+(s/2),[r+(k1/2) w+(l1/2)])*s;
k3 = y{1}(z+(s/2),[r+(k2/2) w+(l2/2)])*s;
l3 = y{2}(z+(s/2),[r+(k2/2) w+(l2/2)])*s;
k4 = y{1}(z+(s/2),[r+k3 w+l3])*s;
l4 = y{2}(z+(s/2),[r+k3 w+l3])*s;
uu = ((1/6)*(k1 + (2*k2) + (2*k3) + k4));
pp = ((1/6)*(l1 + (2*l2) + (2*l3) + l4));
r = r + ((1/6)*(k1 + (2*k2) + (2*k3) + k4));
w = w + ((1/6)*(l1 + (2*l2) + (2*l3) + l4));

end

end
```

```
    poin = [r w];
    break
    end
    cc=cc+1;
end
 PM=poin;
if isempty(poin) == 1
    %disp(['The vector field does not intersect the Poincare',...
    %    ' section with the given initial values and time flow']);
else
    PM = poin;
end
```

6.1.3 Poincaré Nth Step

```
function PM = PoincareNStep(y,y0,x,ns,tspan,h)
point = y0;
%When using this function you must comment line 100-101 of the function PoincareStep
for i = 1:ns
    point = PoincareStep(y,tspan,point,h,x);
    if isempty(point) == 1
       % disp(['The vector field does not intersect the Poincare',...
        %' section ',num2str(ns),' times with the given initial values and time flow
    break
    end

end
PM=point;
```

6.1.4 Poincare Map

```
function PM = PoincareGraph(y,y0,x,ns,tspan,h)
n=length(y0);
fy=zeros(n);
a=0;
%when calling this function it is adviced to comment line 100-101 of
%PoincarStep and line 7-8 of PoincareNStep to avoid repeated results
for i = 1:n
    point = [x y0(i)];
    point = PoincareNStep(y,point,x, ns, tspan,h);
```

```
    if isempty(point) == 1
        disp(['The vector field does not intersect the Poincare',...
        ' section ',num2str(ns),' times with the given initial',...
        ' value ',num2str([x y0(i)]),' and time flow ', num2str(tspan)]);
    a=1;
    break
    end

    fy(i)=point(2);
end
if a~=1
plot(y0,fy(1:n),'r-',y0,y0,'b-')
title('Poincare Graph compared with y=x')
legend('Poincare Graph','y=x Graph')
end
PM=0;
```

6.1.5 Poincaré Map Bisection

```
function PM = PoincareGraphBisec(y,a,b,x,ns,tspan,h)
pointa=[x a];
pointb=[x b];
aux = PoincareNStep(y,pointa,x, ns, tspan,h);
ga= aux(2)-a;
aux = PoincareNStep(y,pointb,x, ns, tspan,h);
gb= aux(2)-b;

while abs(b-a)>10e-10
    mid=(a+b)/2;
    pointmid=[x mid];
    aux = PoincareNStep(y,pointmid,x, ns, tspan,h);
    gmid=aux(2)-mid;

    if gmid*ga<0
        b=mid;
        gb=gmid;

    else
        a=mid;
        ga=gmid;
    end
end
```

```
PM=mid;
```