Report for exercise 3 from group G

Tasks addressed:	5
Authors:	Angelos Kafounis Caner Karaoglu Joaquin Gomez Sanchez
Last compiled:	2022–08–31
Source code:	`https://github.com/joaquimgomez/ml-crowds-group-g`

The work on tasks was divided in the following way:

Angelos Kafounis	Task 1	35%
	Task 2	40%
	Task 3	30%
	Task 4	30%
	Task 5	30%
Caner Karaoglu	Task 1	35%
	Task 2	30%
	Task 3	35%
	Task 4	40%
	Task 5	30%
Joaquin Gomez Sanchez	Task 1	30%
	Task 2	30%
	Task 3	35%
	Task 4	30%
	Task 5	40%

Report on task 1, Vector fields, orbits, and visualization

The first task of the third exercise is to build a figure similar to Fig. 2.5 in the book of Kuznetsov [1]. In Table 1 is our final construction of the figure.

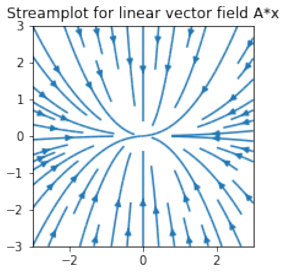
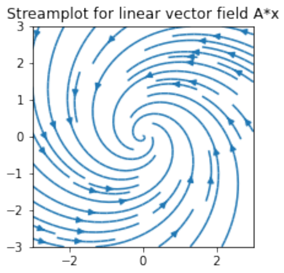
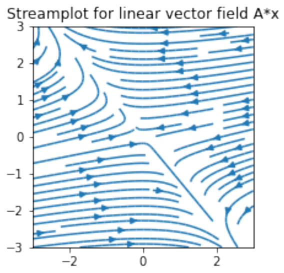
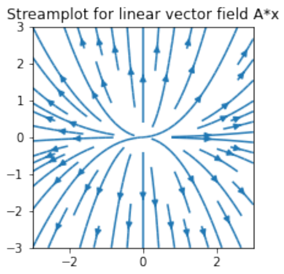
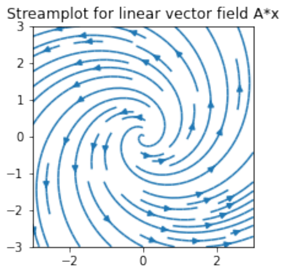
Name	**Phase portrait**	**Matrix**	α
Stable node		$A_\alpha = \begin{bmatrix} \alpha & 0 \\ 0 & -2 \end{bmatrix}$	-1
Stable focus		$A_\alpha = \begin{bmatrix} \alpha & -1 \\ 1 & \alpha \end{bmatrix}$	-0.5
(Unstable) Saddle		$A_\alpha = \begin{bmatrix} \alpha & \alpha \\ -\frac{1}{4} & 0 \end{bmatrix}$	-1
Unstable node		$A_\alpha = \begin{bmatrix} \alpha & 0 \\ 0 & 2 \end{bmatrix}$	1
Unstable focus		$A_\alpha = \begin{bmatrix} \alpha & -1 \\ 1 & \alpha \end{bmatrix}$	0.5

Table 1: Construction of the Fig. 2.5 of [1]

In order to obtain the correct phase portraits we have used 3 different matrices instead of just one.

For the saddle phase portrait we have used the provided as an example matrix, but setting $\alpha = -1$. In this case we do not obtain the same picture as in [1], instead we get a rotated version.

For the other two matrices we have learn through different notes[1][2] how this kind of systems work and how to model them through matrices.

For the node type we have figured out that the behavior will depend on the eigenvalues of the matrix. The simplest way to work with the eigenvalues of a matrix is having a diagonal matrix, where $\text{diag}(A) = \{\lambda_1, \lambda_2\}$ given $A \in \mathbb{R}^{2\times 2}$. With this, if we have that $0 > \lambda_1 > \lambda_2$, we will have a stable node. We set for our matrix $\alpha = -1$ and the condition is true since in our matrix $\lambda_2 = -2$. While maintaining the condition, the matrix will produce the expected stable node. Finally, for the unstable node the condition is that $0 < \lambda_1 < \lambda_2$. Fixing $\lambda_2 = 2$, we have chosen $\alpha = 1$.

For the focus type we figured out that we need a matrix with the form:

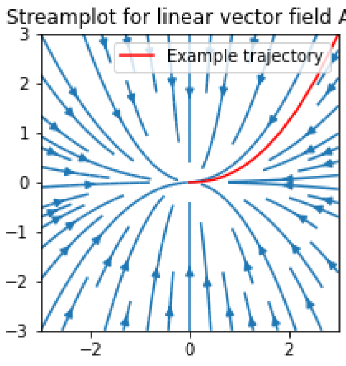
$$\begin{bmatrix} a & -b \\ b & a \end{bmatrix} \tag{1}$$

If we have that $a < 0$ and $b > 0$, we will have a stable focus; and if we have that $a > 0$ and $b > 0$, we will have an unstable focus. Then, b should be always > 0 and the parameter to obtain stable or unstable focuses is a, i.e., our α. In our designed matrix

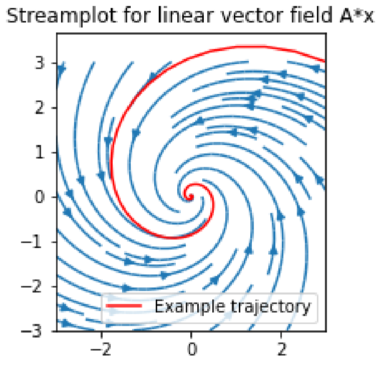
$$A_\alpha = \begin{bmatrix} \alpha & -1 \\ 1 & \alpha \end{bmatrix} \tag{2}$$

we set $\alpha = -0.5$ in order to obtain a stable focus and $\alpha = 0.5$ for the unstable one.

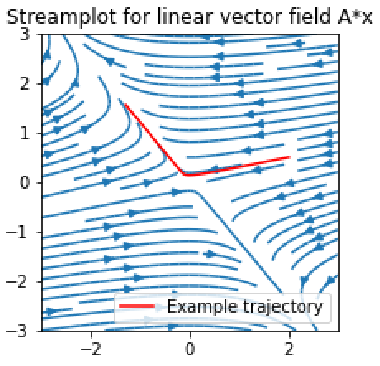
In order to double check that our systems behave as described we have performed simulations with them. The results can be seen in Figure 1. We can see, given the starting points, that the trajectories are the correct ones, i.e., the expected ones for the different phase portraits.



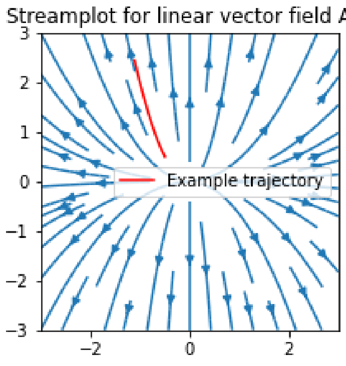
(a) Stable node (starting at (3, 3) and for 15 time steps)



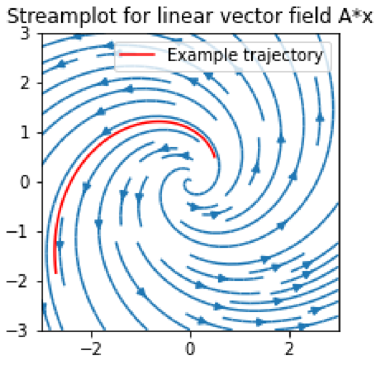
(b) Stable focus (starting at (3, 3) and for 15 time steps)



(c) (Unstable) Saddle (starting at (2, 0.5) and for 15 time steps)



(d) Unstable node (starting at (-0.5, 0.5) and for 0.8 time steps)



(e) Unstable focus (starting at (0.5, 0.5) and for 3 time steps)

Figure 1: Trajectories for the different phase portraits in Table 1

[1]Lecture notes from Prof. Dr. Artem S. Novozhilov for the course "Math 266: Introduction to Ordinary Differential Equations". Available at `https://www.ndsu.edu/pubweb/~novozhil/Teaching/266%20Data/lecture_23.pdf`. Accessed on 06/05/2022.

[2]Lecture notes from Prof. Dr. Ching-Li Chai. Available at `https://www2.math.upenn.edu/~chai/240s15/240hws15/linear_plane_autonomous.pdf`. Accessed on 06/05/2022.

Finally, we can state that the systems here described are not topologically equivalent since they have not the same orbit and the eigenvalues of the equilibrium in the systems differ from the eigenvalues of the others.

All the code used to obtain the exposed results is in the Jupyter notebook `Task1.ipynb`. In order to produce the plots we have used the provided method `plot_phase_portrait` from `utils.py`. For the simulation of the trajectories we have used the provided Euler solder method `solve_euler`.

Report on task 2, Common bifurcations in nonlinear systems

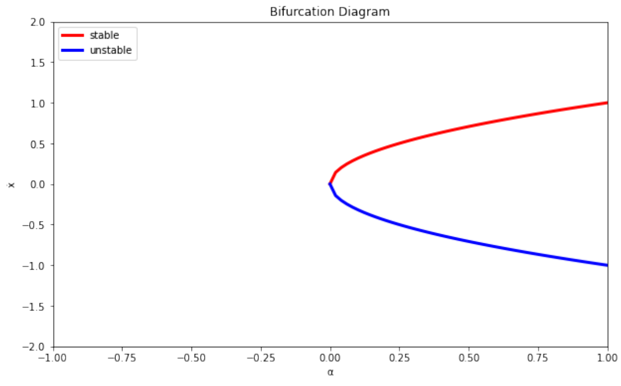
Now, let us analyze two different dynamical systems with the evolutions described by

$$\dot{x} = \alpha - x^2 \tag{3}$$

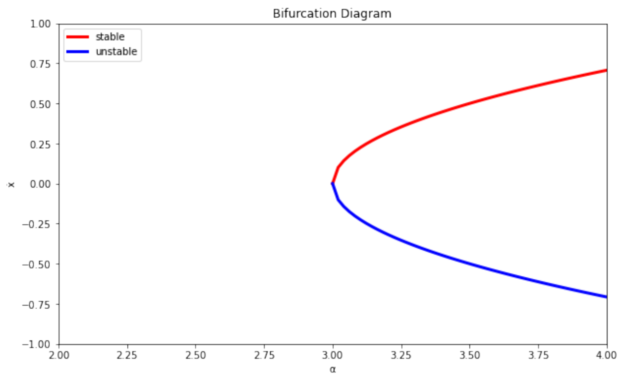
and

$$\dot{x} = \alpha - 2x^2 - 3 \tag{4}$$

The bifurcation diagrams for both systems are in Figure 2. For the first system we plot the bifurcation diagram for values of $\alpha \in [0, 1]$ and for the second system for values of $\alpha \in [3, 4]$.



(a) System Equation 3



(b) System Equation 4

Figure 2: Bifurcation diagrams for systems described by Equation 3 and Equation 4

For the first system, and following the classification of [1], at $\alpha = 0$ we can observe that we have a bifurcation of type saddle-node bifurcation. We have the same for the second system. We can see that both have a stable bifurcation in red and an unstable bifurcation in blue.

For both systems Equation 3 and Equation 3 we can say that they are not topologically equivalent at $\alpha = 1$ because, while the first system is defined for $\alpha = 1$, the second system is not:

$$\alpha = 1 \rightarrow x = \pm\sqrt{\frac{\alpha - 3}{2}} \notin \mathbb{R} \tag{5}$$

and we cannot say that a not defined system is topologically equivalent to another.

The same occurs for $\alpha = -1$, but for this case neither one nor the other are defined and we cannot conclude that they are topologically equivalent.

Regarding the normal form, both systems have the same normal form because we could obtain the same system by applying a transformation to both systems, this preserving the essential behavior of them.

All the code used to obtain the exposed results is in the Jupyter notebook `Task2.ipynb`. We also have added to the provided `utils.py` file a method `plot_bifurcation_diagram` to obtain the bifurcation diagrams.

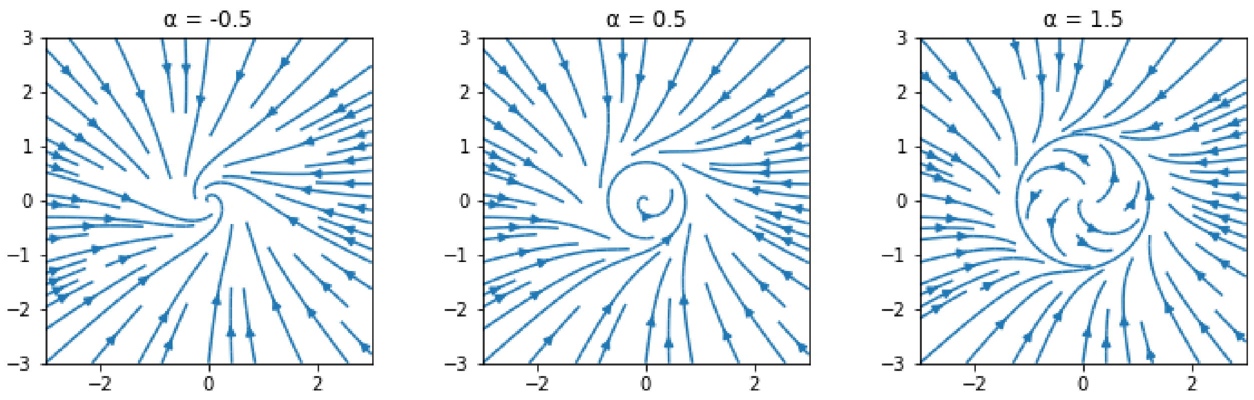
Report on task 3, Bifurcations in higher dimensions

Now, let us analyze two special bifurcation in higher dimensions. The first bifurcation is the **Andronov-Hopf bifurcation** [1] with the vector field in normal form

$$\dot{x}_1 = \alpha x_1 - x_2 - x_1(x_1^2 + x_2^2) \tag{6}$$

$$\dot{x}_2 = x_1 - \alpha x_2 - x_2(x_1^2 + x_2^2) \tag{7}$$

First, let us visualize the bifurcation of the system by plotting three phase diagrams at representative values of α, i.e., $\alpha \in \{-0.5, 0.5, 1.5\}$.

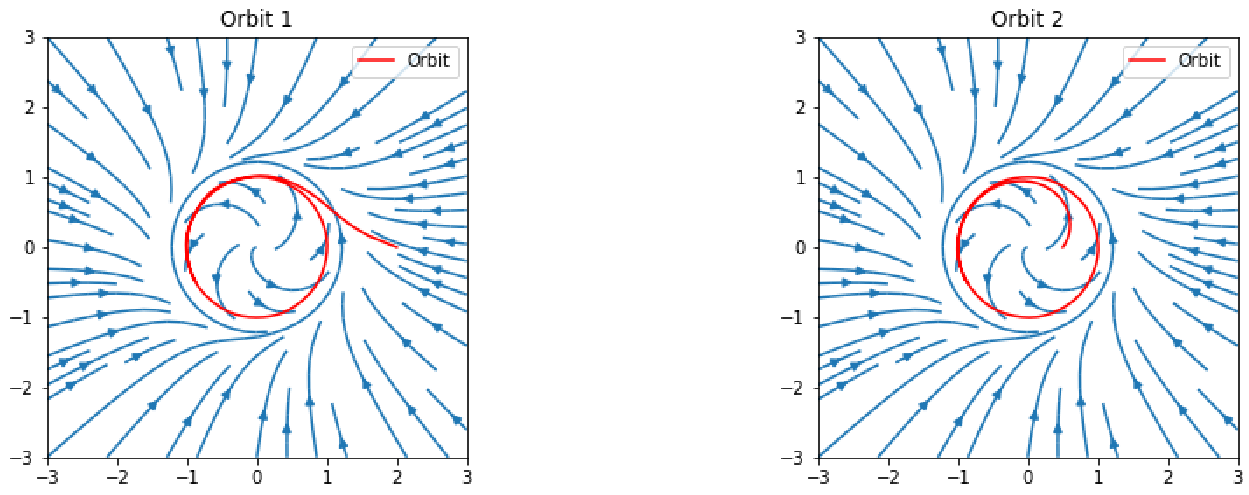


(a) $\alpha = -0.5$ (b) $\alpha = 0.5$ (c) $\alpha = 1.5$

Figure 3: Andronov-Hopf bifurcation of Equation 6 and Equation 7

We can see in Figure 3 that, when we increase alpha, the system creates a stable focus, but for a big one like $\alpha = 1.5$ we have a stable focus with another "stable focus" inside.

Fixing $\alpha = 1$, in Figure 4 we can see two different orbits for the system, starting at $(2, 0)$ and $(0.5, 0)$. With this two orbits we can see the mentioned before behavior similar to a "black hole". In the first orbit the starting point is outside of the "inner stable focus", but the "outside stable focus" put the trajectory inside and it remains there. In the second orbit the starting point is already inside the "inner stable focus" and during the simulation the trajectory remains inside, following a circle at the limit of the inner area.



(a) Starting point at $(2, 0)$ (b) Starting point $0.5, 0$

Figure 4: Two orbits of the system (Equation 6 and Equation 7) starting at different points

As a solver for the orbits we have used the `solve_ivp` solver from `scipy.integrate`. This solver, unlike the provided Euler's one, receives as input a function with the system equations.

We have also added to `utils.py` an extra `plot_phase_portrait` method that directly works with X, Y and the velocities U, V instead of a matrix. All the code used to obtain the exposed results is in the Jupyter notebook `Task3.ipynb`.

The second bifurcation to analyze in this task is the **cusp bifurcation**, with normal form

$$\dot{x} = \alpha_1 + \alpha_2 x - x^3 \tag{8}$$

As it can be seen, the bifurcation occurs in one state space dimension $X \in \mathbb{R}$, but with two parameters $\alpha \in \mathbb{R}^2$.

In Figure 5 we have visualized the bifurcation surface of the cusp bifurcation. We have α_1 and α_2 on the bottom plane and x in the third direction. In order to plot, we have followed the suggested way, i.e., we sample points (x, α_2) uniformly and then we plot the surface as a function α_1 of (x, α_2). From the visualization we can conclude that it is called "cusp" because we have a point (or a set of points in a surface) on the curve where a moving point must reverse direction.

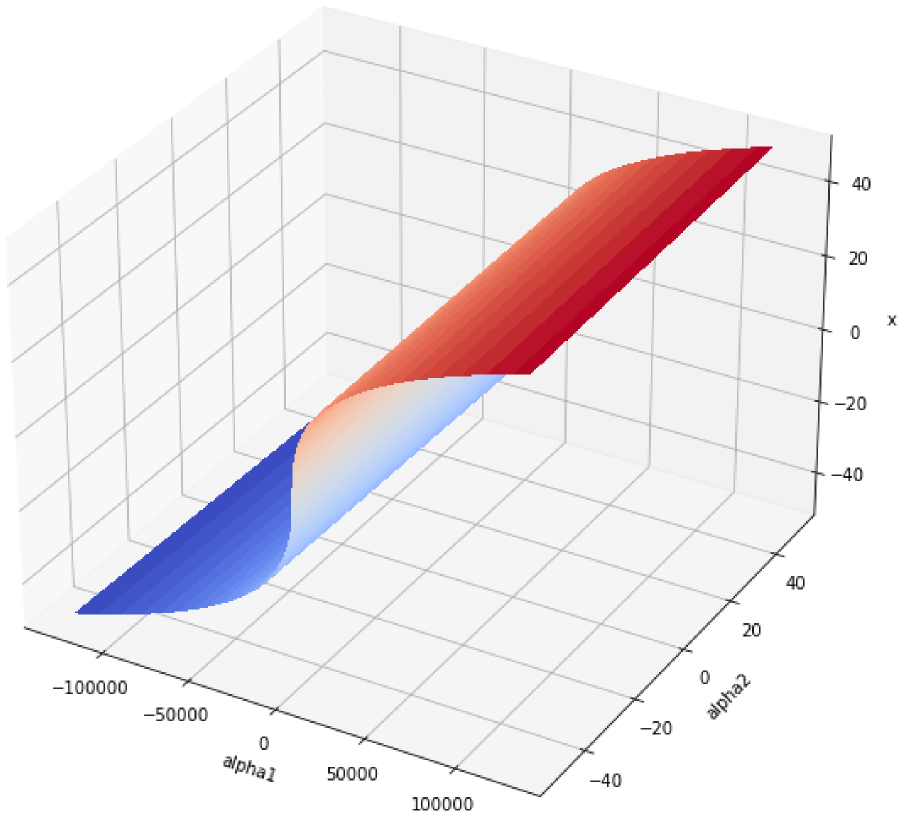


Figure 5: Visualization of the cusp bifurcation

All the code used to obtain the exposed results is in the Jupyter notebook `Task3.ipynb`.

Report on task 4, Chaotic dynamics

For the first part of this task we have to study the discrete map called **the logistic map**, described by

$$x_{n+1} = r x_n (1 - x_n), \quad n \in \mathbb{N}, \tag{9}$$

with the parameter $r \in (0, 4]$.

Firstly, let us see what happens when we vary r from 0 to 2. The results are in Figure 6. We can see how, for values between 0 and 1.2, the system goes to 0, while for values greater than 1.2 the system goes to 0.5. This is a bifurcation in the system with two steady states, one from 0 to ~ 1.2 and another from ~ 1.4 to 2.0. We have tried with different x_0 values and, for example, if we set $x_0 = 0.5$, then the system never goes up. As a chaotic system, it is highly vulnerable due to the given initial point.

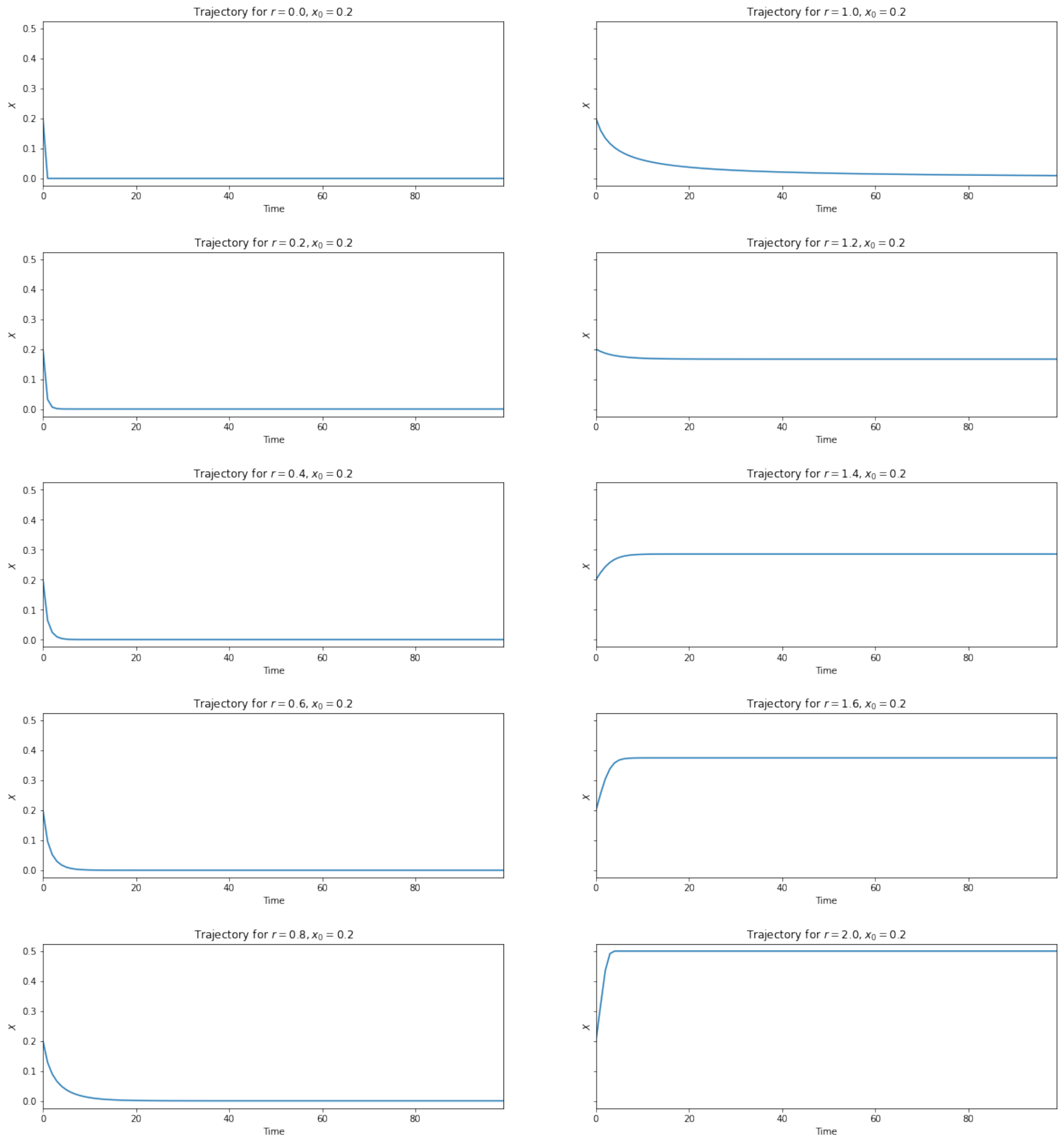


Figure 6: Trajectory for different values of r between 0 and 2, $x_0 = 0.2$ and 100 iterations

For the range between 0 and 2, we can see that the system converges to the well-known logistic curve, and this is the origin of the name for this bifurcation.

Now, let us see what happens for r from 2 to 4. The results are in Figure 7. We can see how, for values between 2 and 4, the systems goes up until an stabilization and then it starts to oscillate. This oscillations seems periodic until r around 3.6 or 3.8. For $r = 4.0$ the oscillation does not look periodic, at least for the tried number of iterations. If we try a different initial value, for example $x_0 = 0.5$, we have the same behavior, except for $r = 4.0$, for which the systems grows to 1 and then decreases and stabilizes at 0. We have the aforementioned vulnerability due to the given initial point of the chaotic systems.

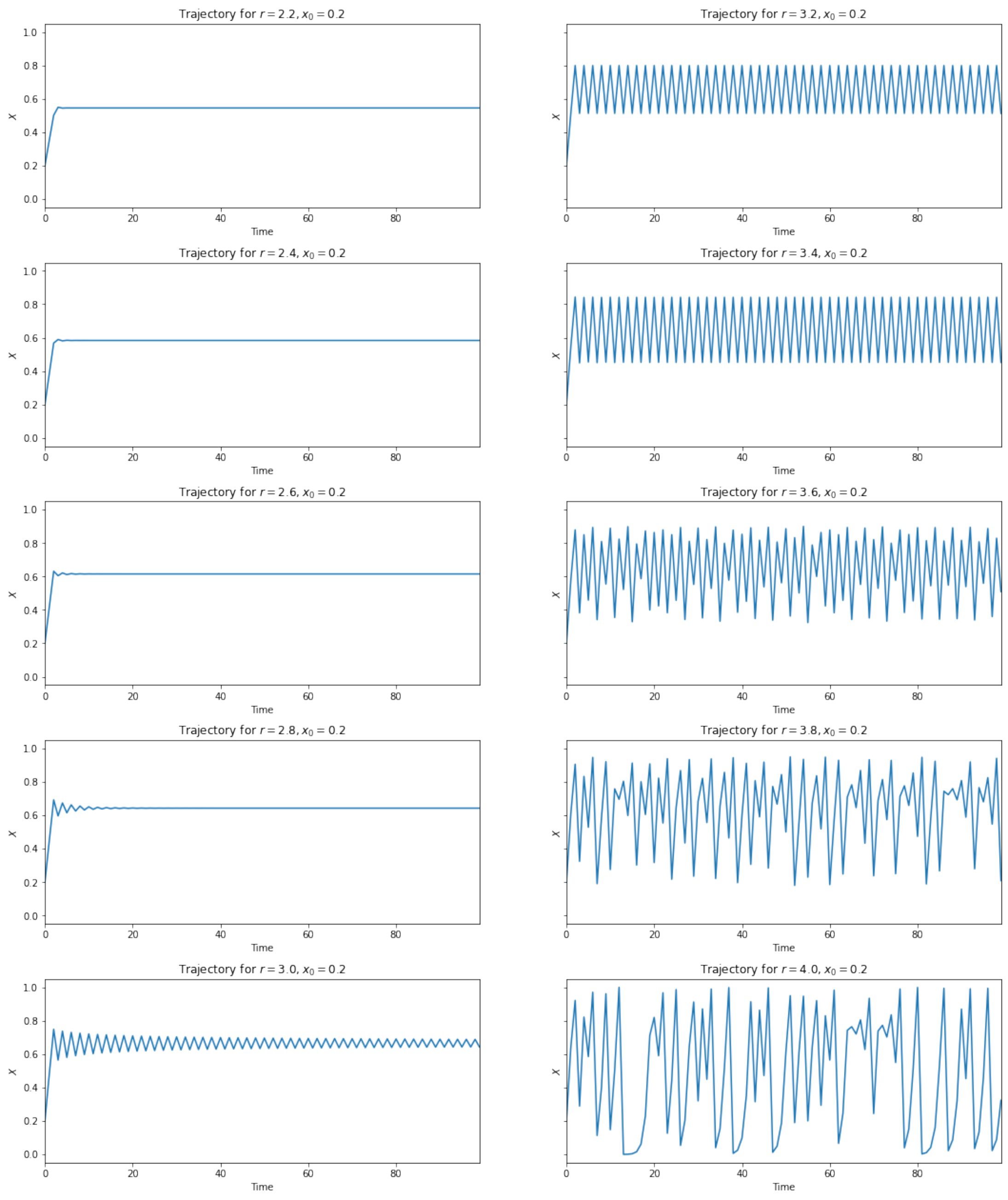


Figure 7: Trajectory for different values of r between 2 and 4, $x_0 = 0.2$ and 100 iterations

As a last analysis for this first part, we are going to plot a bifurcation diagram of the system for r between 0 and 4 (horizontal axis) and x between 0 and 1 (vertical axis). We can see the result in Figure 8.

We can clearly see 4 big steady states between 0 and 1, 1 and ∼3, and 3 and 3.5. After 3.5 we can (hardly) see 10 more steady states and after this, it is impossible. The easy to see bifurcation points are at $r = 1.0$, $r =\sim 3.0$ and $r =\sim 3.5$.

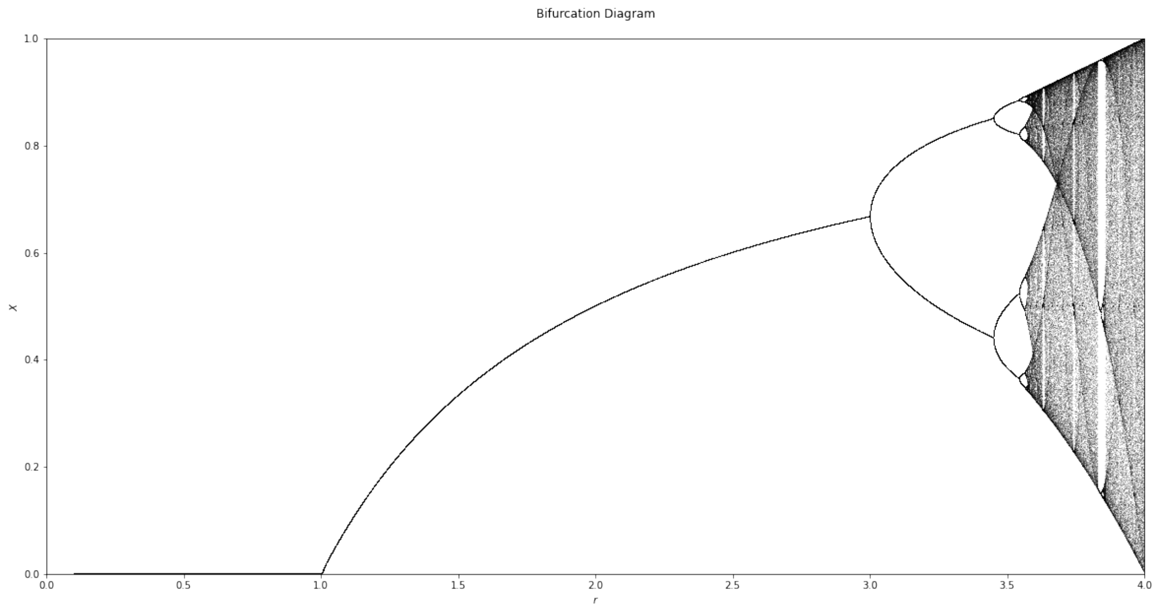


Figure 8: Bifurcation diagram of the system described by Equation 9

For the second part of this task we have to analyze the well-known **Lorenz attractor** [2]. In Figure 9 (left) we can see a trajectory of the system tarting at $x_0 = (10, 10, 10)$ until time $T_{end} = 1000$ with parameter values $\sigma = 10$, $\beta = 8/3$ and $\rho = 28$.

We can see that we have the well-known attractor with two ”gravity centers” that exchange an imaginary point that ”rotates” around them.

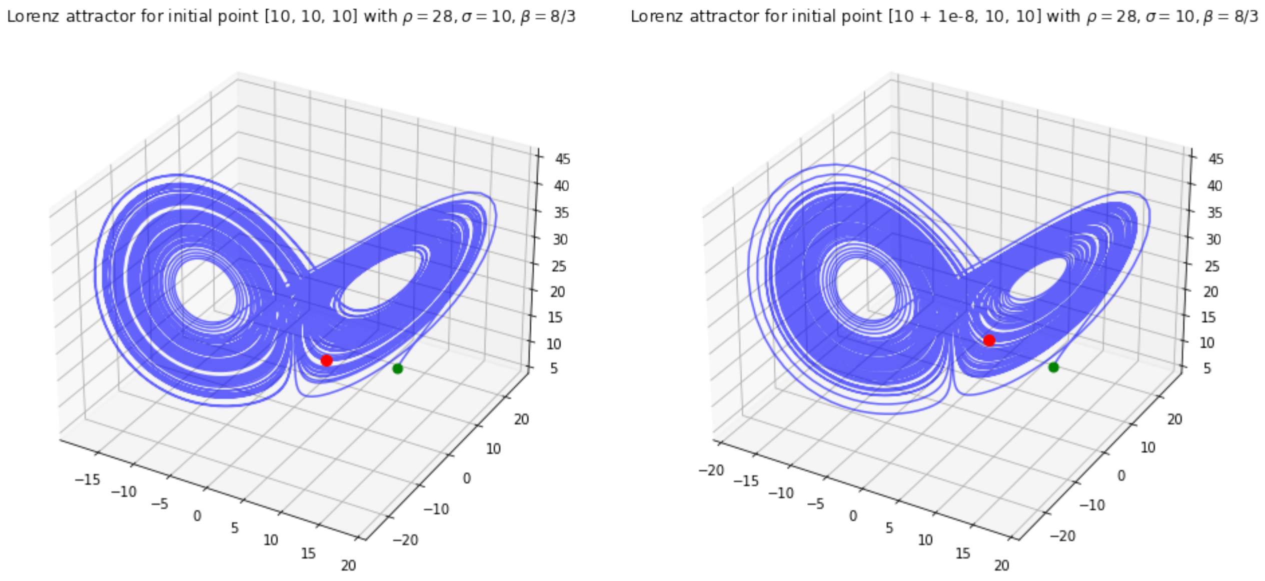
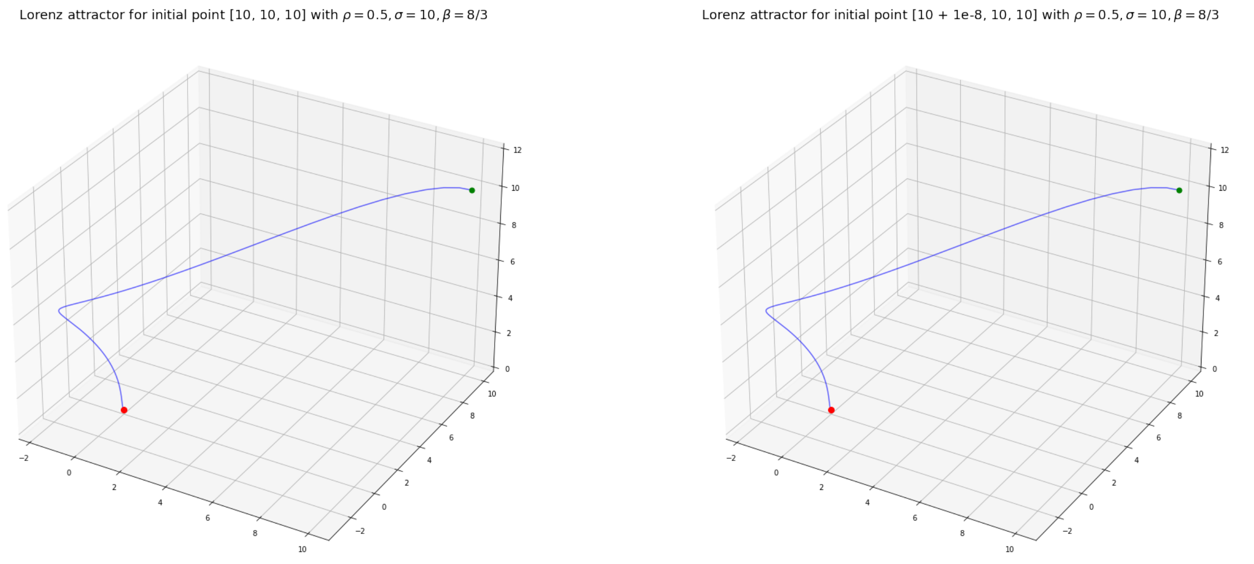


Figure 9: Lorenz attractor simulations starting at $x_0 = (10, 10, 10)$(left) and at $\hat{x}_0 = (10 + 10^{-8}, 10, 10)$, both until time $T_{end} = 1000$ and parameter values $\sigma = 10$, $\beta = 8/3$ and $\rho = 28$

As the exercise sheet states, small perturbations in the initial conditions affect the trajectories drastically, this because of the chaotic nature of such systems. We have simulated the same system again with a small perturbation on the x-coordinate of the initial condition. We have visualized this change by calculating the Euclidean distance between two systems. If the distance between these two systems are larger or equal than 1, trajectories and end points of the system are not the same. The reason for this is that the error is starting to expand the diameter of the Lorenz attractor ellipsoid. Simulation take approximately 0.02262 seconds to extend this 1 limit. We can see the comparison of both simulations in Figure 9.

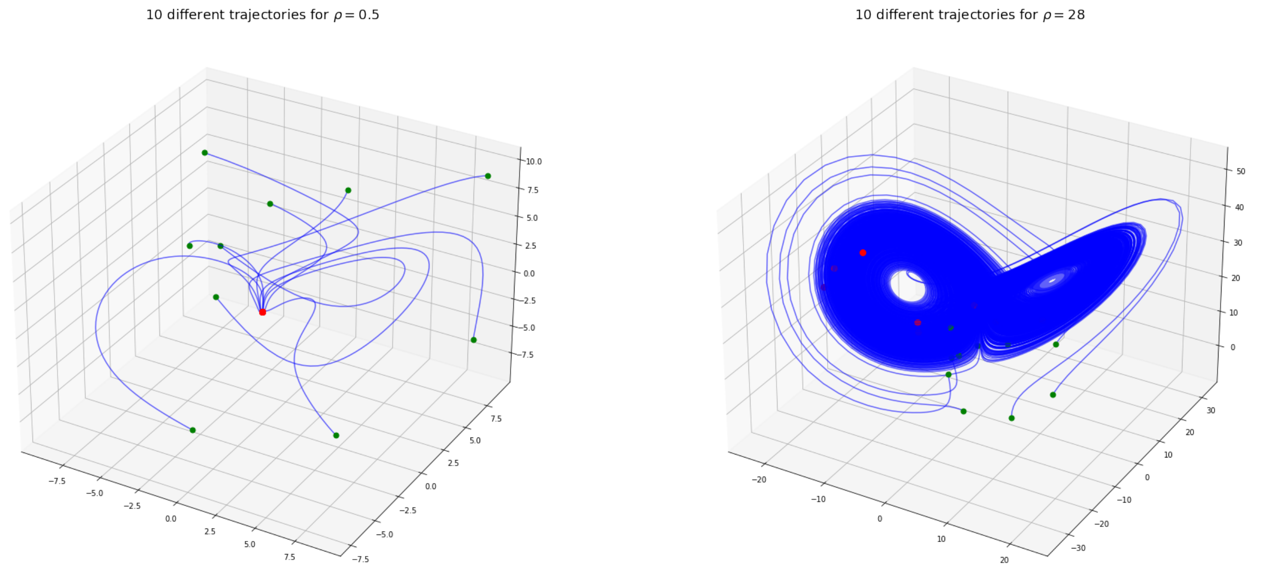
Now, let us change the parameter $\rho = 0.5$ and run different simulations. First, we have simulated the same original initial condition x_0, and perturbed initial condition $\hat{x}_0$ under the new ρ value. We can see that in Figure 10, both systems mostly correlate with each other even when the ρ value is different.



(a) Original system starting from $(10, 10, 10)$ (b) Perturbed system starting from $(10 + 10^{-8}, 10, 10)$

Figure 10: Original and perturbed initially conditioned systems under $\rho = 0.5$

In addition to this experiment, we have artificially created 10 different initial points for the Lorenz attractor system and tested and visualized these both where $\rho = 0.5$ and $\rho = 28$. These initial points can be found in the `Task4.ipynb` Jupyter Notebook. The reader can easily detect that, in the first case, points will converge and stay on the same final position. But in the latter case, there are 2 different convergence points for the system. This can be easily seen in Figure 11.



(a) 10 trajectories with $\rho = 0.5$ (b) 10 trajectories with $\rho = 28$

Figure 11: Visualization of 10 same initial starting conditions (green) using $\rho = 0.5$ and $\rho = 28$

All the code used to obtain the exposed results is in the Jupyter notebook `Task4.ipynb`. For the visualizations of the first part of the task we have taken inspiration from [3] in order to code the plots, because, for example, we have never plotted before a bifurcation diagram like the one in this task. For the simulation of the Lorenz attractor we have used the solver `odeint` from `scipy.integrate`, which allows us to integrate a system of ordinary differential equations.

Report on task 5, Bifurcations in crowd dynamics

For the final task we have to analyze bifurcations, but now in crowd dynamics. Concretely, we have to analyze the SIR model, previously introduced in Exercise 2.

First of all, we have to fix some possible missing imports. In our case, we have detected no missing import in the provided Jupyter Notebook. Then, we have to finish the implementation of the SIR model, provided in `sir_model.py`, by implementing the SIR model equations.

In Listing 1, it can be seen the final implementation of the SIR equations, which has been easy to implement since the different differential equation are simple.

```
def model(t, y, mu0, mu1, beta, A, d, nu, b):
    """
    ...
    """
    S,I,R = y[:]
    m = mu(b, I, mu0, mu1)

    dSdt = A - d * S - (beta * S * I) / (S + I + R)
    dIdt = -(d + nu) * I - m * I + (beta * S * I) / (S + I + R)
    dRdt = m * I - d * R

    return [dSdt, dIdt, dRdt]
```

Listing 1: SIR model implementation in ‘sir_model.py‘

After implementing the introduced SIR equations, and using the provided SIR parameters, we have tested the model for different b (i.e., the number of hospital beds per 10,000 persons) in the range between 0.01 and 0.03. The Figure 12 illustrates the obtained simulation results for different values of b (0.01, 0.015, 0.02, 0.022, 0.25 and 0.03).

We can see that for $b = 0.01$ and $b = 0.015$ the infections and removes fluctuates until both reach a stabilization around 50. This appears to be due to the number of beds, if there are no sufficient beds per number of persons, they cannot be properly recovered and they can cause more infections. The final stabilization is probably due to the equal number of infections and removes at each time, i.e., the number of beds is the one for collapsing both measures.

For values of b between 0.02 and 0.03 the number of infections and removes rapidly decrease, meaning this that the number of beds is sufficient to break the propagation of the infections for the provided SIR parameters.

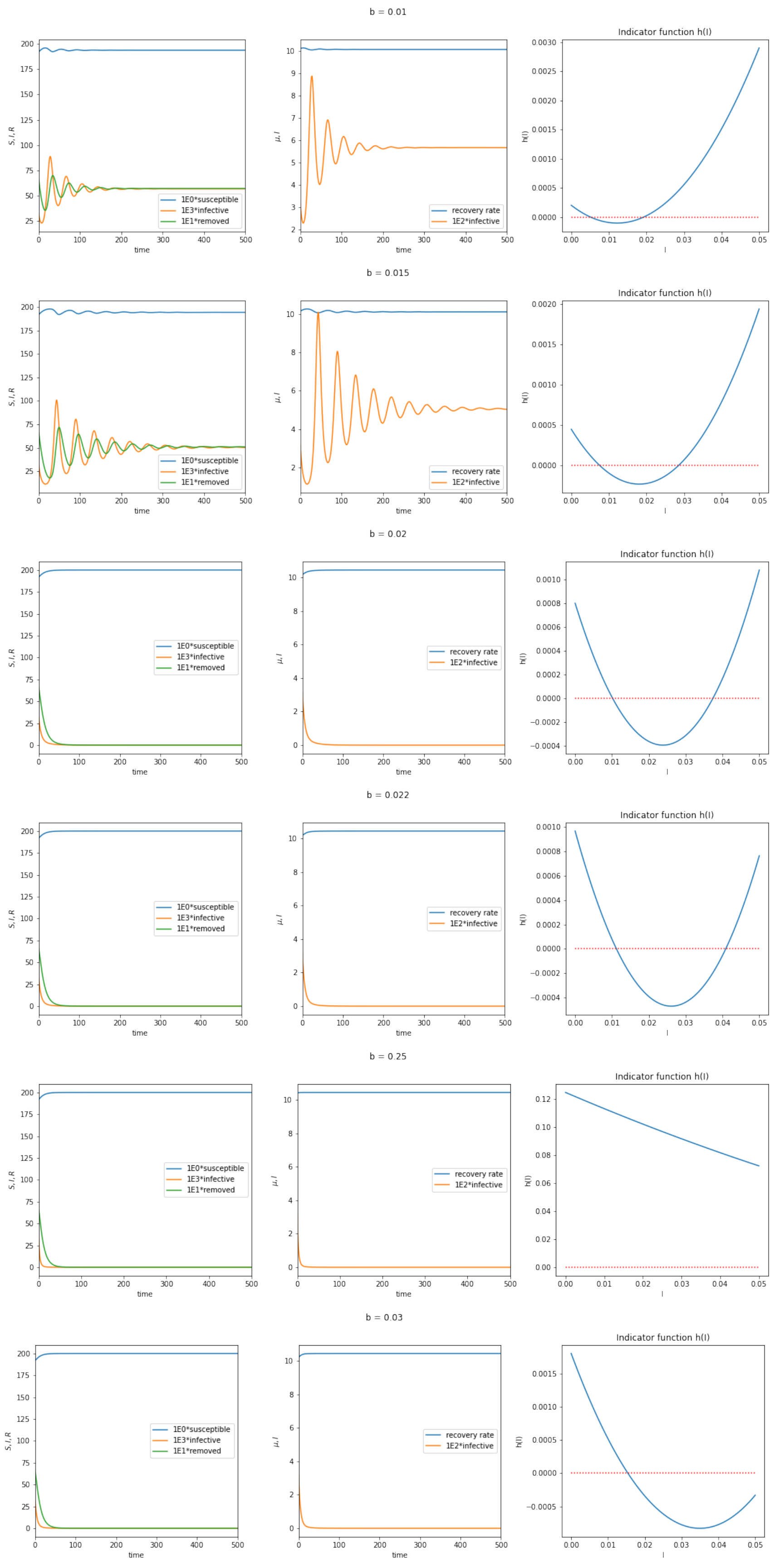
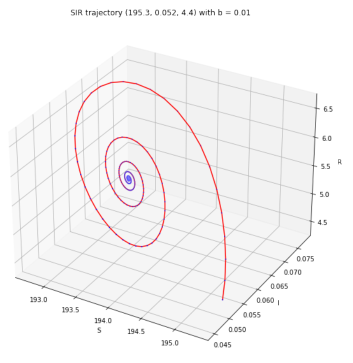
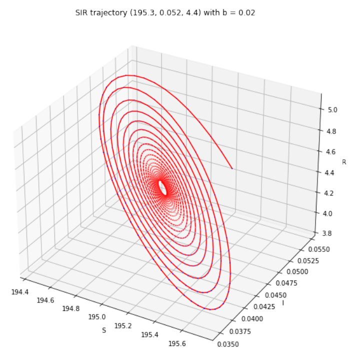


Figure 12: Simulations of SIR model for different values of b

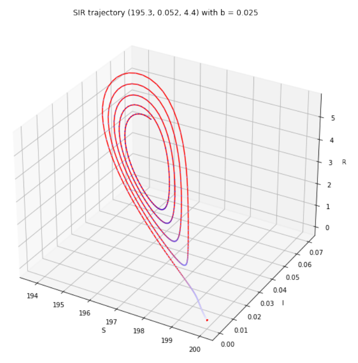
Now, following with the last idea, let us see what happens for different values of b, but at 3 different starting points $(S_0, I_0, R_0) = (195.3, 0.052, 4.4)$ (see 1st row of Figure 13), $(195.7, 0.03, 3.92)$ (see 2nd row of Figure 13) and $(193, 0.08, 6.21)$ (see 3rd row of Figure 13). For the different simulations, all parameters were left as they were, including the end time at 1000.



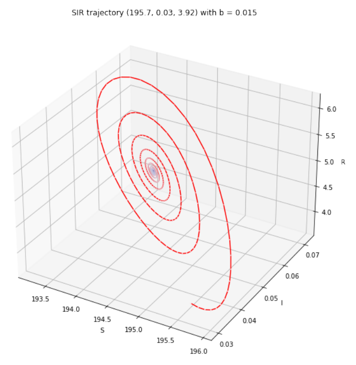
(a) $(195.3, 0.052, 4.4)$ and $b = 0.01$



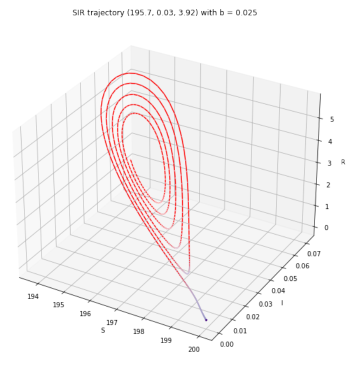
(b) $(195.3, 0.052, 4.4)$ and $b = 0.2$



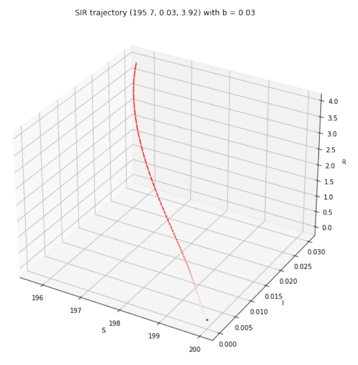
(c) $(195.3, 0.052, 4.4)$ and $b = 0.025$



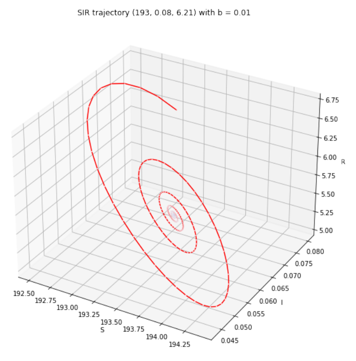
(d) $(195.7, 0.03, 3.92)$ and $b = 0.015$



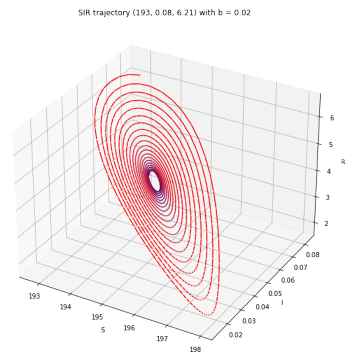
(e) $(195.7, 0.03, 3.92)$ and $b = 0.025$



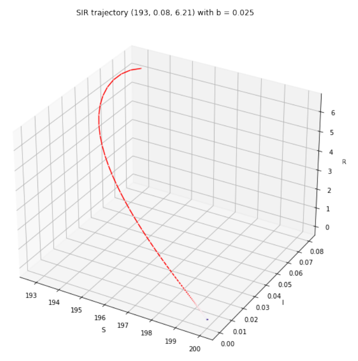
(f) $(195.7, 0.03, 3.92)$ and $b = 0.03$



(g) $(193, 0.08, 6.21)$ and $b = 0.01$



(h) $(193, 0.08, 6.21)$ and $b = 0.02$



(i) $(193, 0.08, 6.21)$ and $b = 0.025$

Figure 13: Simulations of SIR model for different starting points and values of b

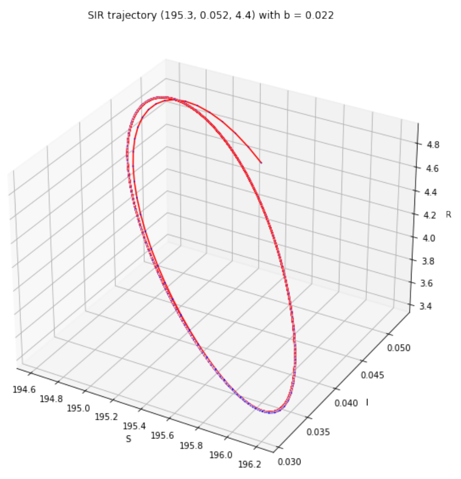
Comparing the last results from Figure 13 with the previous ones in Figure 12, we can see the previously explained: if we have a small number of beds, the number of infections and recoveries fluctuates over the time until both reach the same stabilized value. This can be seen for the two first columns in the figure.

If the number of beds is sufficient to break the transmission, the infections and removes rapidly decrease to 0. This is what happens in Figure 13f and Figure 13i.

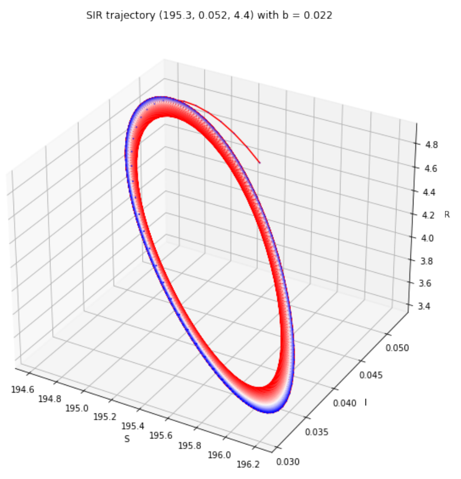
Broadly observing Figure 13, we can see that for the three different starting points the evolution of the behavior is the same when changing b. What is different is the break point of the infections, which, as we already now, depends on b, but also on the starting point.

The most remarkable aspect of the plots is the change between $b = 0.02$ and $b = 0.03$. For $b =\sim 0.025$ we can see that we have an unstable focus, but for $b = 0.03$ the focus have disappeared and we have no bifurcation. This occurs earlier for the third row/starting point.

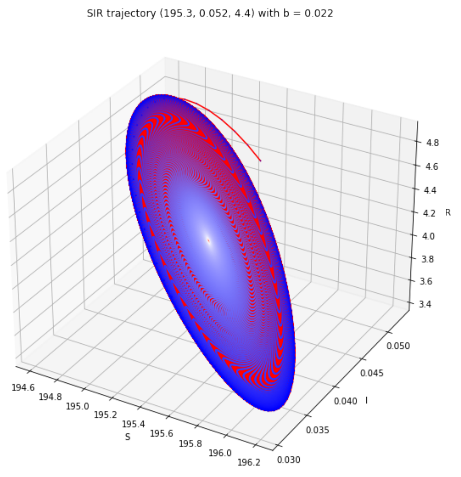
The provided notebook suggest to test what happens for $b = 0.022$ and the first starting point. As we can see in Figure 14a, we need more than 1000 of simulation time to see how the models behaves. Simulating for 100000 we obtain that the model fluctuates for a long time until it reaches a stabilized point.



(a) $t_{\text{end}} = 1000$



(b) $t_{\text{end}} = 10000$



(c) $t_{\text{end}} = 100000$

Figure 14: Simulations of SIR model for $(S_0, I_0, R_0) = (195.3, 0.052, 4.4)$, $b = 0.022$ and different simulation times

In the paper [4], the authors define the reproduction rate $\mathbb{R}_0$ as

$$\mathbb{R}_0 = \frac{\beta}{d + \nu + \mu_1} \tag{10}$$

which means that the reproduction rate is directly proportional to the average number of adequate contacts per unit time with infectious individuals (β) and inversely proportional to the sum of the per capita natural death rate of the population (d), the per capita disease-induced rate (ν) and the per capita recovery rate of infectious individuals (μ_1) when $b \to \infty$.

From the Equation 10 we can say that, if the number β increases, i.e., the average number of adequate contacts per unit time with infectious individuals increases, then we have an increased number of adequate contacts per unit time that can be infected by infectious individuals. With this, the number of infections will increase and also the number of ineffective persons. In order to counteract this effect, following Equation 10, the number of recoveries, for example, should increase. Obviously, we have the opposite case if β decreases.

In the paper [4] we also have the following theorem:

Theorem 1. *For the SIR model, the disease free equilibrium* $E_0 = \left(\frac{A}{d}, 0, 0\right)$ *at* $\mathbb{R}_0 < 0$ *is an attraction node.*

In other words, this theorem states that the disease free equilibrium $E_0 = \left(\frac{A}{d}, 0, 0\right)$ at $\mathbb{R}_0 < 0$ is the case of the first row of the Fig. 2.5 in [1], i.e., we have a stable node hyperbolic equilibrium with eigenvalues $\lambda_1, \lambda_2 \in \mathbb{R}^-$. For values of (S, I, R) close to E_0 we will have a system that reaches its limit faster, ending the transmission of the infection more quickly.

All the code used to obtain the exposed results is in the Jupyter notebook `Task5.ipynb` (a modification of the provided notebook) and in the Python file `sir_model.py`.

References

[1] Yuri A. Kuznetsov. *Elements of applied bifurcation theory.* Vol. 112. Springer, 1998.

[2] Edward N Lorenz. "Deterministic nonperiodic flow". In: *Journal of atmospheric sciences* 20.2 (1963), pp. 130–141.

[3] Cyrille Rossant. *IPython Interactive Computing and Visualization Cookbook: Over 100 hands-on recipes to sharpen your skills in high-performance numerical computing and data science in the Jupyter Notebook.* Packt Publishing Ltd, 2018.

[4] Chunhua Shan and Huaiping Zhu. "Bifurcations and complex dynamics of an SIR model with the impact of the number of hospital beds". In: *Journal of Differential Equations* 257.5 (2014), pp. 1662–1688.