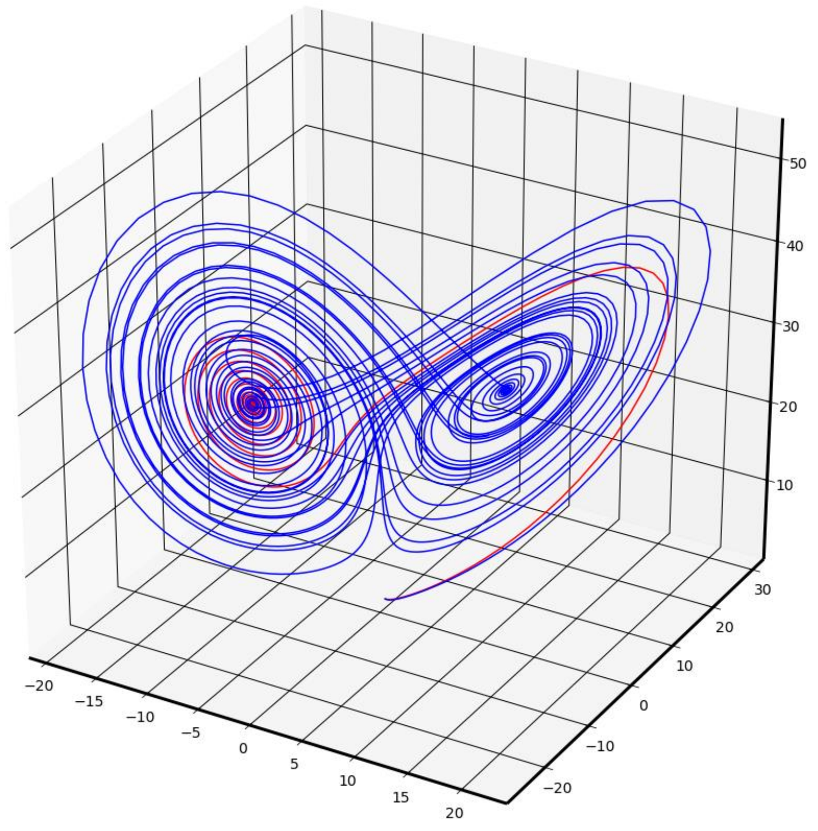
ΑΡΙΣΤΟΤΕΛΕΙΟ ΠΑΝΕΠΙΣΤΗΜΙΟ ΘΕΣΣΑΛΟΝΙΚΗΣ

ΣΧΟΛΗ ΘΕΤΙΚΩΝ ΕΠΙΣΤΗΜΩΝ

ΤΜΗΜΑ ΜΑΘΗΜΑΤΙΚΩΝ

Numerical methods for solving systems of chaotic first-order differential equations

ΦΛΩΡΙΑΣ ΠΑΠΑΔΟΠΟΥΛΟΣ (Α.Ε.Μ.: 874)



ΕΠΙΒΛΕΠΩΝ ΚΑΘΗΓΗΤΗΣ: ΕΥΘΥΜΙΟΣ ΚΑΡΑΤΖΑΣ

ΘΕΣΣΑΛΟΝΙΚΗ 2022

Abstract

This study was made for an assignment of the class "Numerical Methods with Applications to Ordinary and Partial Differential Equations" of my master's program. For the assignment, a differential equation or a system of differential equations had to be chosen and solved using numerical methods and python.

Due to my interest in cryptology, I thought that studying differential equations that are already used in cryptography would benefit me more than anything else and that's why I chose the "Lorenz chaotic equations" for my project. Hence, at the start, this project aimed to solve the Lorenz chaotic equations, as well as one of its extensions, using numerical methods (in theory and in practice, using python). This though quickly changed to trying to solve "all chaotic systems of first-order differential equations" (in practice, by creating a python script that can handle different function inputs for the system), which proved to be a much more interesting goal.

To this end, we start in Chapter 1 by mentioning some key facts and properties of the Lorenz system and then writing a proper form of the generalized problem. In Chapter 2, we introduce the three numerical methods with which the problem was tackled, and finally, in Chapter 3, we show the results of the python code used to solve and visualize the problem for the Lorenz system and other known first-order chaotic systems. Results which, as the reader will see, were in fact quite strange and different depending on the method used, as well as the system that it was used on.

Keywords: Chaotic first-order differential equations, Lorenz system, chaotic cryptography, numerical analysis, Euler method, fourth-order Runge-Kutta, Generalized multidimensional Newton, 4d hyperchaotic Lorenz system, Chen system, Nose-Hoover Oscillator, Sprott-Jafari system

Contents

Chapter 1

Description of the problem

As mentioned in the abstract, for this project, although the starting goal was the solution of the Lorenz chaotic system, as well as one of its extensions (as they were stated in [1], [2] and [3]), it was revealed quite soon that occupation with the more general "chaotic systems of first-order differential equations" would constitute a much more fruitful path to be followed.

Therefore, in this chapter, even though we will begin by giving the background for the Lorenz system and its extensions, we will later occupy ourselves with the broader problem of solving systems of first-order differential equations.

1.1 The Lorenz system

The Lorenz system is a system of three ordinary differential equations which were first used to describe a simplified mathematical model for atmospheric convection, developed in 1963, by Edward Lorenz [4]. It is defined as

$$\begin{cases} \dfrac{dx}{dt} = \sigma(y - x) \\ \dfrac{dy}{dt} = x(\rho - z) - y \\ \dfrac{dz}{dt} = xy - \beta z, \end{cases}$$

where σ, ρ and β are real parameters, which will explain further below.

Our interest in this system, as well as its extensions, stems from its "bizarre" and "chaotic" properties, a lot of which we will simply observe through our visualizations, but not analyze in detail for this project. In truth, these properties are not guaranteed for all values of σ, ρ and β and the term "chaotic" can be used to describe the Lorenz equations only for some values of them. For this reason, from now on, we will be using only the values $\sigma = 10, \rho = 28$ and $\beta = 2.667$, knowing that for different values some of these chaotic properties do not hold and the visualizations change dramatically.

Furthermore, these properties are what makes systems such as the Lorenz system and its extensions, useful in Cryptography too as we will elaborate in more detail at the end of this section.

1.1.1 Origins of the Lorenz system

The Lorenz system is a simplification of one derived by Saltzman in 1962 [5] as a "minimalist" model of thermal convection in a box. Moreover, the Lorenz equations are also often used when thinking about Navier-Stokes dynamics from the perspective of "phase space[1]". Although our prime interest is in the solutions, rather than in its contributions to the convection problem, we shall describe its physical background briefly for purposes of completeness.

The equations stem from examining the flow in a layer of fluid of uniform depth H, with a temperature difference between the upper and lower layer of ΔT, in particular with a linear temperature variation. More precisely, in the case where there is no variation with respect to the y-axis, Saltzman proved that the governing equations can be written as:

$$\begin{aligned}\frac{\partial}{\partial t}\nabla^2\psi &= -\frac{\partial(\psi,\nabla^2\psi)}{\partial(x,z)} + \nu\nabla^4\psi + g\alpha\frac{\partial\theta}{\partial x}\\ \frac{\partial}{\partial t}\theta &= -\frac{\partial(\psi,\theta)}{\partial(x,z)} + \frac{\Delta T}{H}\frac{\partial\psi}{\partial x} + \kappa\nabla^2\theta,\end{aligned}$$

where ψ is a stream function for the two-dimensional motion, θ is the temperature deviation from the steady state, ψ and $\nabla^2\psi$ vanish at the upper and lower boundaries, and g, α, ν and κ denote, respectively the gravitational acceleration, the coefficient of thermal expansion, the kinematic viscosity, and thermal conductivity.

Moreover, Rayleigh, an English mathematician, observed that fields of motions of the form

$$\begin{aligned}\psi &= \psi_0\sin(\pi aH^{-1}x)\sin(\pi H^{-1}z),\\ \theta &= \theta_0\cos(\pi aH^{-1}x)\sin(\pi H^{-1}z),\end{aligned}$$

would develop if $R_a = g\alpha H^3\Delta T\nu^{-1}\kappa^{-1}$, called *Rayleigh* number, exceeded $R_c = \pi^4a^{-2}(1+a^2)^3$.

Finally, by expanding these ψ and θ in double Fourier series and then doing some "extreme" truncations and substitutions, Lorenz arrived at the equations

$$\begin{cases}\dot{X} = -\sigma X + \sigma Y\\ \dot{Y} = -XZ + rX - Y\\ \dot{Z} = XY - bZ,\end{cases}$$

where a dot denotes the derivative with respect to time and the variables X, Y and Z are proportional to the intensity of the convective motion, the temperature difference between ascending and descending currents and the distortion of the vertical temperature profile from linearity, respectively. About the parameters, $\sigma = \kappa^{-1}\nu$ is called the *Prandtl* number, $r = R_c^{-1}R_a$ is the Rayleigh number over the critical Rayleigh number and $b = 4(1+a^2)^{-1}$ gives the size of the region approximated by the system.

The system is only a realistic model of the intended fluid convection if r is close to 1. However, many physical problems can be modelled by the same set of equations for different parameter values [6]. For example, the irregular spiking in lasers, convection in a toroidal region, or even chaotic water can be modelled using them.

1.1.2 The chaotic nature of the Lorenz system

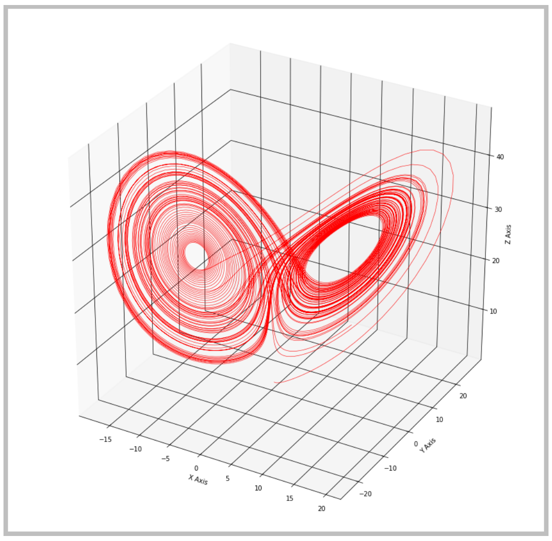
Let's start by talking about Chaos, how is it defined and why do we call the Lorenz equations chaotic? In reality, no definition of the term "chaos" is universally accepted, but almost everyone would agree on the three properties used in this definition by S.H. Strogatz [7]:

*"**Chaos** is aperiodic long-term behaviour in a deterministic system that exhibits sensitive dependence on initial conditions"*

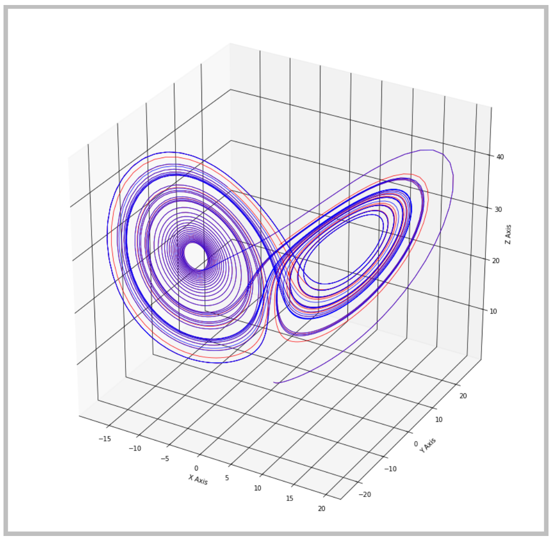
[1]Phase space is an ideal, often multidimensional, space of which the coordinate dimensions represent the variables required to specify the phase or state of a system.

1. Aperiodic long-term behaviour means that there are trajectories[2] which do not settle down to fixed points, periodic or quasi- periodic[3] orbits as $t \to \infty$.
2. Deterministic means that the system's irregular behaviour arises solely from the system's nonlinearity. The system has no random or noisy inputs or parameters.
3. Sensitive dependence on initial conditions means that nearby trajectories diverge exponentially fast.

It is easy to find the nonlinear terms "xz" and "xy" that cause the irregular behaviour of the Lorenz equations, but it is not that easy to confirm the other two properties so we will just explain them using the two visualizations below.



(a) Solution of the Lorenz system with starting value $(0, 1, 1)$, using 4th order Runge-Kutta (see 2.2)



(b) Solution of the Lorenz system with starting value $(0, 1, 1)$ in blue and $(0, 1, 1 + 10^{-10})$ in red, using 4th order Runge-Kutta (see 2.2)

The aperiodic behaviour is apparent in (a) and, in (b), we observe that for slightly different values, an error of 10^{-10} in one coordinate to be exact, although the solutions start the same (the blue and red curves form a purple one) their trajectories vary completely after a small amount of time.

These properties are of course just the tip of the iceberg in the analysis of the system, but they are enough for our use. The interested reader is referred to [8] for more information about the Lorenz system and its properties.

1.1.3 Extensions of the Lorenz system and its use in Cryptology

In general, characteristics like the above and more are what makes chaotic systems attractive for cryptography. This is why we can see that the Lorenz system is being used in image-encryption algorithms [1], [2] as well as an extension of it in four dimensions [3]. However, due to the nature of our project, we will not delve deeper into the cryptographic applications of the system.

Considering the extensions of the Lorenz equations in other dimensions, we will simply mention one form of them in 4D, as it can be found on [3] and we will come back to this system in order to solve it in later chapters. This 4D "hyperchaotic[4]" system, is of the form:

[2] Given some initial values for a system, a trajectory is the set of points that the solution of the system passes from, as time progresses.

[3] A quasi-periodic orbit is one that is almost periodic, recurring at irregular intervals.

[4] More information in `http://www.scholarpedia.org/article/Hyperchaos`.

$$\begin{cases} \dot{x} = a(y - x) \\ \dot{y} = cx - xz - y + ew \\ \dot{z} = x^4 + y^4 - bz \\ \dot{w} = -dy, \end{cases}$$

where a, b, c, d, e are the parameters and x, y, z, w are the time-depended state variables of the system. Moreover, in order to achieve characteristics for use in cryptographic encryption, the parameters are to be taken as $a = 10, b = 2.667, c = 46, d = 2, e = 12$.

1.2 Generalized system of first-order differential equations

Returning to our starting goal, we wanted to solve the two systems mentioned in the previous section using numerical methods. These systems are of the form below, for $n = 3$ and $n = 4$, respectively:

$$\begin{cases} \dot{x_1} = f_1(x_1, \dots, x_n) \\ \quad\vdots \\ \dot{x_n} = f_n(x_1, \dots, x_n) \end{cases}$$

Therefore, instead of solving only the two systems above, I decided that going after solving this more general form would prove a much more fruitful goal.

Moreover, considering that I would also have to create a code for the solutions and the visualizations, some rules had to be put in place for this still abstract problem. For example, the time t couldn't go forever in the models and so it was bounded (from now on $t \in [0, T]$ for some T). Additionally, it was not necessary to make a program that would work for every value $n \in \mathbb{N}$, but only for $n = 3$ or 4, because one can easily modify the program for it to work in more than four dimensions. Adding the above together, we state the final and clearer form of the problem to be solved:

* The general problem *

Assuming input of $n = 3$ or 4, $T > 0$ (along with a natural number M that defines the partition of $[0, T]$ into M parts), the functions[a] $f_1, \dots, f_n : \mathbb{R}^n \to \mathbb{R}$ and the initial values $\{x_1(0), \dots, x_n(0)\}$, solve and visualize the solutions of the system

$$\begin{cases} \dot{x_1} = f_1(x_1, \dots, x_n) \\ \quad\vdots \\ \dot{x_n} = f_n(x_1, \dots, x_n), \end{cases}$$

where all state variables $x_1, \dots, x_n$ are depended on the time t with $t \in [0, T]$.

[a]In order to be chaotic, some of these functions should be non-linear, giving birth to chaos.

Chapter 2

Some fundamental numerical methods for solving the problem

The birth of numerical methods for solving differential equations stems from our incapability in solving most of them analytically - something that is true for the equations that interest us too. In more detail, we are interested in numerical methods that can be used to obtain numerical approximations of the solutions of time-dependent ordinary differential equations. These methods can be divided into two main categories:

- Explicit methods calculate the system status at a future time, using the currently known status of the system.
- Implicit methods solve an algebraic equation involving both the current state of the system and the later one in order to calculate the status of the system at a future time.

Denoting $X(t)$ as the current system state and $X(t+\Delta t)$ as the future system state after a relatively small time step Δt, then the above definitions can be written respectively as:

$$\cdot \;\; X(t+\Delta t) = E(X(t))$$
$$\cdot \;\; I\left(X(t), X(t+\Delta t)\right) = 0,$$

where E and I denote the function that is used by the explicit or implicit method, respectively.

More information about the methods and their properties can be found on [9] but the important thing to remember is that different methods do give different results (better or worse) or even no results at all and it is for the same reason that we use more than one method to approach our problem.

More precisely, two explicit methods, Forward Euler and 4th order Runge-Kutta, and one implicit, Backward Euler will be used to solve the problem. In the case of the implicit method, an additional numerical method is needed to solve the underlying algebraic equation, which in our case will be the Newton-Raphson method. Lastly, before we begin the analysis, we set the step-size $h = T/M$, which will be used in the methods and is closely linked to the accuracy of our calculations (larger $M \Rightarrow$ smaller $h \Rightarrow$ more accuracy).

2.1 Forward Euler

In the forward Euler Method our goal is to create a sequence $x_i^0, \ldots, x_i^M$, where $x_i^k \doteqdot x_i(kh)$ with $k = 0, \ldots, M$ (for every $1 \leq i \leq n$), in order to approximate the solution of the system in its discretization points. The method works by using the Taylor expansion $x_i(t_j + h) = x_i(t_j) + h\dot{x}_i(t_j) + \frac{1}{2}h^2\ddot{x}_i(t_j) + \mathcal{O}(h^3)$ for some $t_j \in \{0, \ldots, (M-1)\cdot h\}$, ignoring the quadratic and higher-order terms to get:

$$\begin{aligned}\dot{x}_i(t_j) &\approx \frac{x_i(t_j+h)-x_i(t_j)}{h},\ (\forall i)\\ \Rightarrow x_i(t_j+h) &\approx x_i(t_j) + h\cdot \dot{x}_i(t_j),\ (\forall i)\\ \Rightarrow x_i(t_j+h) &\approx x_i(t_j) + h\cdot f_i(x_1(t_j),\dots,x_n(t_j)),\ (\forall i)\end{aligned}$$

Using the above approximation and the definition of the sequence $\{x_i^k\}_{k=0}^M$ $(\forall i)$, we get the formula below to compute the sequence which approximates the solution:

$$\boxed{\begin{cases} x_1^{k+1} &= x_1^k + h\cdot f_1(x_1^k,\dots,x_n^k)\\ &\vdots\\ x_n^{k+1} &= x_n^k + h\cdot f_n(x_1^k,\dots,x_n^k)\end{cases}}$$

or in matrix form

$$\boxed{X^{k+1} = X^k + h\cdot f([X^k]^T),}$$

with $X^k = [\ x_1^k \cdots x_n^k\ ]^T$ and $f([X^k]^T) = [\ f_1([X^k]^T) \cdots f_n([X^k]^T)\ ]^T$, for $k = 0,\dots,M-1$.

Remark. It is evident that an error is induced at every time-step due to the truncation of the Taylor series, this is referred to as the local truncation error (LTE) of the method. For the forward Euler method, the LTE is $\mathcal{O}(h^2)$. Hence, the method is referred to as a first order technique[1].

2.2 Fourth order Runge-Kutta

Runge-Kutta methods are a class of methods which judiciously uses the information on the "slope" at more than one point to extrapolate the solution to the future time step. In our project, we will show how the 4th order Runge-Kutta can be used in our problem, but we will not elaborate further about its reasoning. The interested reader can look up to [10] for more.

Similarly to the previous method, our goal is to create a sequence $\{x_i^k\}_{k=0}^M$ (for every $1 \leq i \leq n$). To do that with 4th order Runge-Kutta (RK4), we first have to define for every i the following:

$$\begin{aligned} k_1^i &= f_i(x_1^k,\dots,x_n^k)\\ k_2^i &= f_i(x_1^k + h\frac{k_1^i}{2},\dots,x_n^k + h\frac{k_1^i}{2})\\ k_3^i &= f_i(x_1^k + h\frac{k_2^i}{2},\dots,x_n^k + h\frac{k_2^i}{2})\\ k_4^i &= f_i(x_1^k + hk_3^i,\dots,x_n^k + hk_3^i)\end{aligned}$$

Using the above we can construct the formula for RK4:

$$\boxed{\begin{cases} x_1^{k+1} &= x_1^k + \frac{h}{6}\cdot[k_1^1 + 2k_2^1 + 2k_3^1 + k_4^1]\\ &\vdots\\ x_n^{k+1} &= x_n^k + \frac{h}{6}\cdot[k_1^n + 2k_2^n + 2k_3^n + k_4^n]\end{cases}}$$

[1] In general, a method with $\mathcal{O}(h^{s+1})$ LTE is said to be of sth order. Evidently, higher order techniques provide lower LTE for the same step size.

This translates to the formula below in matrix form

$$\boxed{X^{k+1} = X^k + \tfrac{h}{6} \cdot [K_1 + 2K_2 + 2K_3 + K_4]\,,}$$

with $X^k = [\, x_1^k \cdots x_n^k \,]^T$, $f([X^k]^T) = [\, f_1([X^k]^T) \cdots f_n([X^k]^T) \,]^T$
and $K_1 = f([X^k]^T)$, $K_2 = f([X^k]^T + \frac{h}{2}K_1)$, $K_3 = f([X^k]^T + \frac{h}{2}K_2)$, $K_4 = f([X^k]^T + hK_3)$.

Remark. The LTE of the 4th order Runge-Kutta method is $\mathcal{O}(h^5)$ and as such it is much more accurate than the forward Euler method. For this accuracy, it is usually chosen for the solution of problems, such as ours, even though it requires more computations.

2.3 Backward Euler used with Newton-Raphson method

The backward Euler method is similar to the forward Euler with a slight change in the approximation, which in turn makes this method implicit and not explicit. In particular, we will use the following:

$$\begin{aligned} \dot{x}_i(t_j) &\approx \frac{x_i(t_j) - x_i(t_j - h)}{h},\ (\forall i) \\ \Rightarrow x_i(t_j) &\approx x_i(t_j - h) + h \cdot \dot{x}_i(t_j),\ (\forall i) \\ \Rightarrow x_i(t_j) &\approx x_i(t_j - h) + h \cdot f_i(x_1(t_j), \ldots, x_n(t_j)),\ (\forall i) \end{aligned}$$

Using the above approximation and the definition of the sequence $\left\{x_i^k\right\}_{k=0}^{M}$ $(\forall i)$, we get the formula below to compute the sequence:

$$\boxed{\left\{ \begin{array}{lcl} x_1^{k+1} & = & x_1^k + h \cdot f_1(x_1^{k+1}, \ldots, x_n^{k+1}) \\ & \vdots & \\ x_n^{k+1} & = & x_n^k + h \cdot f_n(x_1^{k+1}, \ldots, x_n^{k+1}) \end{array} \right.}$$

or in matrix form

$$\boxed{X^{k+1} = X^k + h \cdot f([X^{k+1}]^T),}$$

It is now apparent that the difference between the methods is the fact that in the implicit backward Euler we have no knowledge of the value of $f([X^{k+1}]^T)$, as the unknown that we are looking for is the X^{k+1}. Therefore, in each step of the method we have to solve an algebraic equation for this unknown. This can be done with methods like the "fixed-point iteration" or the "Newton-Raphson method", depending on the problem. For this problem, we have chosen to continue with the use of the Newton-Raphson method, or more specifically, a generalization of the method called "Generalized Multidimensional Newton", suitable for our multi-dimensional problem.

The first step in Newton is to put everything on one side of each equation and define the functions g_i and g as in the following:

$$\left\{ \begin{array}{lclcl} g_1(x_1^{k+1}, \ldots, x_n^{k+1}) & \doteqdot & x_1^{k+1} - x_1^k - h \cdot f_1(x_1^{k+1}, \ldots, x_n^{k+1}) & = & 0 \\ & \vdots & & & \vdots \\ g_n(x_1^{k+1}, \ldots, x_n^{k+1}) & \doteqdot & x_n^{k+1} - x_n^k - h \cdot f_n(x_1^{k+1}, \ldots, x_n^{k+1}) & = & 0 \end{array} \right.$$

or in matrix form

$$g(X^{k+1}) \doteqdot X^{k+1} - X^k - h \cdot f([X^{k+1}]^T) = 0$$

The Newton-Raphson method will be used in every iteration of the formula above. To showcase how Newton works in each iteration, we fix an index k for a X_{k+1} that we want to find. We also note that $x_1, \ldots, x_n$ are known (found in the previous step of backward Euler) and we continue by making some substitutions to not further complicate ourselves with more indexes, setting $u_1 = x_1^{k+1}, \ldots, u_n = x_n^{k+1}$ and $U = X^{K+1}$, getting the following:

$$\begin{cases} g_1(u_1, \ldots, u_n) & = & 0 \\ & \vdots & \\ g_n(u_1, \ldots, u_n) & = & 0 \end{cases}$$

or in matrix form, $g(U) = 0$.

In Newton, our goal is to find the root $U = X^{k+1}$ of the function g. For this reason, the method starts by using an initial guess about the root of g (which we denote by $U^0 = (u_1^0, \ldots, u_n^0)$) and continue by trying to improve this initial guess of the root, according to the formula below, until we reach a sufficiently precise value

$$\boxed{U^{s+1} = U^s - J^{-1}(U^s) \cdot g(U^s)}$$

where $s = 0, 1, \ldots$ and J^{-1} is the inverse of the Jacobian matrix $J = \begin{pmatrix} \frac{\partial g_1}{\partial u_1} & \cdots & \frac{\partial g_1}{\partial u_n} \\ \vdots & \ddots & \vdots \\ \frac{\partial g_1}{\partial u_1} & \cdots & \frac{\partial g_1}{\partial u_n} \end{pmatrix}$ of g.

Remarks.

(i) The "sufficiently precise value" mentioned above is decided by an appropriate stopping criterion, with a corresponding error. In our implementation of the method, we used the error value $e = \left[\sum_{i=0}^n \left(g_i(u_1^{k+1}, \ldots, u_n^{k+1}) - 0\right)^2\right]^{\frac{1}{2}}$, which is the Euclidean distance of g from zero. In every step the error value e is calculated and the method stops when e falls below our chosen stop value (10^{-8} in our program).

(ii) In real applications, for reasons of stability and time consumption, instead of calculating the inverse of the Jacobian matrix J, we prefer to solve the matrix equation $J \cdot x = g(U^s)$, for each s.

(iii) Considering the convergence of the Newton method towards the root, we simply note that it cannot be guaranteed for all problems and that it also depends a lot on the choice of the initial value U^0. More information about all of this can be found on [11] and its references.

Chapter 3

Applications in chaotic systems

Using the above theory we have constructed three python functions that output the approximate solutions of the system inputted, named FE_ function, RK4_ function and BE_ function, respectively for forward Euler, 4th order Runge-Kutta and backward Euler. In each one, the parameters T and M, which are giving us the step-size $h = T/M$, are also part of the inputs. Moreover, the smaller this h becomes, the more accurate the approximation gets too.

In this final chapter, we will use these functions and the python module "matplotlib" to create some visualizations for the solutions of some first-order systems with chaotic properties. The main use of these visualizations on this project will be first, to showcase the power of our program and second, to compare the methods. For this reason, in each system, we will be using the same $h = T/M$ in each method, if we can.

Lastly, before we begin, the terms "attractor" and "strange attractor" should be explained, as we will need them later on. Loosely, an attractor is a set of points to which all neighbouring trajectories converge, whereas a strange attractor is one that also exhibits sensitive dependence on initial conditions. More precisely, an attractor is a closed set A with the following properties:

1. A is an invariant set:
   This means that any trajectory $x(t)$ that starts in A stays in A for all time.
2. A attracts an open set of initial conditions:
   This signifies that there exists an open set U containing A such that if $x_0 \in U$, then the distance from $x(t)$ to A tends to zero as time t progresses to infinity. In other words, A attracts all trajectories that start sufficiently close to it.
3. A is minimal:
   This denotes that there is no proper subset of A that satisfies the previous two conditions.

3.1 Applications in the Lorenz chaotic system and its extensions

We choose to start our comparisons with our main goal, the chaotic Lorenz system and its extension, the 4d hyperchaotic Lorenz system. As a general remark, we note that the explicit methods FE and RK4 seem to be the best approximations, with RK4 being the best, while the implicit BE method trails the furthest from the truth, giving non-chaotic solutions! In fact, our results seem to coincide with two other papers ([12] and [13]) which have also noted this strange behaviour when using this method, as well as other implicit ones.

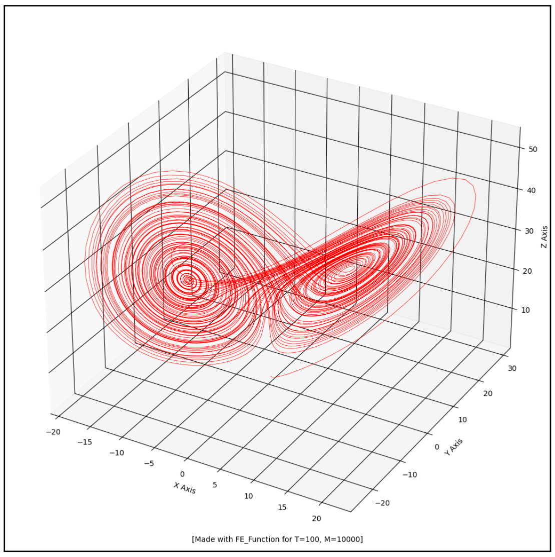
3.1.1 Classic Lorenz chaotic system

First of all, we remind again the form of the chaotic Lorenz system:

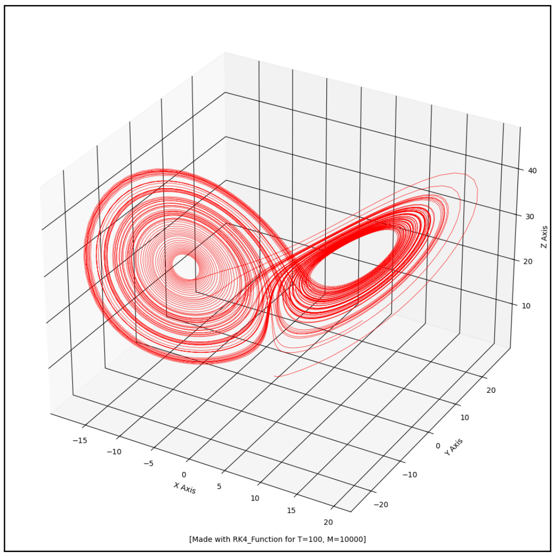
$$\begin{cases} \dot{x} = \sigma(y - x) \\ \dot{y} = x(\rho - z) - y \\ \dot{z} = xy - \beta z, \end{cases}$$

with $\sigma = 10,\ \rho = 28$ and $\beta = 2.667$.

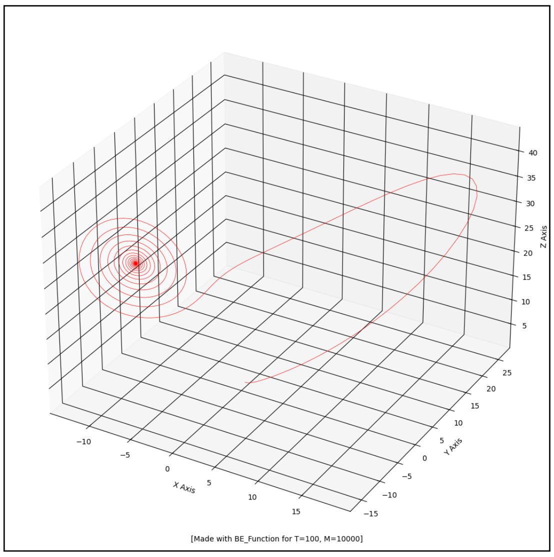
Seeing the visualizations, the first thing we note is the whole of the Lorenz strange attractor, which has two fixed points that constantly repel the trajectory from one to the other. Moreover, we can see that the trajectory of the solution comes closer to the fixed points in the FE case, if we compare it to the RK4 one. We attribute this to the fact that RK4 has smaller error in general. Moreover, if we were to keep T constant while choosing a larger M, we would see that the FE case would look more like the RK4 case as M increases (and therefore, the step-size h, which is related to the error, decreases). Finally, we see that the BE case is completely different from the other two and the trajectory of the solution is attracted to the fixed point, losing all chaotic properties, a strange behaviour that could be analysed further in future projects.



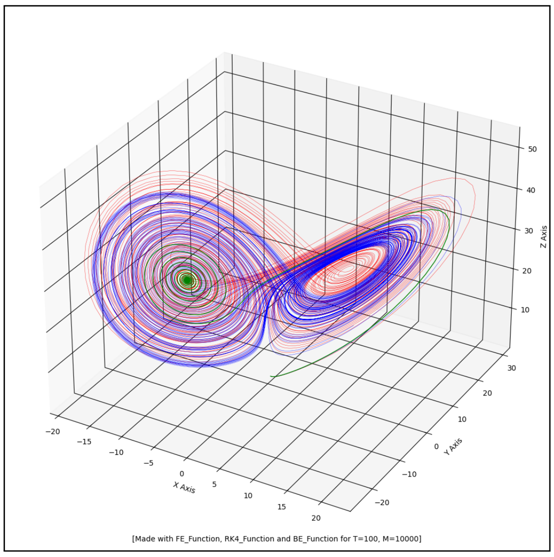
(a) Lorenz system with initial values $(0, 1, 1)$ using FE, for $h = 0.01$



(b) Lorenz system with initial values $(0, 1, 1)$ using RK4, for $h = 0.01$



(c) Lorenz system with initial values $(0, 1, 1)$ using BE, for $h = 0.01$



(d) Lorenz system with initial values $(0, 1, 1)$ using FE, RK4 and BE, for $h = 0.01$

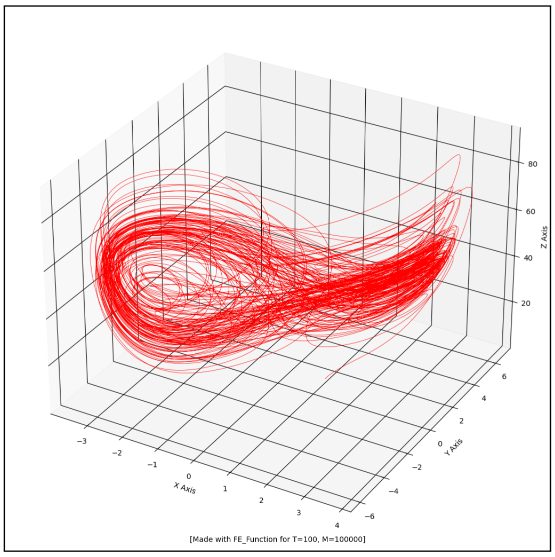
3.1.2 4D hyperchaotic Lorenz system

We remind again the form of the 4d hyperchaotic Lorenz system:

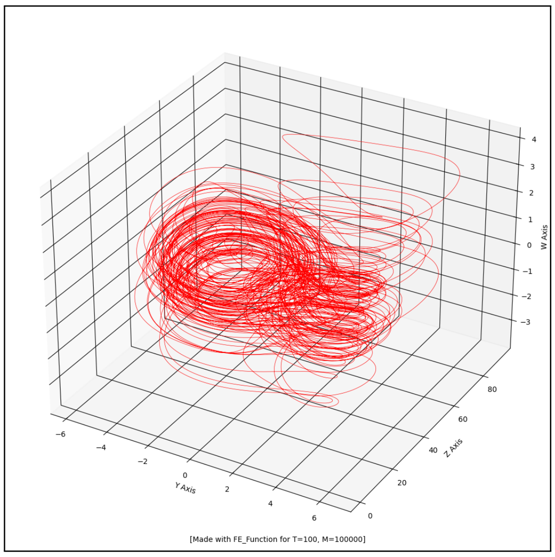
$$\begin{cases} \dot{x} = a(y-x) \\ \dot{y} = cx - xz - y + ew \\ \dot{z} = x^4 + y^4 - bz \\ \dot{w} = -dy, \end{cases}$$

where $a = 10$, $b = 2.667$, $c = 46$, $d = 2$ and $e = 12$.

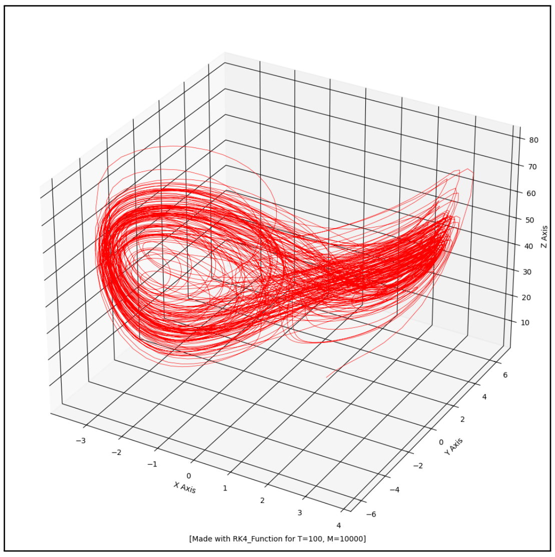
In this case, we can tell that the system resembles that of the classic Lorenz and all our previous comments also hold. Firstly, the trajectory seems to be again repelled by fixed points. Moreover, the trajectory in the RK4 case seems "cleaner", staying further away from the fixed points, if we compare it to the FE case, possibly due to its larger error. Finally, we observe again that the BE method completely breaks down the chaotic nature of the system and is following these strange trajectories near the fixed points. A fascinating result really that should also be further researched!



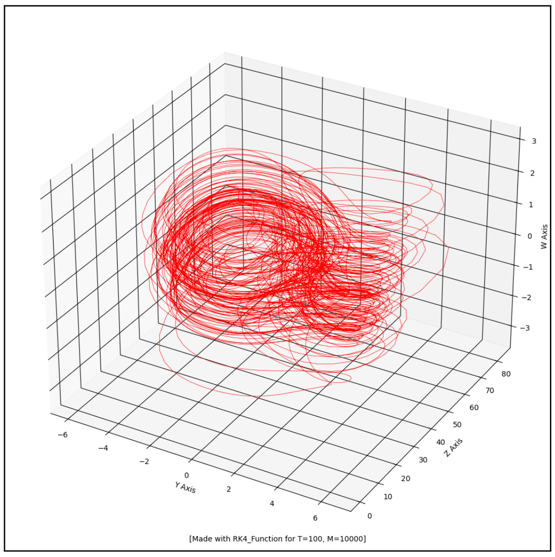
(e) xyz-plane of 4d Lorenz system with initial values $(1, 1, 1, 1)$ using FE, $h = 0.001$



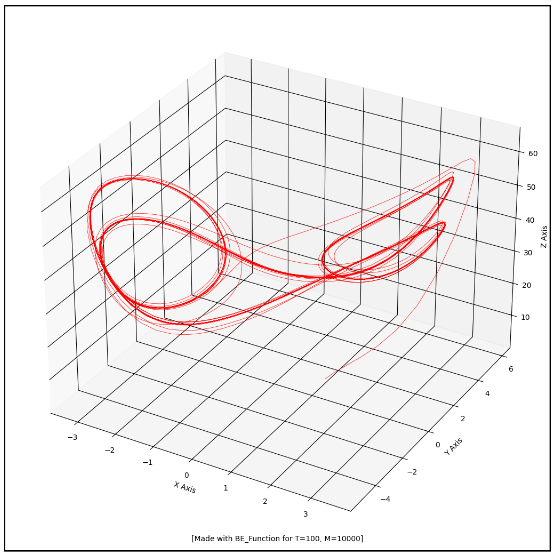
(f) yzw-plane of 4d Lorenz system with initial values $(1, 1, 1, 1)$ using FE, $h = 0.001$



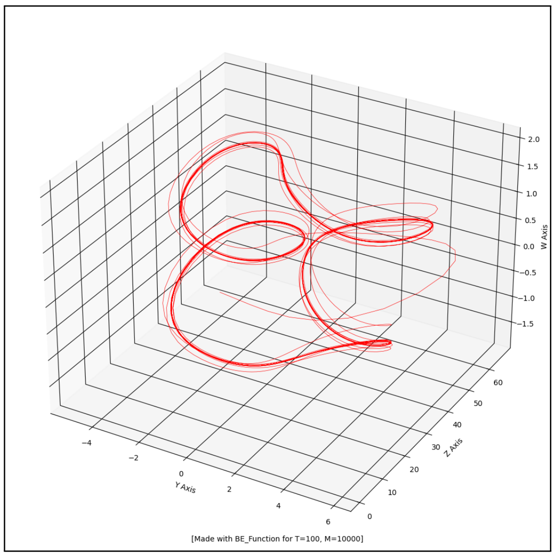
(g) xyz-plane of 4d Lorenz system with initial values $(1, 1, 1, 1)$ using RK4, $h = 0.01$



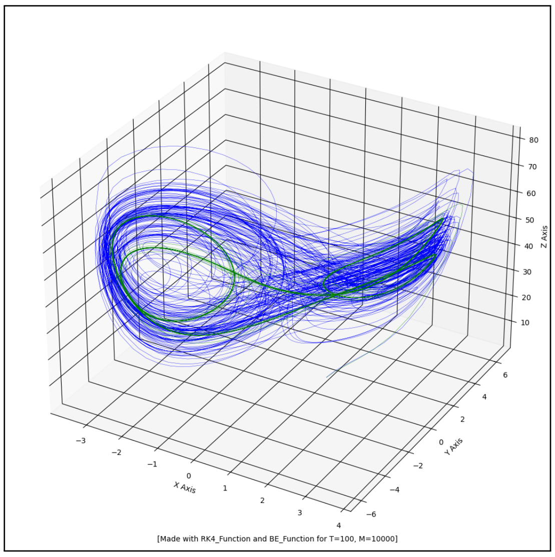
(h) yzw-plane of 4d Lorenz system with initial values $(1, 1, 1, 1)$ using RK4, $h = 0.01$



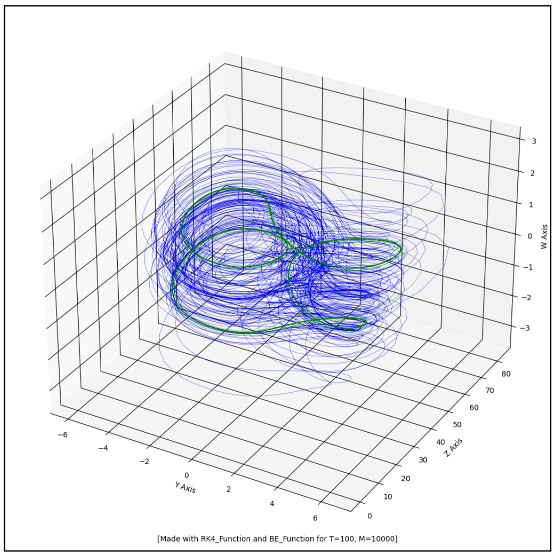
(i) xyz-plane of 4d Lorenz system with initial values $(1, 1, 1, 1)$ using BE, $h = 0.01$



(j) yzw-plane of 4d Lorenz system with initial values $(1, 1, 1, 1)$ using BE, $h = 0.01$



(k) xyz-plane of the 4d Lorenz system with initial values $(1, 1, 1, 1)$ using RK4 and BE, for $h = 0.01$



(l) yzw-plane of the 4d Lorenz system with initial values $(1, 1, 1, 1)$ using RK4 and BE, for $h = 0.01$

Remark. Though we try to maintain the same step-size for all methods, in this system, in the FE case, the program could not work properly given $h = 0.01$ and so we had to make it smaller ($h = 0.001$). This smaller step-size, which gives a better approximation of the solution, is probably why the FE case looks more similar to the RK4 method in this system than it was in the classic case.

3.2 Descending further into chaos

In the final section of the project, we shall visualize the behaviour of some other chaotic first-order differential systems, namely the Chen system, the Nose-Hoover oscillator and the Sprott-Jafari system. The main reason for this "extra" research was to further analyse the behaviour of the BE method in different systems and see if it gives us strange results in these systems too. An answer to this question, positive or negative, could ignite further questions and research. If it does follow this strange pattern, why does it do it? i.e. why does this method does not work well in these systems? Similarly, if it doesn't exhibit strange behaviour in these cases then, why is the Lorenz system special? is it the only one? and why?

Therefore, due to all the above, the systems will not be analysed in terms of their chaotic nature and respective properties. Instead, we care only about how the numerical methods, represented by our python functions, perform in each case. For further analysis of the systems, we refer the interested reader to the citations placed in each subsection.

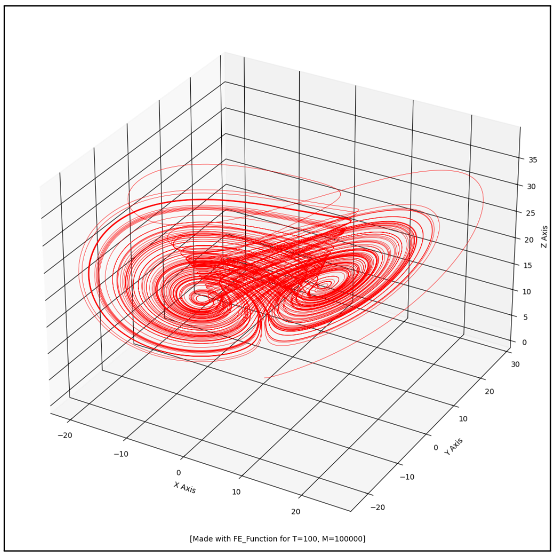
3.2.1 The Chen system

We first start with the Chen system, for which further information can be found here [14], [15]. Its form is:

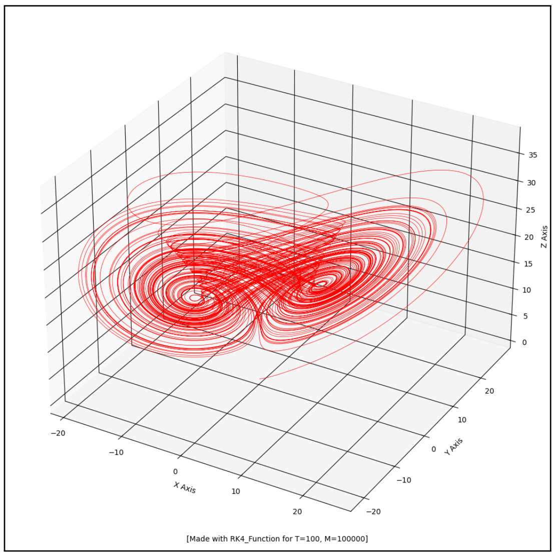
$$\begin{cases} \dot{x} = a(y-x) \\ \dot{y} = (c-a)x - xz + cy \\ \dot{z} = xy - bz, \end{cases}$$

with $a = 40,\ b = 3$ and $c = 28$.

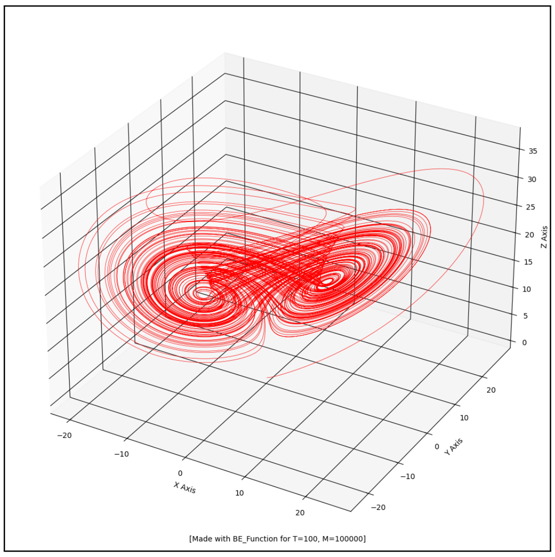
Looking at the visualizations, the first thing we note is that all methods give us more or less the same solutions. Hence, in this system, the BE method is able to perform well. Moreover, about the system itself, we see that in this time too, there are two fixed points that repel the trajectory from one to the other.



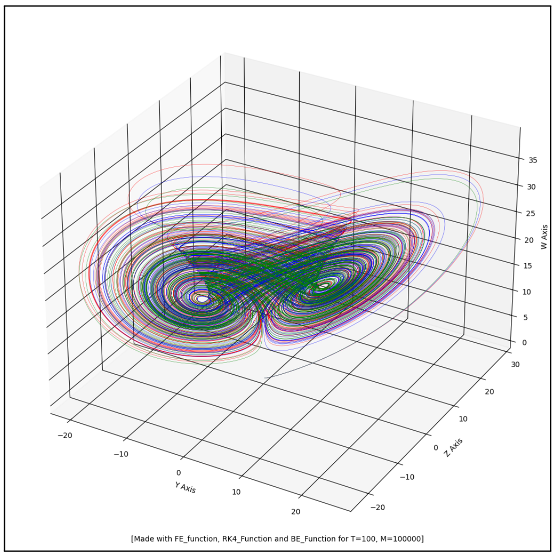
(m) Chen system with initial values $(-0.1, 0.5, -0.6)$ using FE, for $h = 0.01$



(n) Chen system with initial values $(-0.1, 0.5, -0.6)$ using RK4, for $h = 0.01$



(o) Chen system with initial values $(-0.1, 0.5, -0.6)$ using BE, for $h = 0.01$



(p) Chen system with initial values $(-0.1, 0.5, -0.6)$ using FE, RK4 and BE, for $h = 0.01$

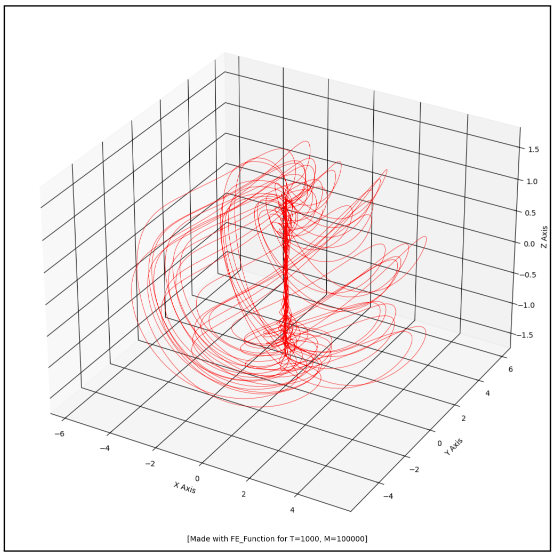
3.2.2 The Nose-Hoover oscillator

The second system stems from [16] and it is of the form:

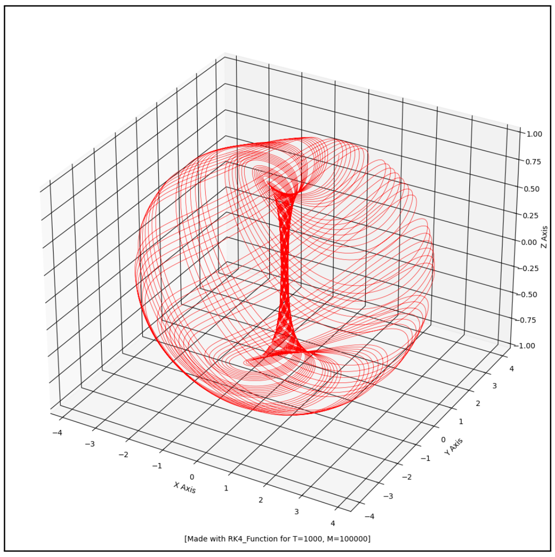
$$\begin{cases} \dot{x} = y \\ \dot{y} = -x - yz \\ \dot{z} = (y^2 - 1)/q, \end{cases}$$

with $q = 10$.[1]

Looking at the visualizations, it is apparent that this time the trajectory of the system is totally different from before. However, instead of delving on the system itself, our goal is the comparison of the methods.



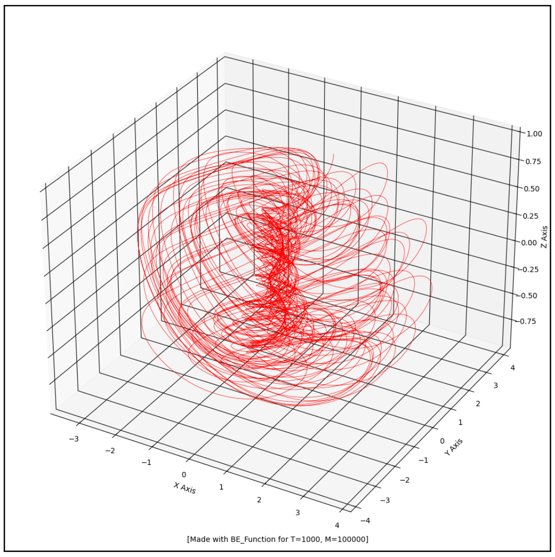
(q) Nose-Hoover oscillator with initial values $(1, 1, 1)$ using FE, for $h = 0.01$



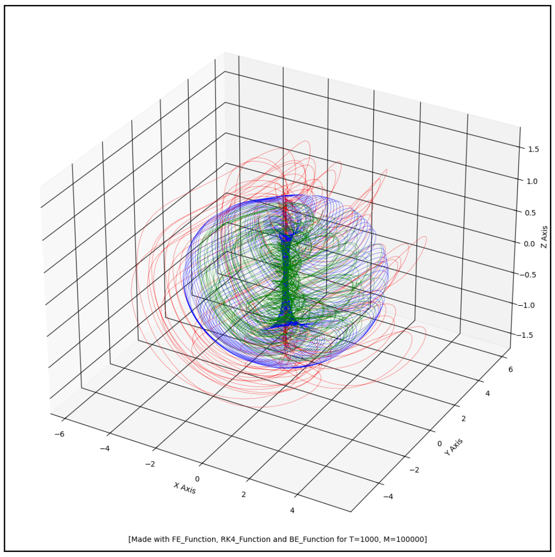
(r) Nose-Hoover oscillator with initial values $(1, 1, 1)$ using RK4, for $h = 0.01$

[1]After doing some tests with the value q, the value $q = 10$ was decided after seeing the amazing visual on the RK4 case.

We note that again the three methods appear to tell the same story, but with much less accuracy in the cases of FE and BE. If we were to use a different a smaller step-size h in these methods, we would see that they would look more like the RK4 case, as the error value would drop.



(s) Nose-Hoover oscillator with initial values $(1, 1, 1)$ using BE, for $h = 0.01$



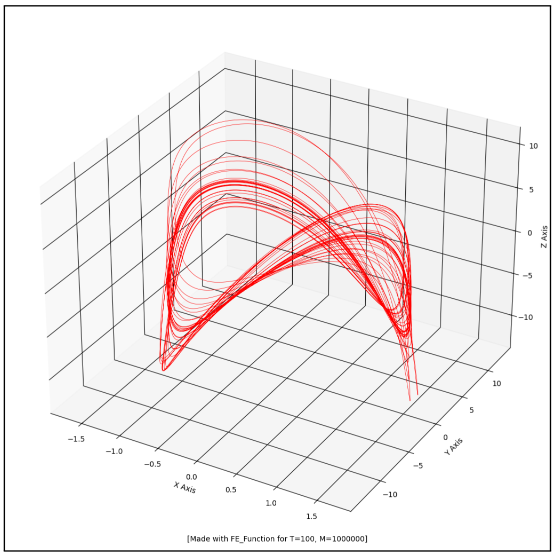
(t) Nose-Hoover oscillator with initial values $(1, 1, 1)$ using FE, RK4 and BE, for $h = 0.01$

3.2.3 The Sprott-Jafari system

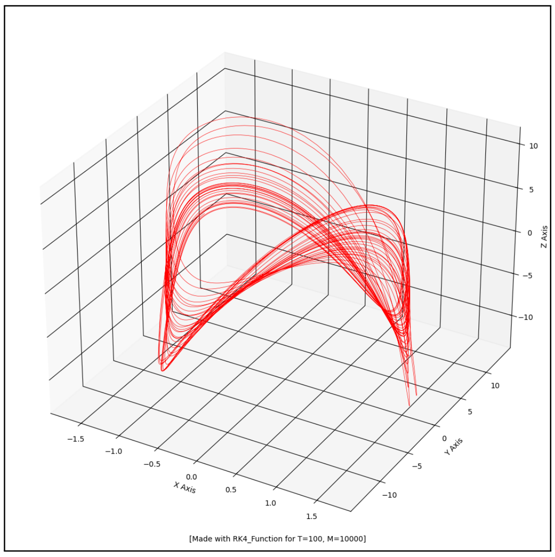
Lastly, more on our final system can be found here [17]. Its form is:

$$\begin{cases} \dot{x} = y \\ \dot{y} = -x + yz \\ \dot{z} = z + ax^2 - y^2 - b, \end{cases}$$

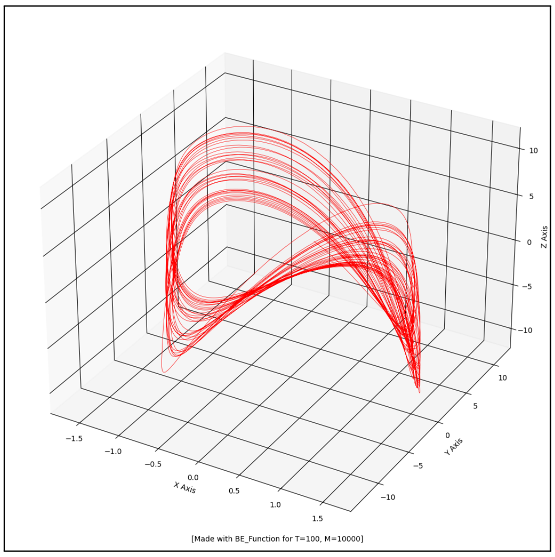
with $a = 8.894$ and $b = 4$.



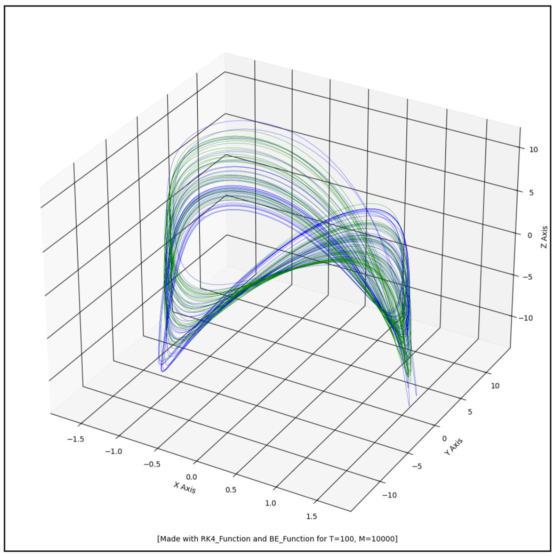
(u) Sprott-Jafari system with initial values $(0, 3.8, 0.7)$ using FE, for $h = 0.0001$



(v) Sprott-Jafari system with initial values $(0, 3.8, 0.7)$ using FE, for $h = 0.01$



(w) Sprott-Jafari system with initial values $(0, 3.8, 0.7)$ using FE, for $h = 0.01$



(x) Sprott-Jafari system with initial values $(0, 3.8, 0.7)$ using RK4 and BE, for $h = 0.01$

In the last system, at first glance, it is apparent that the methods give us similar results but, in reality, this is only true for the RK4 and BE cases. Particularly, as it was the case in the hyperchaotic 4d Lorenz system, FE_function cannot work properly for $h = 0.01$ and not even for $h = 0.0001$. Hence, the visual for the FE case was done with $h = 0.001$ and this is why it is so similar to the other too.

3.3 Conclusion and Future prospects

The purpose of this project was to find and visualise the solutions of the chaotic Lorenz system and the hyperchaotic 4d Lorenz system. Having this aim in mind, an opportunity to explore something more broad (and more interesting) was seen. Hence, the generalized problem of solving "all" first-order differential equations was stated and tackled. This happened with three different methods and three different programs, each accompanying one of them. Lastly, with the help of these programs, not only the Lorenz systems but others as well were solved, visualized giving a new insight not only into the systems, but the methods themselves.

Concerning future research, our research and results have indeed created more questions than answers. As we have seen, the third method "Backward Euler used with Raphson-Newton", an implicit method, did not agree with the other two, explicit methods, (only) in the Lorenz systems, giving results that were not even chaotic. These results have been observed by other researchers too, as we can see in [12] and [13], and have been observed when using other implicit methods too when solving the Lorenz system.

In the end, all the above gives birth to more questions. Why do implicit methods do not work for this system? Does it happen for all implicit methods? As Lorenz is probably not the only system with this feature, can we define a class of chaotic systems that, like Lorenz, can be reduced from their chaotic status, when using implicit methods?

References

[1] Temadher Alassiry Al-Maadeed et al. “An image encryption algorithm based on chaotic Lorenz system and novel primitive polynomial S-boxes”. In: *CoRR* abs/2006.11847 (2020). arXiv: `2006.11847`. URL: `https://arxiv.org/abs/2006.11847`.

[2] Kayhan Çelik and Erol Kurt. “A new image encryption algorithm based on lorenz system”. In: June 2016, pp. 1–6. DOI: `10.1109/ECAI.2016.7861097`.

[3] Avishek Kumar. “Image Encryption using Four-Dimensional Hyper Chaotic Lorenz System”. In: *Elixir International Journal* 97 (July 2016), pp. 41904–41909.

[4] Edward N. Lorenz. “Deterministic Nonperiodic Flow”. In: *Journal of Atmospheric Sciences* 20.2 (1963), pp. 130–141. DOI: `10.1175/1520-0469(1963)020<0130:DNF>2.0.CO;2`. URL: `https://journals.ametsoc.org/view/journals/atsc/20/2/1520-0469_1963_020_0130_dnf_2_0_co_2.xml` (visited on 04/01/2023).

[5] Barry Saltzman. “Finite Amplitude Free Convection as an Initial Value Problem—I”. In: *Journal of Atmospheric Sciences* 19.4 (1962), pp. 329–341. DOI: `10.1175/1520-0469(1962)019<0329:FAFCAA>2.0.CO;2`. URL: `https://journals.ametsoc.org/view/journals/atsc/19/4/1520-0469_1962_019_0329_fafcaa_2_0_co_2.xml` (visited on 04/01/2023).

[6] Collin Sparrow. *The Lorenz Equations: Bifurcations, Chaos and Strange Attractors*. Springer New York, 2012. DOI: `https://doi.org/10.1007/978-1-4612-5767-7`.

[7] S.H. Strogatz. *Nonlinear Dynamics and Chaos: With Application to Physics, Biology, Chemistry and Engineering*. Perseus Books Publishing, Cambridge, 1994.

[8] James Hateley. *The Lorenz system (Lecture Notes)*. URL: `https://web.math.ucsb.edu/~jhateley/research.html` (visited on 04/01/2023).

[9] K. Atkinson, W. Han, and D.E. Stewart. *Numerical Solution of Ordinary Differential Equations*. Pure and Applied Mathematics: A Wiley Series of Texts, Monographs and Tracts. Wiley, 2011. ISBN: 9781118164525. URL: `https://books.google.gr/books?id=QzjGgLlKCYQC`.

[10] Julyan H. E. Cartwright and Oreste Piro. “The Dynamics of Runge-Kutta Methods”. In: *International Journal of Bifurcation and Chaos* 02 (1992), pp. 427–449.

[11] Boris Polyak. “Newton’s method and its use in optimization”. In: *European Journal of Operational Research* 181 (Sept. 2007), pp. 1086–1096. DOI: `10.1016/j.ejor.2005.06.076`.

[12] Xuan Chen. “Numerical Counterexamples of Lorenz System in Implicit Time Scheme”. In: (2018). DOI: `10.20944/preprints201801.0277.v1`.

[13] Noorhelyna Razali and R. Ahmad. “Solving Lorenz system by using Runge-Kutta method”. In: 32 (Jan. 2009), pp. 241–251.

[14] G. Chen and T. Ueta. “Yet Another Chaotic Attractor”. In: *International Journal of Bifurcation and Chaos* 9 (7 1999), pp. 1465–1466. DOI: `10.1142/S0218127499001024`.

[15] G. Chen and T. Ueta. “Bifurcation Analysis of Chen’s Attractor”. In: *International Journal of Bifurcation and Chaos* 10 (8 2000), pp. 1917–1931. DOI: 10.1142/S0218127400001183.

[16] William G. Hoover. “Canonical dynamics: Equilibrium phase-space distributions”. In: *Physical Review A* 31 (3 1985), p. 1695. DOI: 10.1103/PhysRevA.31.1695.

[17] S. Jafari, J. C. Sprott, and F. Nazarimehr. “Recent new examples of hidden attractors”. In: *The European Physical Journal Special Topics* 224 (8 2015), pp. 1469–1476. DOI: 10.1140/epjst/e2015-02472-1.