Assignment

Lorentz equations 1

Name TN 1: Tim Peinkofer

tim.peinkofer@student.uni-tuebingen.de

Name TN 2: Fabian Kostow

fabian.kostow@student.uni-tuebingen.de

Tutor: Jose Carlos Olvera Meneses

Date: February 4, 2025

Contents

1 Vector form rk4

First we need to define a System of Equations in one function.

```
def Lorenz_Attractor (r):
    sigma = 10
    R = 28
    beta = 8 / 3
    x, y, z = r
    Xprime = sigma * (y - x)
    yprime = x * (R - z) - y
    Zprime = x * y - beta * z

    return np.array([Xprime, yprime, Zprime])
```

where we get a vector back.
Now we can generalise our Runge-Kutta method

```
    def rk4(System, y0, t, h):
    n = len(t)
    y = np.zeros((n, len(y0)))
    y[0] = y0

    for i in range(1, n):
        k1 = System( y[i - 1])
        k2 = System( y[i - 1] + h * k1 / 2)
        k3 = System( y[i - 1] + h * k2 / 2)
        k4 = System( y[i - 1] + h * k3)
        y[i] = y[i - 1] + h * (k1 + 2 * k2 + 2 * k3 + k4) / 6
    return y
```

It looks similar to the previous rk4, functions instead that the k's are now vectors and our solution is a matrix (a list of all the solution vectors).

2 Question 1

With this setup, we can solve the System and plot the solution over the intervall t = [0,50] with initial conditions $\vec{r}(0) = (5, 5, 5)$.

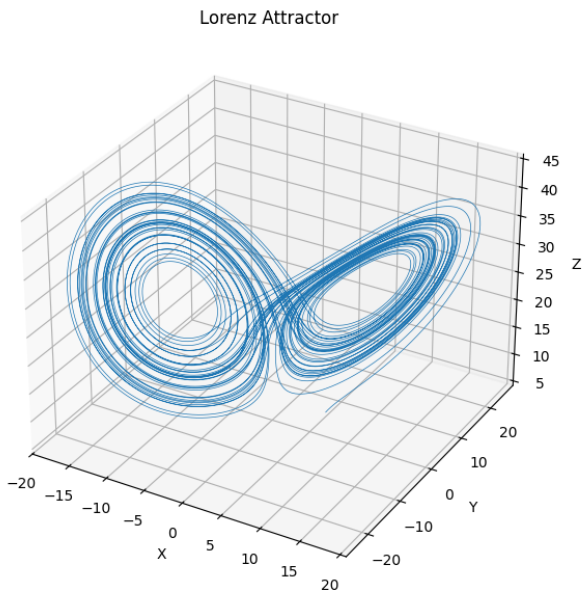

Figure 1: Lorenz Attractor

3 Question 2 a

Now we want to check the distance between two solutions with different, but close to each, initial conditions. Therefore we need to calculate the euklidean distance between two points at each time step.

```
def Distance(y0,y1,t,h):
r0 = rk4(Lorenz_Attractor, y0, t, h)
r1 = rk4(Lorenz_Attractor, y1, t, h)
distance = np.zeros(len(r0))

for i in range(len(r0)):
    deltax = r0[i][0] - r1[i][0]
    deltay = r0[i][1] - r1[i][1]
    deltaz = r0[i][2] - r1[i][2]
    distance[i] = np.sqrt(deltax**2 + deltay**2 + deltaz
                                    **2)
return distance
```

Now we plot the distance on a log scale.

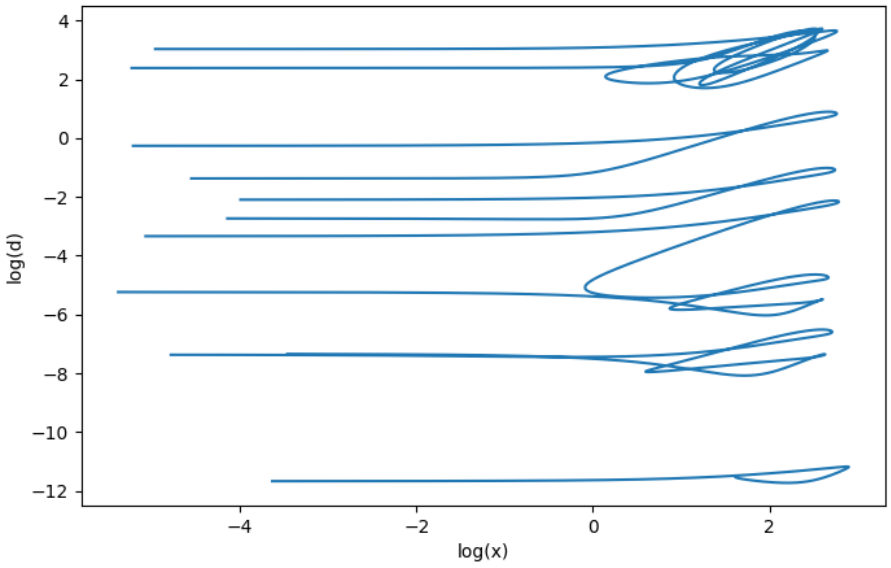

Figure 2: Logarithmic distance

4 Question 2 b

Here we want to use the same initial conditions but different step sizes and check how the solution changes.

$$h_1 = 10^{-6}$$
$$h_2 = 5 \cdot 10^{-4}$$

The difference between the solutions is

$$d(r_{h1}(20), r_{h2}(20)) = 7.649533430479264$$

5 Question 2 c

Now we want to calculate the difference between two solutions with really close initial conditions, because of the question above we can guess that the error will be in the same order of magnitude.

$$y_0 = \begin{pmatrix} 5 \\ 5 \\ 5 \end{pmatrix}$$

$$y_1 = \begin{pmatrix} 5 \\ 5 \\ 5 + 5 \cdot 10^{-15} \end{pmatrix}$$

for 50 time steps and check the distance between the solutions, we get:

$$d(r_0(50), r_1(50)) = 8.764083314737304$$