Jeremy Palmerio
I6239656
PRA3021

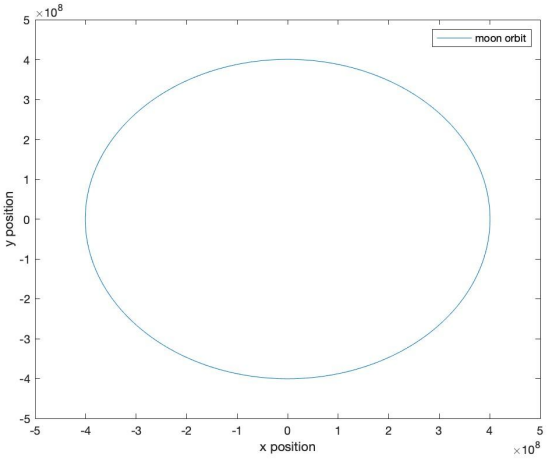
Topics in Scientific Computing Week 4

Introduction

This week's tasks were very fun for me personally as I am very fascinated with orbital mechanics and rocket science, albeit my practical skills and knowledge on these subjects is quite limited. Using matlab to solve differential equations however, I am quite familiar with. Both with the Differential Equations course at MSP and the Mathematical Modeling course from DACS I have been confronted with the built-in ode solvers, with state space representation, with transfer functions etc. In the end I much enjoyed the content of this week as it allowed me to recall some skills and refine them even further. The first 4 exercises were done in class next to Sophia Widmer, so there is a high probability that we have similar codes, however the rest of the exercises were written alone although based on pseudo code we had discussed together.

Exercise 1 - Moon Position

This function is quite basic and mostly uses math to derive a time parameterisation of the angular distance of the moon. There was very little difficulty here, it was quite straightforward. The obtained graph displays a circular orbit around the origin (where earth is supposed to be) with a radius of $d_{em} = 400.5 \times 106\, m$, see Figure 1. Notice the elliptical orbit is due to non normalized x and y axes.



Exercise 2 - Spaceship Acceleration

This exercise consisted of computing the acceleration due to gravitational forces on the spaceship exerted by the Moon and the Earth. As the formula was already derived, there was very little space to mess up. Overall the function takes the position of the moon and the space

ship and based on the formula from the notes it outputs a 2x1 vector of the gravitational acceleration in x and y.

Exercise 3 - State Space

State space representation always evaded me and appeared quite esoteric until I realized its utility with this function. It allows to store all relevant dynamic information of a system to perform simple matrix calculations to study the evolution of the state. In another class we had to apply an impulse input and look at the response of the system. I didn't do it in this case as I didn't see the utility but it seems interesting to unpack the meaning behind it. This task took me longer to understand as I was overcomplicating things in my head. In the end the point is to input the current state vector of the spaceship (the x and y position and the x and y instantaneous velocity) and the output the derivative of the state vector (i.e. the state matrix).

Exercise 4 - Matlab Methods

As I was already familiar with using ode45 to solve differential equations, this task was no problem. One difficulty I encountered was the long processing time when trying to solve the equations over a whole lunar month. I thus opted to create a time vector with time step 100s to make the rendering faster. Moreover, I was confronted with difficulties in plotting. Indeed, I still have yet to fully grasp the nuances of the hold on/off function which meant a lot of plots were mixed up or were plotted multiple times. This is something I want to explore and work on further.

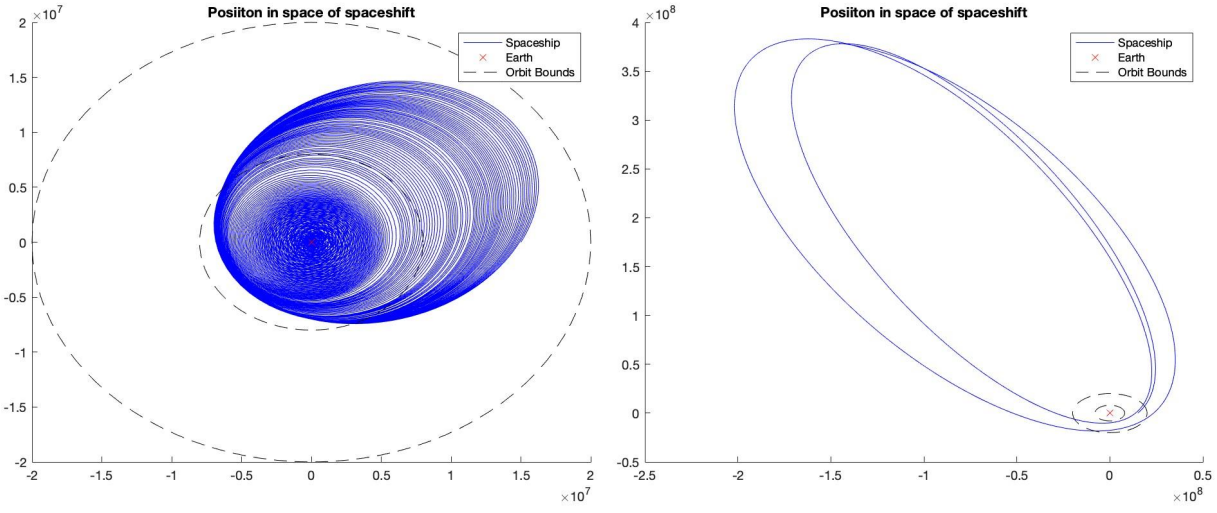


Figure 2: Testing ode45 solver to compute spaceship position over time vector $0\ to\ 2358730s$ (a lunar month) with different initial velocity and same initial position.

Exercise 5 - Euler's Method

Euler's method was presented with pseudo code and was thus easy to get up and running. However, there were some problems regarding the dimension of the time vector which confused me and it was all the result of a typo. Overall, I compared the euler method with ode45 solver for different tasks and obtained various results. When using the euler method to plot the orbit of the spacecraft with the same initial parameters as the second orbit form exercise 4 I noticed dramatically different results (see figure 3). However, when testing the euler method on a simple DE $\frac{dy}{dt} = -y$ which should yield a negative exponential. I noticed the quality of the result was very dependent on the step size of the time vector. Indeed the smaller the time step, the more accurate the euler method which makes sense. This might also explain why the orbit was significantly different using the euler method, because the step size was 100.

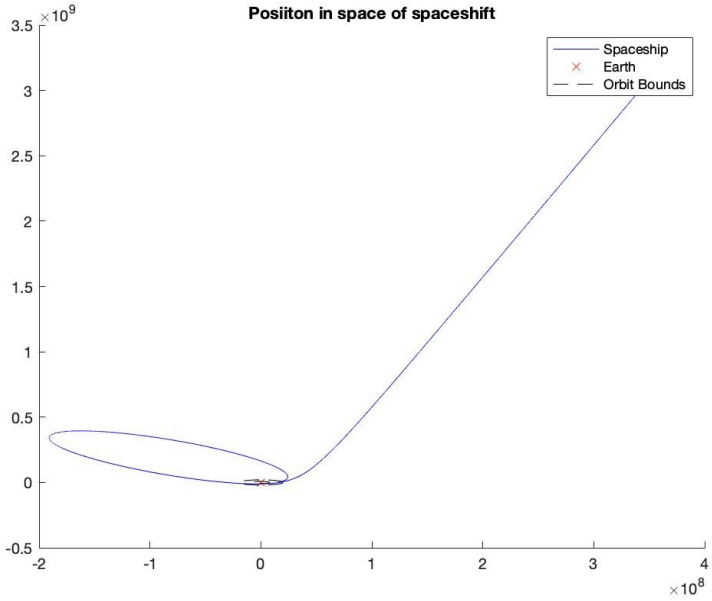


Figure 3: Orbit calculated using the euler method with same initial parameters as Figure 2b.

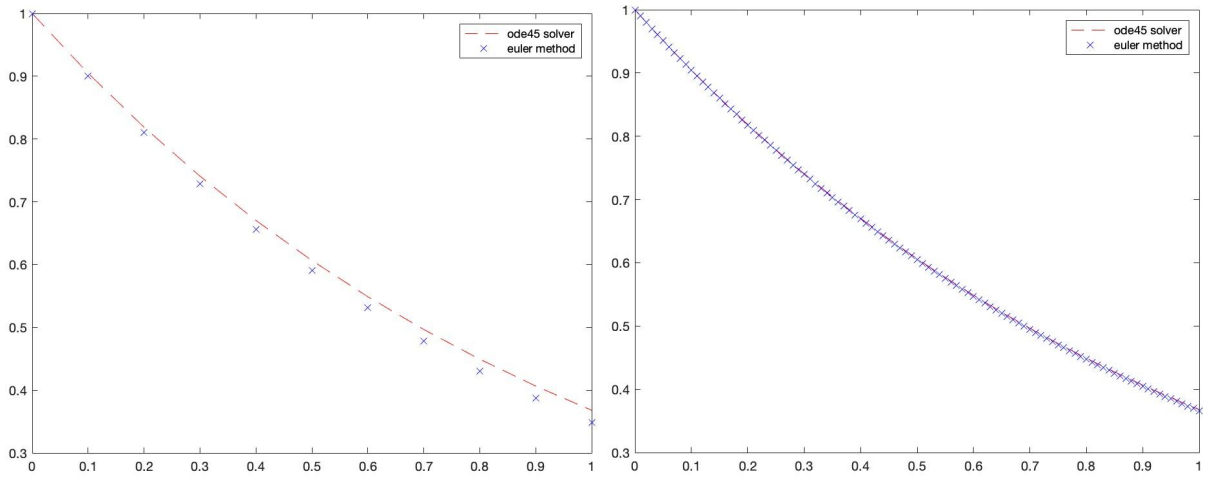


Figure 4: Comparison between ode45 and euler method for step size 0.1 in (a) and 0.01 in (b).

Exercise 6 - Higher Order Methods

For the higher order methods I chose to try the 2nd order euler method and the Heun method. Both of these methods are heavily inspired from the original euler method except that inside the loop there are more in between steps to be calculated. For both, the complexity resided in translating the series into a matlab loop which once the concept is understood, applies to all other methods. The idea is to start by initializing an empty array of the same size as the provided time vector. Then for each element in the time vector, we calculate the in between steps and then stitch them together in the final y matrix which is then outputed with the time vector. The higher order methods provide similar results to the ode45, especially the kutta as that is the algorithm ode45 uses I believe. To quantify their result, I put them through the same test as Euler's method. As noticeable in figure 4, the heun method performs great even at high step sizes, as opposed to euler's method. The same is true for euler 2nd order method compared to euler's 1st order method.

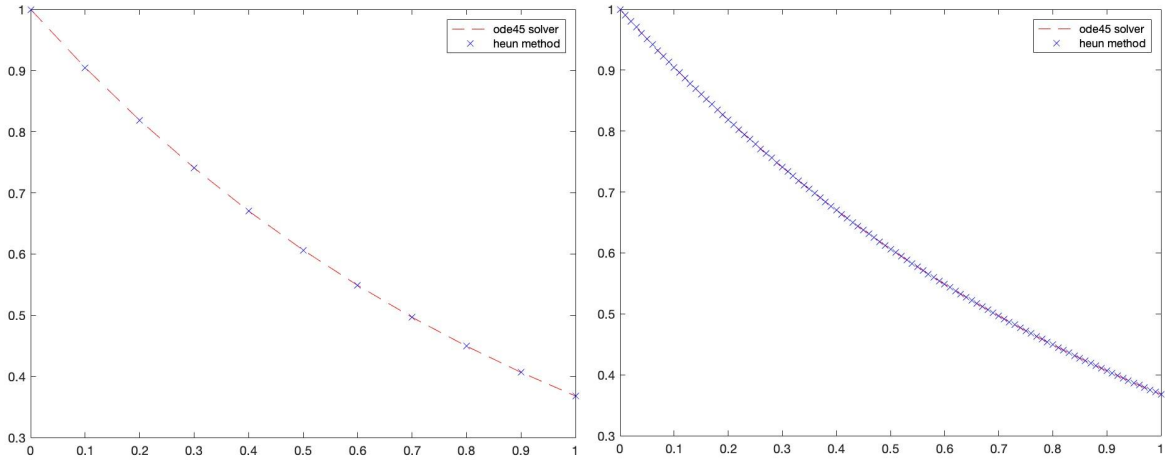


Figure 5 : Comparing Heun method with ode45 with step 0.1 in (&) and step 0.01 in (b)

Exercise 7 - Mission to the Moon

In this exercise the goal is to establish control of the spaceship. In the context of dynamical systems control is at the core of control theory, however this is not the meaning I took for this task. I attempted to create a function thrust(t, v) to add to the acceleration in the state space representation which I can program to thrust accordingly at time t and based on the current speed. Unfortunately I have not had enough time to experiment with this function and I believe I lack some orbital mechanics skills, or maybe I'm diving in the wrong direction. Either way I was able to change the orbit whenever I wanted but I haven't fully grasped how to do this in a controlled and autonomous manner.I managed to find certain states that led to an orbit around the moon or the earth, but I couldn't figure out how to automate it. You can see my attempt in the thrust function.

Exercise 8 - Application

As I did not succeed in controlling the spacecraft after several days of trying, I decided to give up and complete this task instead. This was infinitely easier for me as it only entailed solving a system of differential equations using state space representation. As I have been presented many times with the typical SIR and Lotka Volterra models I chose something different which also fascinates me: chaos theory. Based on the simple model of the lorenz attractor, I found a set of differential equations from wikipedia[1], see figure 6. This system is incredibly sensitive to initial conditions so I looked up which ones would produce a nice system. To see the specifics see the 'lorentzattractor' function. To solve the system I used my kutta method function I also wrote. The result (Figure 7) shows the famous Lorentz attractor plotted in 3D. This result shows the chaotic behavior of this system as it oscillates between two stable equilibrium states seemingly randomly. According to wikipedia this system can be used to model lasers, dynamos, chemical reactions, osmosis etc..[1]

$$\frac{\mathrm{d}x}{\mathrm{d}t} = \sigma(y - x),$$

$$\frac{\mathrm{d}y}{\mathrm{d}t} = x(\rho - z) - y,$$

$$\frac{\mathrm{d}z}{\mathrm{d}t} = xy - \beta z.$$

Figure 6 : The Lorentz system which models atmospheric convection

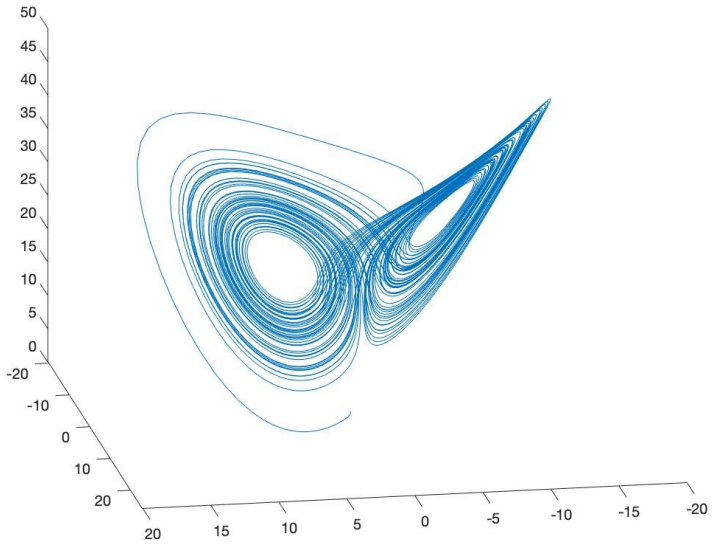


Figure 7: the Lorentz attractor solved by the kutta method in matlab with parameters $\beta = 1.9$, $\sigma = 11$, $\rho = 27$

1 - https://en.wikipedia.org/wiki/Lorenz_system