# PHYC30170 Physics with Astronomy and Space Science Lab 1; The Brusselator - A Computational Example of Chemical Oscillations

Daragh Hollman*
(Dated: February 9, 2023)

This report aims to demonstrate and investigate the dynamics of oscillating chemical systems through a study of a specific system known as the Brusselator. This was carried out using Euler's method of numerical integration. It was determined that for ratios of A to B less that 1:2 the system would oscillate but with dampening amplitude until converging to the stable point. Conversely increasing the ratio of A to B resulted in very short periods of dampening before reaching a constant amplitude oscillation. It was also determined that the steady state behaviour of the system was independent of the initial conditions of X and Y as the system converged to the same pattern of constant oscillation regardless of initial conditions with the only difference being in the phase. These findings agree with those found in simulations by Lozno-Parada et al[1] which exhibits the same steady state behaviour and similarly shows cases where the system converges to the stable point.

## I. INTRODUCTION

A chemical oscillator is a non-linear system of reacting chemicals in which exhibits oscillations in the concentrations of their chemicals [2]. Non-linear systems have many applications in modern areas of science and engineering [1] and find use in non-linear control systems such as periodic drug delivery [3]. Due to their complex nature, these kinds of systems can be difficult to set up and maintain as they tend to be very sensitive any changes in their environment conditions. This sensitivity does however does provide some use in the field of analytical chemistry as trace chemicals might upset the oscillatory process [4]. The Brusselator is one such system of chemical oscillations and is the focus of this report.

### A. Chemical Equations

Equations describing chemical reactions can be written as follows:

$$A + B \rightarrow C + D \tag{1}$$

where chemicals $A$ and $B$ interact with each other to form chemicals $C$ and $D$. This reaction will occur at a specific rate defined by an expression known as a rate equation [5]. It is common to assume that any chemical system will follow mass action kinetics [5][6], that the reaction rate is direction proportional to the concentration of the reactants. A rate equation for the system in equation 1 would look similar to:

$$-\frac{dA}{dt} = k[A][B] \tag{2}$$

where $[A]$ and $[B]$ are the concentrations of chemicals A and B respectively and $k$ is the rate constant of the reaction. A negative sign is used to denote that A is decreasing in concentration. In this report we will consider the rate constant to be unity for simplicity.

### B. The Brusselator System

The chemical equations of the Brusselator are typically described as follows [7]:

$$\begin{aligned} A &\rightarrow X & (a)\\ B + X &\rightarrow Y + D & (b)\\ 2X + Y &\rightarrow 3X & (c)\\ X &\rightarrow C & (d) \end{aligned} \tag{3}$$

With ODEs given by:

$$\begin{aligned} \frac{dX}{dt} &= A - (B+1)X + X^2Y & (a)\\ \frac{dY}{dt} &= BX - X^2Y & (b) \end{aligned} \tag{4}$$

The steady state solution of this system is one which stays stationary over time, sometimes referred to as a stable point. At any stable point, the rate of change of $X$ and $Y$ is zero.

$$\frac{dX}{dt} = 0 \ ; \ \frac{dY}{dt} = 0 \tag{5}$$

Hence we can find the stable point by solving for $X$ and $Y$. A full derivation is included in appendix 1, however a single point at $(X, Y) = \left(A, \frac{B}{A}\right)$ was calculated to be the only stable point in the system.

In this report we will investigate the evolution of the Brusselator system over time, and discuss the oscillatory nature of the reaction using phase space diagrams and concentration diagrams. This will be carried out over a range of initial conditions for X and Y, but also varying ratios of A and B.

## II. COMPUTATIONAL METHODS

### A. The Euler Method

The Euler method was chosen to numerically integrate the rate equations to evolve the system over time. The

* daragh.hollman@ucdconnect.ie

Euler method is used to solve the first-order initial value problem [8]:

$$\frac{dy}{dx} = f\left(x, y\right), y(x_0) = y_0 \tag{6}$$

Here we have a first-order ordinary differential equation with a known initial condition. Euler's method makes use of a relatively simple process which takes the slope of the function at an initial point and assumes a linear path between that point and the next some arbitrary step away. The formula is given as follows [9]:

$$y(x+h) = y(x) + hf(x,y) \tag{7}$$

where $h$ is the step size. This will construct the tangent at coordinate $x$, and find the value of $y(x+h)$ to determine the next point. Hence to use Euler's method we can pick a starting point around which we want to approximate and then evaluate equation 7 until we have reached the desired number of steps.

#### 1. *The Application of the Euler method to the System*

The Brusselator system is has two dependent variables which vary with time as shown in the rate equations, see equation 4, and hence we need to run two calculations of Euler's method simultaneously. Rewriting these rate equations in the form of equation 7, we have the following:

$$\begin{aligned} X_{i+1} &= X_i + \Delta t \frac{dX}{dt} \quad (a) \\ Y_{i+1} &= Y_i + \Delta t \frac{dY}{dt} \quad (b) \end{aligned} \tag{8}$$

where $X_i$ is value of $X(t)$ and $X_{i+1}$ is the next step, $X(t + \Delta t)$ with step size $\Delta t$. These variables have the same meaning for equation 8b but in terms of $Y$.

### B. Error Analysis of the Euler Method

#### 1. *Round-off Error*

In this simulation two types of error are introduced in the calculations, round-off error and truncation error. Although the round-off error is negligible compared to the truncation error of this simulation, it is important to reference in the context of a computational report. By default Python uses 64 bits to represent a floating point number [10]. One for the sign, 11 for the exponent and 52 for the fraction. This means that we can only accurately represent a maximum of $2^{1024}$ and a minimum of $2^{-1024}$ [11]. This range is entirely sufficient for the purposes of this report and hence any errors due to round-off are negligible.

#### 2. *Truncation Error*

Aside from the round-off error there exists a truncation error between the exact solution and the solution estimated by Euler's method. There is the local error between each step and the global error which is the summation of all the local errors up to a certain step [12]. If we take the Taylor series approximation for a function:

$$y_{i+1} = y_i + y_i' h + \frac{y_i''}{2} h^2 + \cdots + \frac{y_i^{(n)}}{n!} h^n + \ldots \tag{9}$$

we can clearly see that the first two terms of this are the same as Euler's method shown in equation 7. The Taylor series will be an exact solution if all terms are included however if we truncate it after the first two terms, what remains will be the difference between Euler's method and the exact solution over a step. This is the local error, and for an ODE with:

$$\begin{aligned} \frac{dy}{dx} &= f(x,y) \\ y' &= f(x,y) \end{aligned}$$

we have:

$$y_{i+1} = \underbrace{y_i + f(x_i, y_i)h}_{\text{Euler's Method}} + \underbrace{\frac{f'(x_i, y_i)}{2} h^2 + \ldots}_{\text{Error}} \tag{10}$$

We can assume that, with a small step size $h$, that higher order terms will be small. Hence we can take the local error to scale with $\mathcal{O}(h^2)$ and that terms of $\mathcal{O}(h^3)$ or higher are negligible.

The global error is the summation of the local errors for each step. As the step size decreases, the number of steps within a length L increases with $h^{-1}$. Hence the global error is given by the following:

$$\epsilon_g = \frac{1}{h} \sum \epsilon_l \tag{11}$$

and as the local error scales with $h^2$ and the global error scales with the local error and $h^{-1}$. The global error on the system for any number of steps scales with $\mathcal{O}(h)$, the step size.

## III. RESULTS AND DISCUSSION

Two kinds of plots were created, one showing directly how the concentrations of X and Y vary over time, and the other was a phase space plot of the entire evolution. Chemicals A and B were kept at constant values throughout the evolution while chemicals X and Y were free to vary. Chemical A was given a value of 1 as the concentration of the chemicals is in arbitrary units it made sense to solely vary the concentration of B and inspect different ratios of A to B. For an initial phase space position of $(X, Y) = (0, 0)$, we looked at ratios of A:B of 1:1.5, 1:1.9, 1:2, 1:2.5, and 1:4. These were simulated and plotted in figures 1, 2, 3, 4, and 5 respectively.

From these simulations, it was determined that there is a limit on the ratio of A and B to whether the oscillations dampen. We found that a ratio of A:B of less than 1:2 results in continued dampening of the oscillations of chemicals X and Y until arriving at the stable point. This can be seen clearly in figures 1 and 2. When $B = 1.5$ we see rapid dampening with X and Y reaching constant values after 30 seconds, whereas closer to a ratio of 1:2, when $B = 1.9$ we see longer oscillations with X and Y only reaching constant values after 100 seconds. This is contrasted by figure 3 where A:B is equal to 1:2 and recurring oscillations are present.

For higher ratios of A:B, such as in figures 4 and 5 we still see consistent oscillations however we also see much larger amplitude oscillations for increasing ratios. It can also be seen that the concentrations approach their constant amplitude much quicker meaning the damping time is shorter. The simulation in figure 4 reaches constant amplitude in less than 20 seconds and the simulation in figure 5 reaches constant amplitude almost immediately (less than 10 seconds). This is contrasted by figure 3 where the oscillations reach constant amplitude after approximately 150 seconds.

Figure 6 shows the effect of the initial conditions of X and Y on the oscillations. It can be clearly seen from the phase space plot that the steady state behaviour is independent of the initial conditions of X and Y. It can be seen that regardless of initial location on the phase space plot, each path dampens towards the same constant path in phase space of constant amplitude oscillations. This is demonstrated too by the second plot in figure 6. Despite each line of concentration being out of phase with one another, it can clearly be seen that they follow the same pattern of oscillations, dampening at the same rate until reaching a constant amplitude.

## IV. CONCLUSION

The aim of this report was to demonstrate and investigate, using Euler's numerical method, the dynamics of the Brusselator system as an example of oscillating chemical reactions. The system was looked at with varying initial constants A and B, along with varying initial conditions of the chemicals being investigated X and Y. It was determined that for ratios of A to B less that 1:2 the system would oscillate but with dampening amplitude until converging to the stable point. Conversely increasing the ratio of A to B resulted in very short periods of dampening before reaching a constant amplitude oscillation. It was also determined that the steady state behaviour of the system was independent of the initial conditions of X and Y as the system converged to the same pattern of constant oscillation regardless of initial conditions with the only difference being in the phase. These findings agree with those found in simulations by Lozno-Parada et al[1] which exhibits the same steady state behaviour and similarly shows cases where the system converges to the stable point.

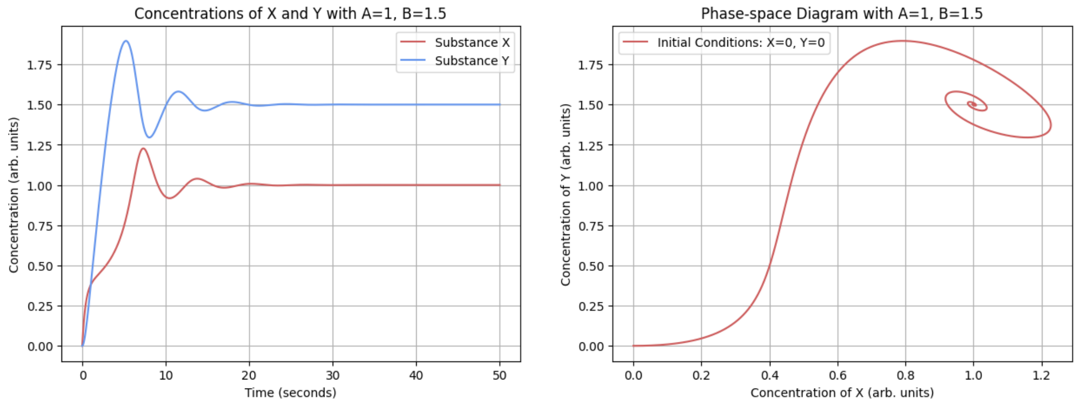

**Figure 1:** Evolutions of the system for 50 seconds ($50,000$ steps with a size of $10^{-3}$) with $B = 1.5$ and initial conditions $X = 0$, $Y = 0$. **Left:** Concentration of each chemical over time. We see a damping in the oscillations as the system falls towards the stable point of the system. **Right:** A phase space plot of the evolution of the concentrations of X and Y over time. We can visually see the damping of the oscillations as the system evolves and stops on the stable point (1, 1.5).

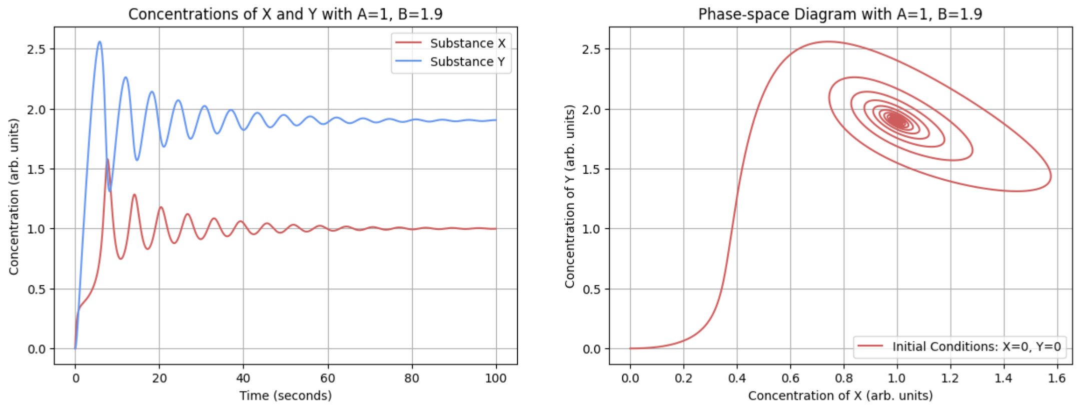

**Figure 2:** Evolutions of the system for 100 seconds ($100,000$ steps with a size of $10^{-3}$) with $B = 1.9$ and initial conditions $X = 0$, $Y = 0$. Similarly to figure 1 we see dampening before converging on the stable point.

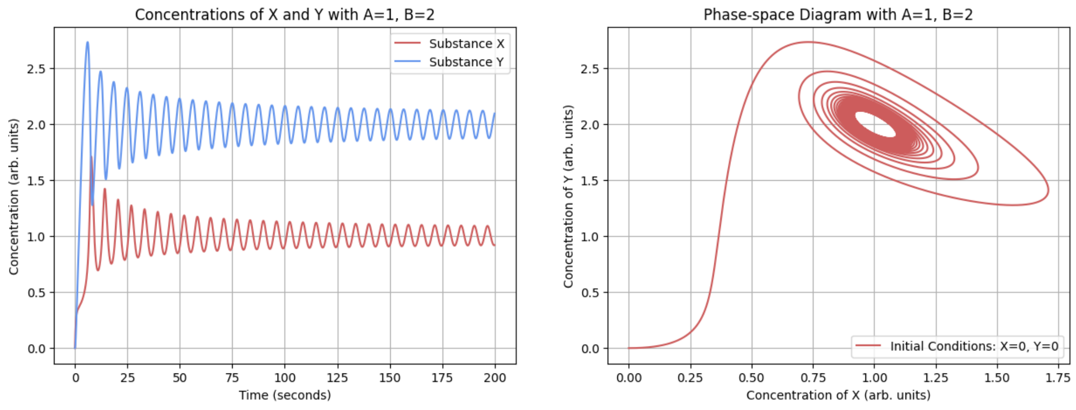

**Figure 3:** Evolutions of the system for 200 seconds ($200,000$ steps with a size of $10^{-3}$) with $B = 2$ and initial conditions $X = 0$, $Y = 0$. **Left:** Concentration of each chemical over time. The oscillations in the concentration of each chemical can be clearly seen. **Right:** A phase space plot of the evolution of the concentrations of X and Y over time. We can see the amplitude of the oscillations decreases before reaching a stable constant amplitude. The oscillations are around the stable point (1, 2).

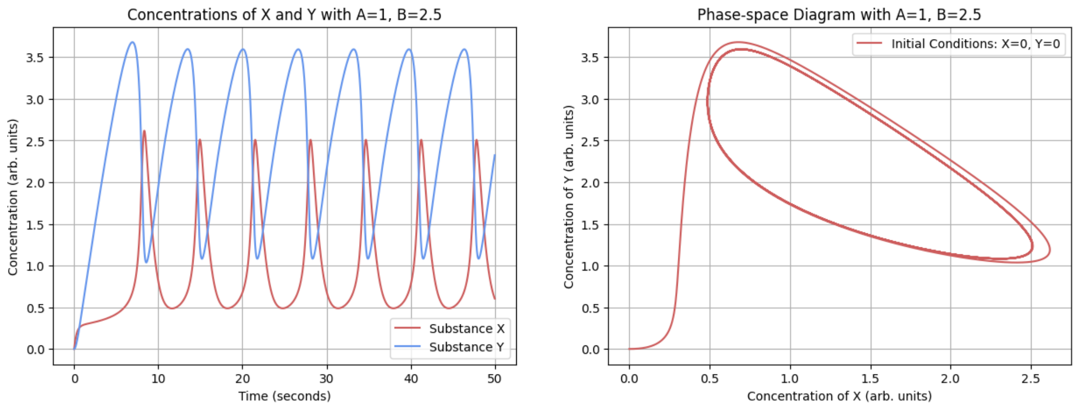

**Figure 4:** Evolutions of the system for 50 seconds ($50,000$ steps with a size of $10^{-3}$) with $B = 2.5$ and initial conditions $X = 0$, $Y = 0$. Here we see that a larger ratio of A:B results in a quicker approach to constant amplitude oscillations.

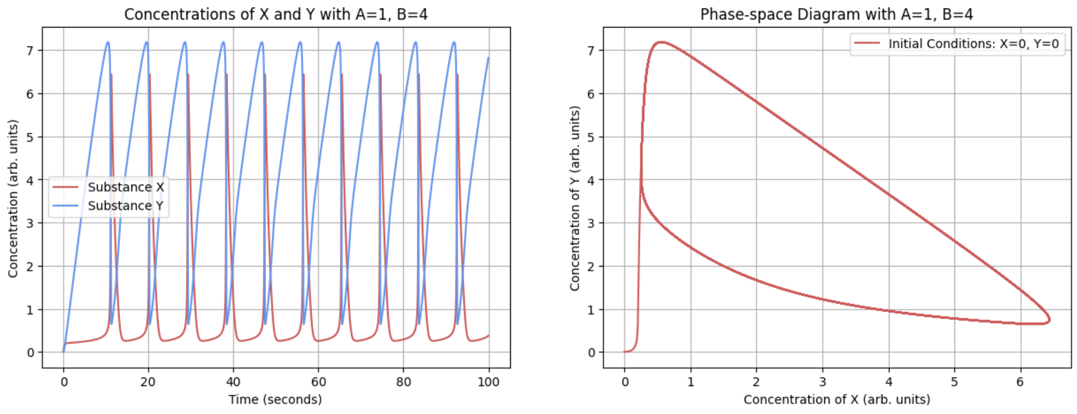

**Figure 5:** Evolutions of the system for 100 seconds ($100,000$ steps with a size of $10^{-3}$) with $B = 4$ and initial conditions $X = 0$, $Y = 0$. Again we see that a larger ratio of A:B results in a quicker approach to constant amplitude oscillations. We also see much larger amplitude oscillations due to the increase in the constant supply of chemical B.

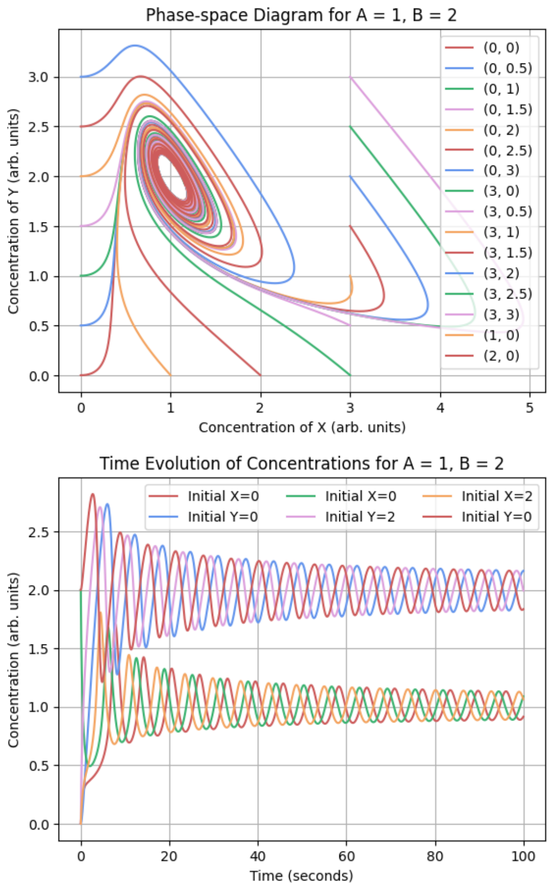

**Figure 6:** Plots showing the differences between 6 different initial conditions for $B = 2$. We see all initial conditions yield the same steady state behaviour. The upper plot shows that each system dampens before reaching the same steady amplitude oscillation surrounding the stable point at (1,2). Although in the lower plot we see each system is out of phase with each other, we see that they still all track the same shape showing an equal dampening regardless of their initial conditions.

[1] J. H. Lozano-Parada, H. Burnham, and F. Machuca Martinez, Pedagogical approach to the modeling and simulation of oscillating chemical systems with modern software: The brusselator model, Journal of chemical education **95**, 758 (2018).

[2] G. Nicolis and J. Portnow, Chemical oscillations, Chemical reviews **73**, 365 (1973).

[3] Z. D. Cupic, A. F. Taylor, D. Horvth, M. Orlik, and I. R. Epstein, Editorial: Advances in oscillating reactions, Frontiers in Chemistry **9**, 10.3389/fchem.2021.690699 (2021).

[4] J. Ren, X. Zhang, J. Gao, and W. Yang, The application of oscillating chemical reactions to analytical determinations, Open Chemistry **11**, 1023 (2013).

[5] S. Ault and E. Holmgreen, Dynamics of the brusselator, (2003).

[6] F. Horn and R. Jackson, General mass action kinetics, Archive for rational mechanics and analysis **47**, 81 (1972).

[7] *Chemical Oscillations*, University College Dublin.

[8] R. B. Israel, Error analysis of the euler method, `https://personal.math.ubc.ca/~israel/m215/euler2/euler2.html`, visited on 01/02/23.

[9] P. Dawkins, Euler's method, `https://tutorial.math.lamar.edu/classes/de/eulersmethod.aspx`, visited on 01/02/23, last modified on 16/11/22.

[10] Python 3.11.1 Documentation, The Python Software Foundation, `https://docs.python.org/3/`, accessed on 08/02/23.

[11] N. Warburton, ACM20030: Computational Science - Introduction and Python Programming.

[12] M. Owkes, Error analysis for Euler's Method, Montana State University.

## APPENDIX 1 - DERRIVATION OF THE STABLE POINT

$$\begin{aligned} A-(B+1)X+X^2Y &= 0 \\ BX-X^2Y &= 0 \end{aligned}$$

$$\therefore \qquad X^2Y = BX$$

$$\begin{aligned} \implies A-(B+1)X+BX &= 0 \\ X &= A \end{aligned}$$

$$\begin{aligned} A^2Y &= BA \\ \implies Y &= \frac{B}{A} \end{aligned}$$

$$(X,Y) = \left(A, \frac{B}{A}\right)$$

## APPENDIX 2 - PYTHON CODE

```
import matplotlib as mpl
import matplotlib.pyplot as plt
import numpy as np

# Change default colours to personal colour scheme
mpl.rcParams['axes.prop_cycle'] = mpl.cycler(color=["indianred", "cornflowerblue",\
 "mediumseagreen", "plum", "sandybrown"])

def StabilityPoint(A, B):
    return (A, B/A)

# Representation of the ODEs as functions

def dXdt(X, Y, A, B):
    return A - (B + 1) * X + X**2 * Y

def dYdt(X, Y, A, B):
    return B * X - X**2 * Y

# Function must input initial values of X and Y along with constants A and B and output
# x against y over time
# Note global error on Euler method is of order O(stepSize)

def Brusselator(initialX=0, initialY=0, A=1, B=2, stepSize=1e-3, numberOfSteps=100000):

    currentX = initialX
    currentY = initialY

    currentTime = 0

    outputs = []
    for i in range(0, numberOfSteps+1):
        # plus one o numberOfSteps so it finalises the last step and doesn't throw an exceptio
        outputs.append([currentX, currentY])

        slopeX = dXdt(currentX, currentY, A, B)
        slopeY = dYdt(currentX, currentY, A, B)

        nextX = currentX + stepSize * slopeX
        nextY = currentY + stepSize * slopeY

        currentX = nextX
        currentY = nextY
        currentTime = currentTime + stepSize

    return outputs

def MakePhasePlot(dataset, A, B, labels=[], stationaryPoint=None):
    i=0
    for data in dataset:
        data = np.array(data)

        if labels != []: plt.plot(data[:,0], data[:,1], label=labels[i])
        else: plt.plot(data[:,0], data[:,1])

        i+=1

    if stationaryPoint != None: plt.scatter(stationaryPoint[0], stationaryPoint[1], label="St

    plt.grid()
    plt.legend()

    plt.title(f"Phase-space Diagram for A = {A}, B = {B}")
    plt.xlabel("Concentration of X (arb. units)")
    plt.ylabel("Concentration of Y (arb. units)")

def MakeEvolutionPlot(data, A, B, stepSize, initalCoords=[]):

    times = np.arange(0, len(data))
    times = [el*stepSize for el in times]

    xValues = []
    yValues = []

    for el in data:
        xValues.append(el[0])
        yValues.append(el[1])

    if initalCoords != []:
        plt.plot(times, xValues, label=f"Initial X={initalCoords[0]}")
        plt.plot(times, yValues, label=f"Initial Y={initalCoords[1]}")

    else:
        plt.plot(times, xValues, label="Substance X")
        plt.plot(times, yValues, label="Substance Y")

    plt.title(f"Time Evolution of Concentrations for A = {A}, B = {B}")
    plt.xlabel("Time (seconds)")
    plt.ylabel("Concentration (arb. units)")
    plt.legend()
    plt.grid()

def MakePlots(initialX, initialY, A, B, stepSize, numberOfSteps, initialConditionsLabel):

    data = Brusselator(initialX, initialY, A, B, stepSize, numberOfSteps)
    data = np.array(data)

    times = np.arange(0, len(data))
    times = [el*stepSize for el in times]

    xValues = []
    yValues = []

    for el in data:
        xValues.append(el[0])
        yValues.append(el[1])

    figure, (concentrationAxis, phasespaceAxis) = plt.subplots(1, 2)

    figure.set_size_inches(15, 5)

    concentrationAxis.plot(times, xValues, label="Substance␣X")
    concentrationAxis.plot(times, yValues, label="Substance␣Y")

    concentrationAxis.set_title(f"Concentrations␣of␣X␣and␣Y␣with␣A={A},␣B={B}")
    concentrationAxis.set_xlabel("Time␣(seconds)")
    concentrationAxis.set_ylabel("Concentration␣(arb.␣units)")
    concentrationAxis.legend()
    concentrationAxis.grid()

    phasespaceAxis.plot(data[:,0], data[:,1], label=f"Initial␣Conditions:␣X={initialX},␣Y={in
    phasespaceAxis.set_title(f"Phase-space␣Diagram␣with␣A={A},␣B={B}")
    phasespaceAxis.set_xlabel("Concentration␣of␣X␣(arb.␣units)")
    phasespaceAxis.set_ylabel("Concentration␣of␣Y␣(arb.␣units)")
    phasespaceAxis.legend()
    phasespaceAxis.grid()
```