Hopf Bifurcations (an iterative approach)

Rachel Huang

December 8, 2025

1 Research Questions

How can iterative methods and matrix properties be applied to the analysis of a Hopf Bifurcation, a nonlinear phenomena? How can Hopf Bifurcations be visualized?

2 Research Plan

1. Read the documentation from (Strogatz, 2024) regarding the Hopf bifurcation and any other related sections
2. Setup a valid Hopf bifurcation system and its corresponding equations
3. Find the critical value of the parameter for the Hopf bifurcation system, and select two parameter values, where one is less than the critical value, and one is greater than the critical value
4. Apply the iterative Newton's method from (Freire et al., 2024) to the selected system at each selected parameter value, using the Jacobian matrix to approximate the system's equilibrium point(s)
5. Calculate the eigenvalues of the system based on the methods described in (Boyce, 2017)
6. Draw the phase portraits for the system at each selected parameter value, and plot the equilibrium (fixed) points that were found
7. Analyze how the calculated eigenvalues are correlated to the stability of the fixed points and behavior of the phase portraits
8. Analyze implementation of Newton's method and its convergence to the system's equilibrium point(s)

3 Introduction

A bifurcation is a qualitative change in a system's behavior that occurs as a result of a change in the system's parameter(s). The system selected for this project has a single parameter, μ.

A Hopf bifurcation occurs when a system settles down to equilibrium as μ approaches a critical value μ_c from the left. As μ increases past μ_c, the system moves away from equilibrium and periodic behavior emerges, as demonstrated by Figure 1.

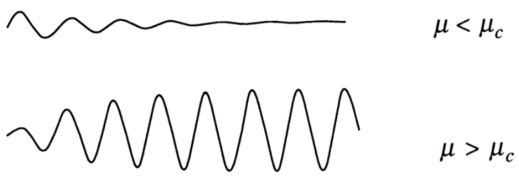

Figure 1: System settling down to equilibrium when $\mu < \mu_c$; System moving away from equilibrium and periodic behavior appearing when $\mu > \mu_c$ (Strogatz, 2024, p. 275)

Physical systems that may exhibit a Hopf bifurcation include structurally unsound airplane wings that begin fluttering as a plane moves past a certain velocity, a overfilled trailer that begins swaying as the vehicle it is attached to drives past a certain velocity, and the Belousov-Zhabotinsky (BZ) chemical reaction in which visible pulsing appears in the solution when a catalyst input surpasses a certain amount.

4 System Analysis

The following system was selected from (Strogatz, 2024), where we are given that a Hopf bifurcation occurs at the critical value of the parameter $\mu_c = 0$.

$$\dot{x} = \mu x - y + xy^2$$

$$\dot{y} = x + \mu y + y^3$$

The parameter values of $\mu = -1$, $\mu = 0$, and $\mu = 1$ were chosen to analyze the system's behavior around the parameter's critical value.

The selected parameter values were plugged into the system for further analysis, yielding:

[1] $\mu = -1$

$$\dot{x} = -x - y + xy^2$$
$$\dot{y} = x - y + y^3$$

[2] $\mu = 0$

$$\dot{x} = -y + xy^2$$
$$\dot{y} = x + y^3$$

[3] $\mu = 1$

$$\dot{x} = x - y + xy^2$$
$$\dot{y} = x + y + y^3$$

In order to find the system's equilibrium point that the system should approach and depart from (in a Hopf bifurcation) as μ moves through μ_c, the iterative Newton's method was used. The corresponding equations used for Newton's method are listed below. Given:

$$f_1(x, y) = 0$$
$$f_2(x, y) = 0$$

- Input: initial estimate x_0
- To find subsequent estimates for x_i (Freire et al., 2024),
  - $J(x_i)\Delta x_i = -F(x_i)$
  - Solve for Δx_i
  - $x_{i+1} = x_i - \Delta x_i$
- $J(x_i)$ = the system's Jacobian matrix evaluated at the current estimate x_i; $F(x_i)$ = the evaluation of the system ($F = f_1, f_2$) at x_i; Δx_i = the difference between the current estimate and the next estimate x_{i+1}.

For example, $f_1(x, y) = -x - y + xy^2$ and $f_2(x, y) = x - y + y^3$ when $\mu = -1$.

By inspection, it seems that the only values of x and y that would satisfy f_1 and f_2 are $(x^*, y^*) = (0, 0)$. Thus, x_0 is set to $(0.5, 0.5)$ as the initial estimate should be near the suspected equilibrium point, if possible, to accelerate method convergence.

Newton's method was implemented using Python and the Sympy library, with the full code below:

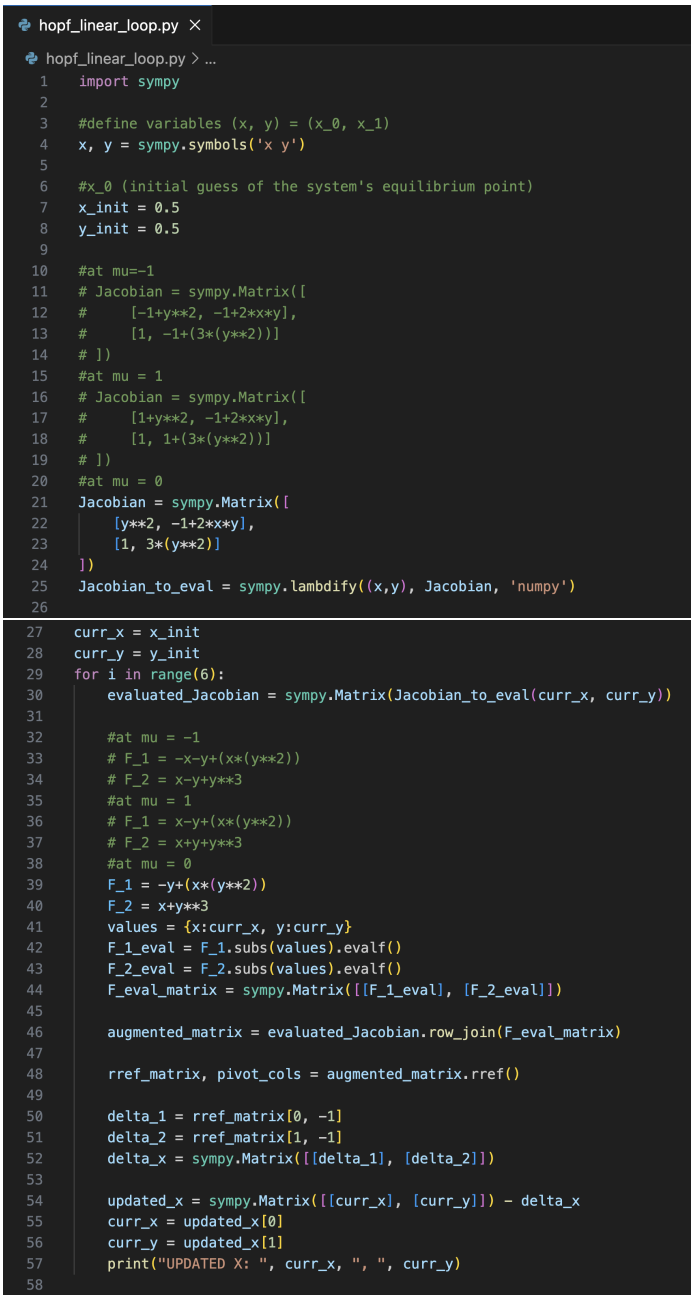

Figure 2: Python implementation of Newton's method and application to selected system

The results of applying the code from Figure 2 to equations [1], [2], and [3] are shown below:

Run command: python3 hopf_linear_loop.py

Results for [1]:
UPDATED X: 0.0909090909090909 , -0.636363636363636
UPDATED X: -0.566096559873785 , 0.235924899272753
UPDATED X: 0.0417644158224028 , 0.0186082253163741
UPDATED X: -8.02498787670636e-6 , -2.09335269199216e-5
UPDATED X: -5.65668253340798e-15 , 1.26899833414844e-14
UPDATED X: 0 , 0

Results for [2]:
UPDATED X: 0.454545454545455 , -0.272727272727273
UPDATED X: -0.0281064943482573 , -0.0558594478871026
UPDATED X: -0.000350230805073407 , 0.000174856383581674
UPDATED X: 1.06923801410086e-11 , 2.14164397479254e-11
UPDATED X: 0 , 0

Results for [3]:
UPDATED X: 0.209302325581395 , 0.0232558139534884
UPDATED X: 0.000126314459632182 , -0.000100995576648420
UPDATED X: 2.58255331182788e-13 , -2.31858652557319e-12
UPDATED X: 0 , 0

Hence, the system's equilibrium point is in fact at $(x, y) = (0, 0)$, as confirmed by the results that Newton's method provided for the selected values of μ.

Next, we calculate the eigenvalues of [1], [2], and [3] based on the methods described in (Boyce, 2017). We begin by calculating the Jacobian matrix for each set of equations. The Jacobian matrix of a system gives the linear approximation of the system at any given point, with each row corresponding to one of the system's equations, and each row entry corresponding to the partial derivative of the equation with respect to one of the input variables.

The calculated Jacobian matrix for the system at each selected parameter value is shown below:

$$J_{\mu=-1} = \begin{bmatrix} -1+y^2 & -1+2xy \\ 1 & -1+3y^2 \end{bmatrix} \qquad J_{\mu=0} = \begin{bmatrix} y^2 & -1+2xy \\ 1 & 3y^2 \end{bmatrix} \qquad J_{\mu=1} = \begin{bmatrix} 1+y^2 & -1+2xy \\ 1 & 1+3y^2 \end{bmatrix}$$

We then evaluate each Jacobian matrix at the equilibrium point $(0, 0)$.

$$J_{\mu=-1}\Big|_{(0,0)} = \begin{bmatrix} -1 & -1 \\ 1 & -1 \end{bmatrix} \qquad J_{\mu=0}\Big|_{(0,0)} = \begin{bmatrix} 0 & -1 \\ 1 & 0 \end{bmatrix} \qquad J_{\mu=1}\Big|_{(0,0)} = \begin{bmatrix} 1 & -1 \\ 1 & 1 \end{bmatrix}$$

The eigenvalues for the system can be found by solving the equation $det(J - \lambda I) = 0$. Thus, we begin by subtracting λ from each entry along the diagonals of the Jacobian matrices.

$$J_{\mu=-1}\Big|_{(0,0)} = \begin{bmatrix} -1-\lambda & -1 \\ 1 & -1-\lambda \end{bmatrix} \qquad J_{\mu=0}\Big|_{(0,0)} = \begin{bmatrix} 0-\lambda & -1 \\ 1 & 0-\lambda \end{bmatrix} \qquad J_{\mu=1}\Big|_{(0,0)} = \begin{bmatrix} 1-\lambda & -1 \\ 1 & 1-\lambda \end{bmatrix}$$

We then calculate the determinant of each matrix, which gives us an equation called the characteristic equation. The roots are the eigenvalues of the system, and we solve for λ, with the resulting eigenvalues provided below:

$$(-\lambda-1)(-\lambda-1)+1=0 \qquad (-\lambda)(-\lambda)+1=0 \qquad (-\lambda+1)(-\lambda+1)+1=0$$

$$\lambda^2+2\lambda+2=0 \qquad \lambda^2+1=0 \qquad \lambda^2-2\lambda+2=0$$

$$\lambda=\frac{-2\pm\sqrt{4-8}}{2} \qquad \lambda^2=-1 \qquad \lambda=\frac{2\pm\sqrt{4-8}}{2}$$

$$\lambda=-1\pm i \qquad \lambda=\pm i \qquad \lambda=1\pm i$$

The phase portrait was then generated for the system at each selected parameter value with the code implemented in Figure 3.

phase_portrait.py

```python
import numpy as np
import matplotlib.pyplot as plt
from scipy.integrate import odeint

def system_dynamics(state, t):
    x, y = state
    #for mu = -1
    # dxdt = -x-y+(x+y**2)
    # dydt = x-y+(y**3)
    #for mu = 0
    dxdt = -y+(x+y**2)
    dydt = x+(y**3)
    #for mu = 1
    # dxdt = x-y+(x+y**2)
    # dydt = x+y+(y**3)
    return [dxdt, dydt]

x_min, x_max = -0.5, 0.5
y_min, y_max = -0.5, 0.5
num_points = 20

x_vals = np.linspace(x_min, x_max, num_points)
y_vals = np.linspace(y_min, y_max, num_points)
X, Y = np.meshgrid(x_vals, y_vals)

U = np.zeros(X.shape)
V = np.zeros(Y.shape)
for i in range(num_points):
    for j in range(num_points):
        state_at_point = [X[i, j], Y[i, j]]
        dxdt_val, dydt_val = system_dynamics(state_at_point, 0)
        U[i, j] = dxdt_val
        V[i, j] = dydt_val

plt.figure(figsize = (8,8))
plt.quiver(X, Y, U, V, color='blue', alpha=0.7, angles='xy', scale_units='xy', scale=17)
# t = np.linspace(0, 50, 5000)
t = np.linspace(0, 3, 2000)
#initial_conditions = [[0.5, 0.0], [0.0, 0.5], [1.0, -0.5], [-0.5, 1.0], [0.5, 0.5], [-0.5, -0.5], [-0.75, -0.75], [0.75, 0.75], [-0.75, 0.75], [0.75,-0.75]]
initial_conditions = [[0.05, 0.0], [0.0, 0.05], [0.05, 0.05], [-0.05, -0.05], [0.07, 0.07], [-0.07, -0.07], [-0.08, -0.08], [-0.02, -0.02], [-0.01, -0.01]]
for ic in initial_conditions:
    solution = odeint(system_dynamics, ic, t)
    plt.plot(solution[:, 0], solution[:, 1], color='red', linewidth=1.5)

plt.xlabel("X")
plt.ylabel("Y")
plt.title("Phase Portrait for μ = 0 (λ = +/- i)")
plt.grid(True)
plt.xlim(x_min, x_max)
plt.ylim(y_min, y_max)
plt.show()
```

Figure 3: Phase portrait generator code implemented in Python.

The directional vectors are shown in blue and selected trajectories are plotted in red. The directional vectors of the phase portrait represent the rate of change (derivatives) of the system at each point. The trajectories plotted on the phase portrait are selected solution curves of the system that are guided by the directional vectors.

$$\mu = -1$$

$$J = \begin{bmatrix} -1 + y^2 & -1 + 2xy \\ 1 & -1 + 3y^2 \end{bmatrix}$$

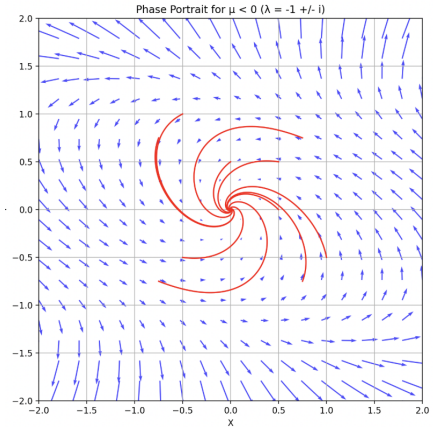

Figure 4: Phase portrait of the system when $\mu = -1$, with selected trajectories in red

$$\mu = 0$$

$$J = \begin{bmatrix} y^2 & -1 + 2xy \\ 1 & 3y^2 \end{bmatrix}$$

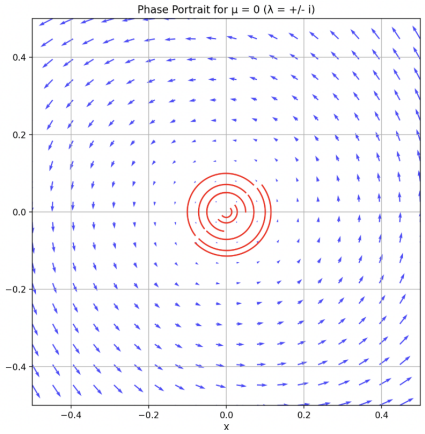

Figure 5: Phase portrait of the system when $\mu = 0$, with selected trajectories in red

$$\mu = 1$$

$$J = \begin{bmatrix} 1 + y^2 & -1 + 2xy \\ 1 & 1 + 3y^2 \end{bmatrix}$$

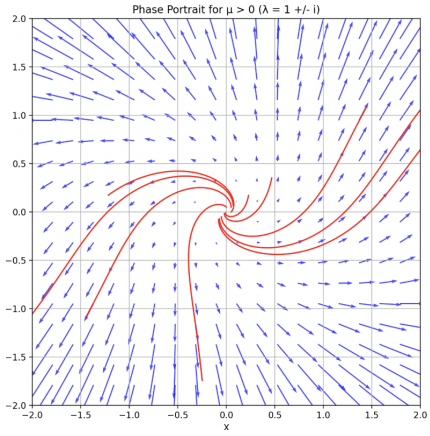

Figure 6: Phase portrait of the system when $\mu = 1$, with selected trajectories in red

To classify the behavior of the system at each parameter value, the matrix can be analyzed with regards to its eigenvalues. The selected system is a continuous time system, since the rate of change equations were provided as derivatives. Therefore, the real components of the eigenvalues and their relation to 0 $(<, >, =)$ determine the stability of the system's behavior at each selected

parameter value.

For Figure 4, the eigenvalues of the system are $\lambda = -1 \pm i$ when $\mu = -1$, and the real component of the eigenvalues is $-1 < 0$. A real eigenvalue component less than 0 corresponds to a classification of the system as stable. Thus, the phase portrait in Figure 3 is classified as a stable spiral, since the directional arrows lead the trajectories to spiral in towards the equilibrium point at $(0, 0)$ over time.

For Figure 5, the real component of the system's eigenvalues is equal to 0. Since classification of system behavior is based on the real component's relation to 0, if the real component is equal to 0, this tell us that this is the parameter value where a change in the system's behavior is occurring. This aligns with the initial information that we were given where the critical value of the parameter (μ_c) is equal to 0.

For Figure 6, the real component of the system's eigenvalues is equal to $1 > 0$. This corresponds to a classification of the system as unstable. Thus, the phase portrait in Figure 5 is classified as an unstable spiral, since the directional arrows lead the trajectories to spiral away from the equilibrium point at $(0, 0)$ over time.

The observed transition of the system's behavior from a stable spiral to an unstable spiral when the parameter moves through 0 from -1 to 1 aligns with the general expected behavior of a Hopf bifurcation previously described in Section 3. For further detail, the phase portraits of a generalized Hopf bifurcation are displayed in Figure 6. In the diagram, there is also a stable spiral when $\mu < 0$, and an unstable spiral when $\mu > 0$, given that $\mu_c = 0$. The circular bound around the unstable spiral is the fixed amplitude of the periodic behavior that the unstable behavior approaches (and is bounded by) in Section 3.

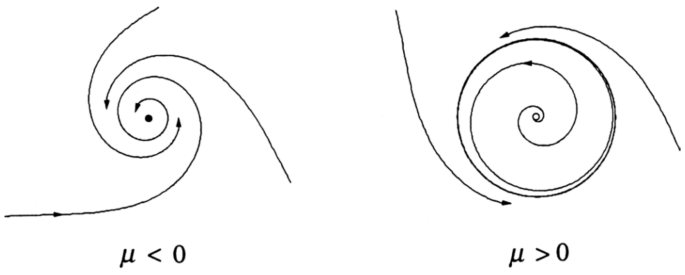

Figure 7: Transition in system (phase portrait) behavior when a Hopf bifurcation occurs at $\mu_c = 0$ (Strogatz, 2024, p.276)

5 Overall Analysis

The application of the iterative Newton's method to an analysis of the Hopf Bifurcation, a nonlinear phenomena, can likely be extended to other nontraditionally linear problems. Newton's method was able to successfully converge to the system's true equilibrium point in a reasonable number of iterations for each selected parameter value. Since the selected initial input to Newton's method was $(0.5, 0.5)$ and reasonably close to the equilibrium point at $(0, 0)$, various initial inputs were also trired to confirm that Newton's method could converge to the origin in a reasonable number of steps.

Input x_0	**Number of Iterations to Convergence for $\mu = -1$**	**Number of Iterations to Convergence for $\mu = 0$**	**Number of Iterations to Convergence for $\mu = 1$**
$(0.5, 0.5)$*	6	5	4
$(0.2, 0.2)$	4	4	3
$(0.6, 0.6)$	15	6	4
$(1.0, 1.0)$	51	2	5

Table 1: The number of iterations to convergence for each selected parameter value. * marks the input x_0 that was analyzed primarily in this project.

Table 1 displays the number of iterations to convergence for each selected parameter value, which when averaged together give that the average number of iterations to convergence across all values in the table is approximately 9. Even when the initial input of $(1.0, 1.0)$ was $\sqrt{2}$ away from the equilibrium point at $(0, 0)$ (as calculated with the distance formula), the average number of iterations across the three parameter values is still less than 20. Based on these results, it seems plausible to conclude that Newton's method could be applied to the analysis of system equilibrium points in a relatively efficient manner as long as the initial input is within a certain range of the true equilibrium point. Implementing Newton's method in code significantly improves the speed at which subsequent iterations can be calculated, and makes the project parameters for this project easily modifiable.

Future work:

- The bounding circle around the unstable spiral found in generalized Hopf bifurcations was not included in the generated phase portraits, since the plotted trajectories were only clearly visible very close to the origin. The phase portrait generator code implemented in this project could likely be improved so that trajectories could still be clearly visible from a greater distance away from the origin, so that more non-local behavior could be visualized.

- The system could be converted to polar coordinates $(\dot{r}, \dot{\theta})$ for direct analysis of the amplitude of the Hopf bifurcation's periodic and angular behavior. In this system representation, $\dot{r}$ represents the rate of change of the system's radius, which aligns with the circular behavior that we saw in the project's generated phase portraits. $\dot{\theta}$ represents the rate of change of the system's angle towards, and away from (or around at a fixed radius) as the system evolves through time.

- The methods from this project could be applied to more complex system where a Hopf bifurcation occurs, with particular interest to systems with multiple equilibrium points and/or multiple parameters.

Acknowledgments: Thank you to Professor Amanda Landi at William & Mary for her guidance in ensuring that the idea of applying Newton's method to a nonlinear phenomena was a valid approach.

Bibliography

Boyce, William E, Richard C DiPrima, and Douglas B Meade. 2017. *Elementary Differential Equations and Boundary Value Problems*. John Wiley & Sons.

Freire, Rusdrael Antony de Araújo, Márcio Matheus de Lima Barboza, and Francisco Márcio Barboza. 2024. “Newton’s Method Applied to Nonlinear Boundary Value Problems: A Numerical Approach,” November. https://arxiv.org/html/2411.09029v1.

Strogatz, Steven H. 2024. *Nonlinear Dynamics and Chaos*. CRC Press.