Bifurcation Diagram of Positive Feedback Circuit

Eloi Vilella

2024-10-22

Contents

The given equation describes the time evolution of a molecule or species X under the influence of a positive feedback loop. The equation is:

$$\frac{dX}{dt} = S + \frac{\beta X^n}{k^n + X^n} - \delta X$$

Explanation of Terms:

- S: A constant input signal that adds to X.
- $\frac{\beta X^n}{k^n+X^n}$: A Hill function describing positive feedback with cooperativity. The parameter n determines the degree of cooperativity, k is the half-saturation constant, and β is the maximal strength of feedback.
- δX: A decay term with rate δ.

What we are going to do:

To explore the system's behavior, we will generate a **bifurcation diagram**. The bifurcation diagram will show how the steady-state values of X change as we vary the decay rate δ. Specifically, we will: 1. Set the time derivative $\frac{dX}{dt} = 0$ to find the steady states (since X is at equilibrium when the net rate of change is zero). 2. Solve the resulting algebraic equation for X for a range of values of δ. 3. Plot X as a function of δ, showing how the system transitions between different behaviors, such as monostability, bistability, or other types of feedback-related behaviors.

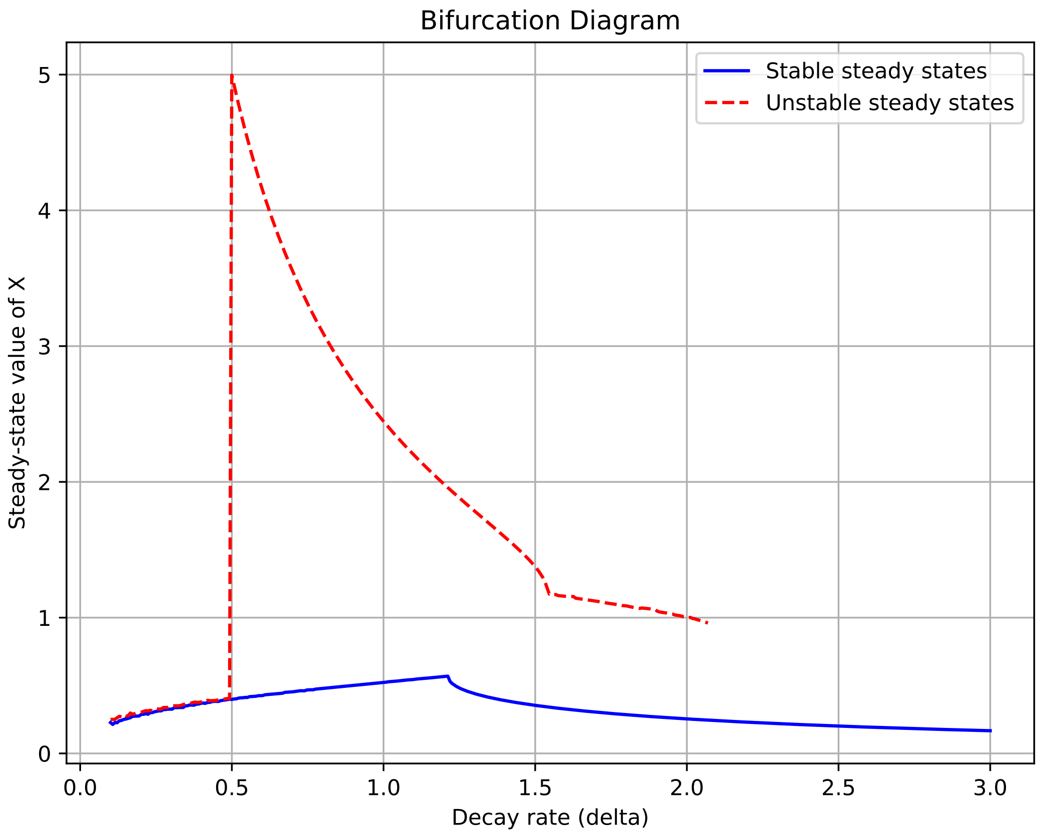
Let's start by generating the bifurcation diagram.

Steps:

1. Rearrange the equation to find the steady states where $\frac{dX}{dt} = 0$.
2. Use numerical methods to solve for X at different values of δ, while keeping other parameters constant.
3. Plot the stable and unstable steady states of X as a function of δ.

```python
import numpy as np
import matplotlib.pyplot as plt
from scipy.optimize import fsolve

S = 0.5
beta = 2.0
k = 1.0
```

```python
n = 4
delta_vals = np.linspace(0.1, 3.0, 400)
X_vals = np.linspace(0.0, 5.0, 400)

def steady_state_eq(X, delta, S, beta, k, n):
    return S + (beta * X**n) / (k**n + X**n) - delta * X

stable_X = []
unstable_X = []

for delta in delta_vals:
    steady_states = []
    for X_guess in X_vals:
        X_sol = fsolve(steady_state_eq, X_guess, args=(delta, S, beta, k, n))
        if 0 <= X_sol[0] <= 5:
            steady_states.append(X_sol[0])

    steady_states = np.unique(np.round(steady_states, decimals=5))

    if len(steady_states) == 1:
        stable_X.append(steady_states[0])
    elif len(steady_states) > 1:
        stable_X.append(steady_states[0])
        unstable_X.append(steady_states[-1])

plt.figure(figsize=(8, 6))
plt.plot(delta_vals[:len(stable_X)], stable_X, 'b-', label="Stable steady states")
if unstable_X:
    plt.plot(delta_vals[:len(unstable_X)], unstable_X, 'r--', label="Unstable steady states")

plt.title('Bifurcation Diagram')
plt.xlabel('Decay rate (delta)')
plt.ylabel('Steady-state value of X')
plt.legend()
plt.grid(True)
plt.show()
```


Bifurcation Diagram
Stable steady states
Unstable steady states
Steady-state value of X
Decay rate (delta)
0
1
2
3
4
5
0.0
0.5
1.0
1.5
2.0
2.5
3.0