Spatio–Temporal Chaos in Belousov–Zhabotinsky Systems:

Modelling & Experimental Characterization

Sarthak Singh
Department of Chemical Engineering, IIT Kanpur
Mentor: Prof. Dipin S. Pillai

1 Abstract

The Belousov–Zhabotinsky (BZ) reaction is a classic nonlinear chemical oscillator known for periodic color changes driven by redox cycles involving ferroin. Unlike equilibrium-driven reactions, the BZ system exhibits sustained oscillations and complex pattern formation arising from autocatalytic and inhibitory feedback loops, modeled by the Field–Körös–Noyes (FKN) mechanism. Its reduced form, the Oregonator, effectively captures the system's nonlinear dynamics. When performed in thin layers (e.g., Petri dishes or gels), the BZ reaction gives rise to concentric waves and spirals through reaction–diffusion coupling. This study integrates experiments and simulations to analyze wave dynamics: time-series data are extracted via RGB analysis, and pattern formation is modeled using the Oregonator framework. This dual approach enables parameter estimation and deeper understanding of spatio-temporal chaos.

2 Objective of the Study

The aim of this work is to experimentally and computationally investigate spatio-temporal chaos in the BZ reaction system. Specifically, we seek to:

- Conduct BZ experiments to observe temporal oscillations and wave patterns, and extract quantitative data via RGB analysis.
- Simulate wave dynamics using the Oregonator model to replicate and understand the observed behavior.
- Compare simulation results with experimental data to validate the model and identify transitions to complex behavior.

3 Experimental Methods

The experimental protocol is adapted from Arthur T. Winfree's "*The Geometry of Biological Time*" with specific modifications to suit our observational setup.

Reagents and Solutions

- **Solution A:** 2.57 g $NaBrO_3$ + 34.57 mL H_2O + 1.08 mL H_2SO_4 (99%)
- **Solution B:** 1 g Malonic Acid in 10 mL H_2O (6 mL used)
- **Solution C:** 1 g NaBr in 10 mL H_2O (3 mL used)
- **Solution D:** Ferroin Indicator (6 mL used)

Procedure

1. Mix 6 mL of Solution A with 6 mL of Solution B. Then, add 3 mL of Solution C, followed by adding 6 mL of Solution D to initiate oscillation.
2. Observe color changes (red $\leftrightarrow$ blue) for 5–10 cycles in a beaker.
3. Transfer to a shallow Petri dish (10 cm diameter).
4. Record top-view video of the reaction for analysis.

(a) Red-colored state after mixing all solutions

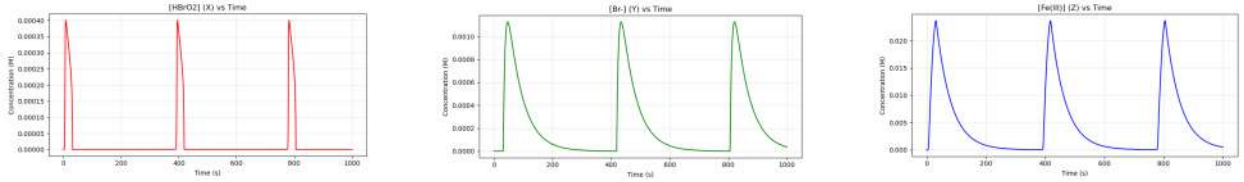
(b) Blue state after oxidation – indicating oscillation

Figure 1: Color change in beaker during BZ reaction indicating redox oscillations

4 Oregonator Model

The Oregonator model is a reduced version of the Field–Körös–Noyes (FKN) mechanism, which captures the essential nonlinear dynamics of the BZ reaction using a minimal set of species: an activator (X), an inhibitor (Z), and an intermediate (Y).

The model simulates the time evolution of these species through a set of coupled ordinary differential equations (ODEs). The simulation was implemented using a stiff solver via `solve_ivp` in Python with tight tolerances to handle sharp transitions in the dynamics. The resulting time series show periodic oscillations characteristic of autocatalytic behavior in the BZ system.



(a) $x(t)$ – [$HBrO_2$] (b) $y(t)$ – [Br^-] (c) $z(t)$ – [Fe(III)]

Figure 2: Oscillatory dynamics of the activator, intermediate, and inhibitor species in the Oregonator model

The system exhibits a limit cycle behavior in its phase space, where the concentrations of species evolve in a closed loop trajectory. This is a hallmark of sustained oscillations in nonlinear dynamical systems.

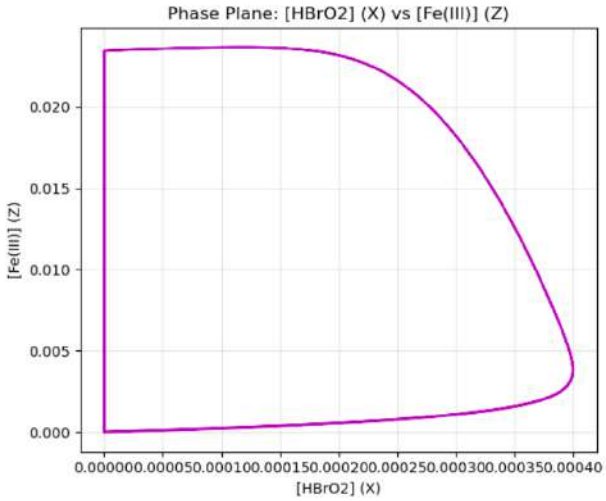


Figure 3: Limit cycle in phase space: x vs. z (activator vs. inhibitor)

5 Simulation Overview

The simulation was implemented using a custom MATLAB workflow incorporating the Oregonator model:

1. **Video Processing:** Extract frames and define ROI.
2. **RGB Analysis:** Normalize and smooth red-channel intensity.
3. **Model Implementation:** Solve Oregonator PDE using finite differences.
4. **Parameter Tuning:** Adjust $\varepsilon = 0.0424$, $f = \frac{2}{3}$, $q = 0.08$.
5. **Pattern Comparison:** Match model output to experimental wavefronts.

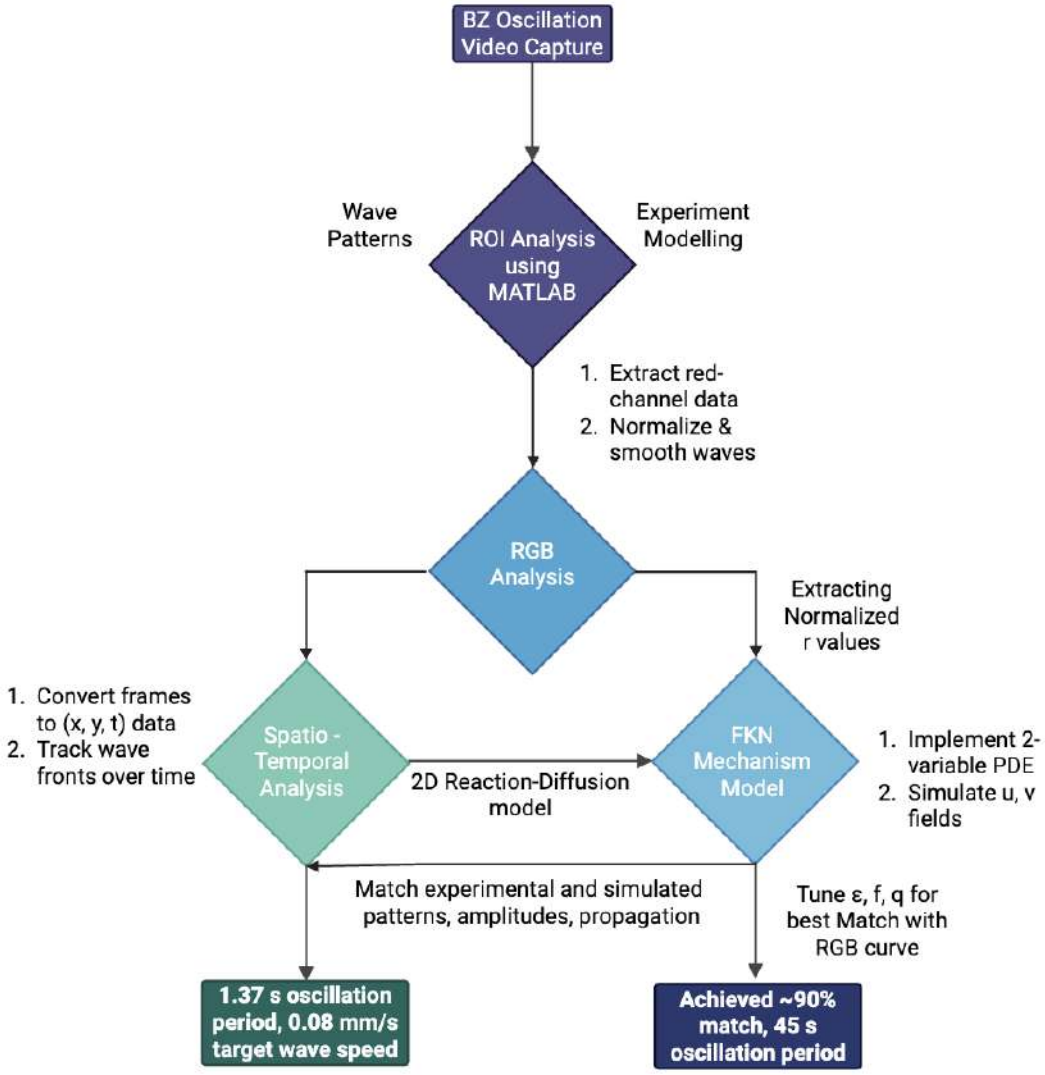


Figure 4: Workflow pipeline for video analysis and Oregonator simulation

6 Model Equations and Parameters

The Oregonator model is a simplified set of reaction-diffusion equations representing the dynamics of the BZ system:

$$\frac{\partial u}{\partial t} = \frac{1}{\varepsilon}\left(u - u^2 - (fv + q)\frac{u - q}{u + q}\right) + D_u \nabla^2 u$$

$$\frac{\partial v}{\partial t} = u - v + D_v \nabla^2 v$$

where:

- u, v are dimensionless concentrations of the activator and inhibitor.
- ε controls the timescale separation (smaller ε yields sharper, faster waves).
- f is the stoichiometric factor relating bromate to malonic acid.
- q is the inhibitor control parameter affecting the autocatalytic term.
- D_u, D_v are diffusion coefficients.

Parameter Values Used: $\varepsilon = 0.0424$, $f = \frac{2}{3}$, $q = 0.08$

7 Pattern Observations in Petri-Dish

(a) Early diffusion in petri dish

(b) Wavefront propagation in petri dish

(c) Concentric pattern formation

Figure 5: Snapshots of wave evolution in the Petri dish during the BZ reaction

8 Results Obtained

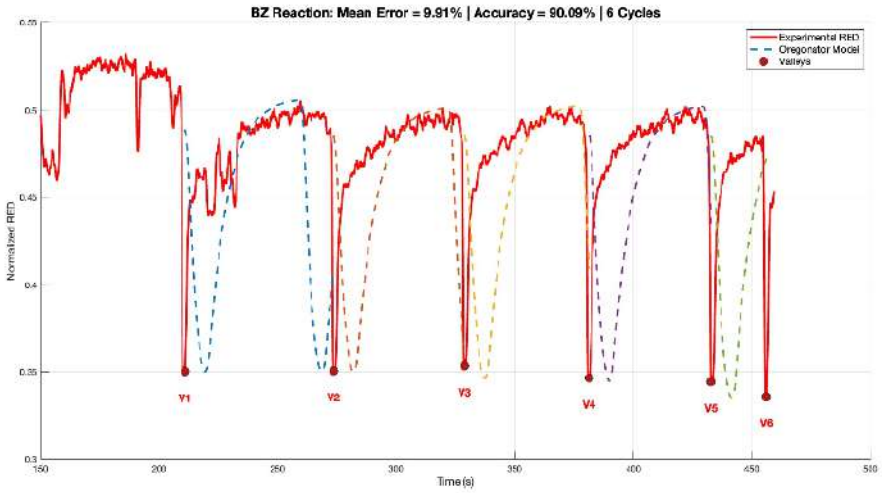


Figure 6: Comparison between experimental RED intensity curve and Oregonator model output, showing strong agreement in oscillation profile and periodicity.

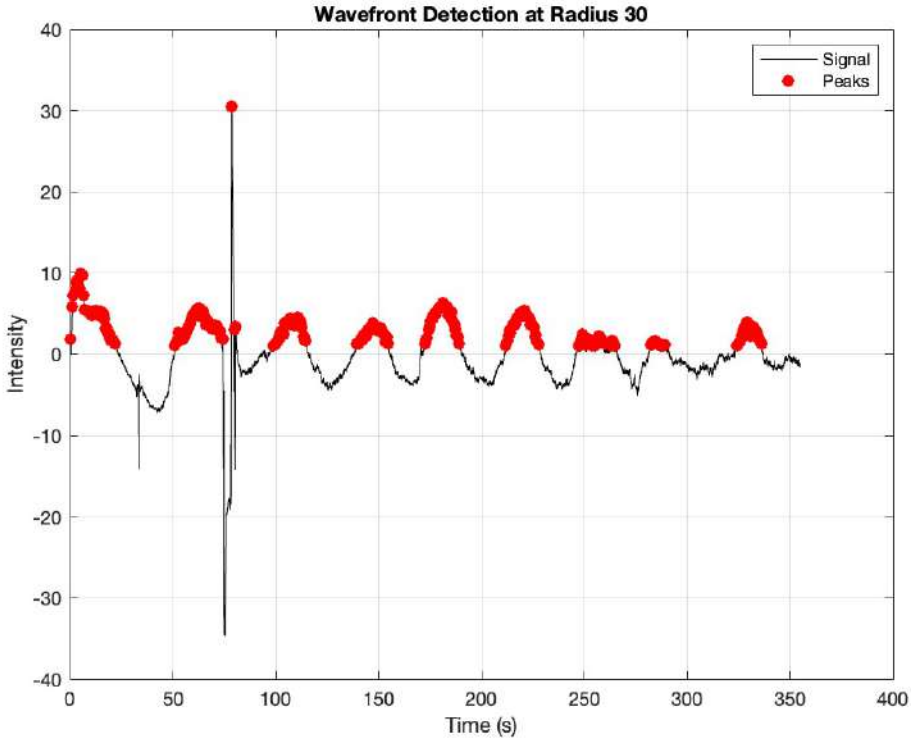


Figure 7: Radial intensity variation at fixed radius = 30 pixels. The detected peaks correspond to successive wavefronts passing through the point, enabling period estimation.

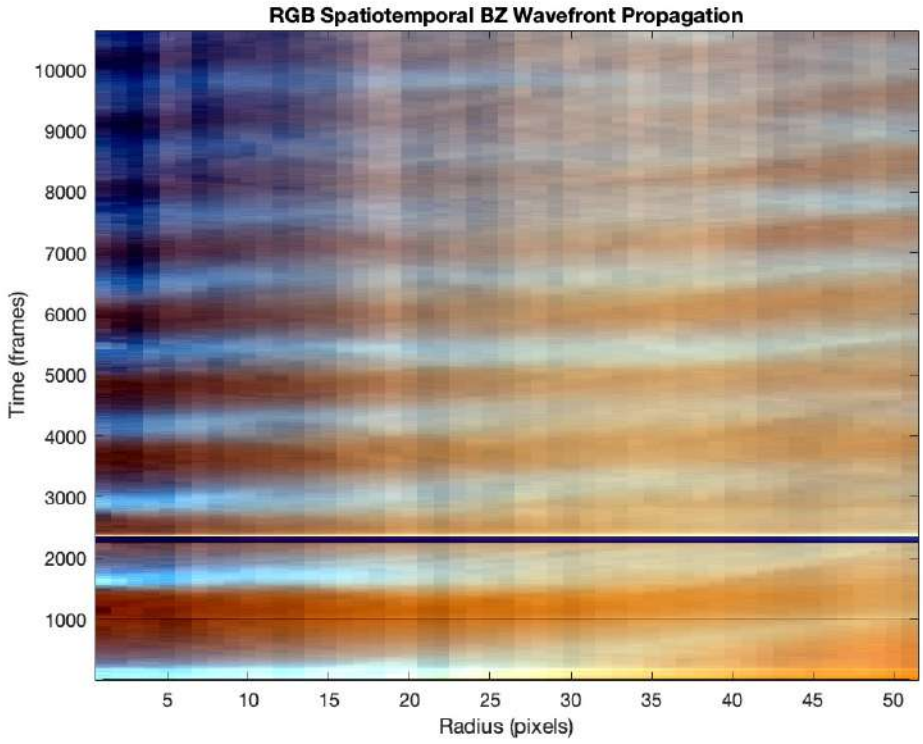


Figure 8: Spatio-temporal analysis of wavefront propagation using RGB intensity profiles. Each stripe corresponds to a radial wavefront detected in the Petri dish.

- Periodic redox oscillations with an average cycle of 45 seconds.
- Red channel yielded highest contrast for intensity tracking.
- Experimental wavefront speed averaged 0.08 mm/s.
- Oregonator model achieved 90% match with RED intensity plot.
- Wavefronts appeared 3–5 minutes post initiation.

9 Future Scope

- Transition to gel-based media for stable quasi-2D wave propagation.
- Extend simulation to include chaotic regimes and spiral turbulence.
- Study transition to chaos using bifurcation analysis and Lyapunov exponents.
- Explore alternative models (e.g., Brusselator) for richer dynamics.
- Develop 3D simulations and real-time analysis tools using computer vision.

References

1. Gray, C. *An Analysis of the Belousov-Zhabotinskii Reaction*.
2. Ai, J., Zhai, C., Sun, W. *Formation of Complex Chemical Waveforms via Computational Methods*.
3. Winfree, A. T. *The Geometry of Biological Time*, Chapter 11.

A Experimental Concentration Table

Chemical	Moles	Concentration (M)
$NaBrO_3$ (A)	0.01703	0.337
Malonic Acid (B)	0.00577	0.114
NaBr	0.00292	0.0578
Ferroin Solution	0.00015	0.0030

Table 1: Calculated concentrations for 50.5 mL total volume

B Parameter Calculation Table

Parameter	Expression	Value Used	Remarks
ε	Timescale ratio	0.0424	Set for sharp wavefronts
q	Inhibitor control	0.08	Affects autocatalysis
f	Stoichiometric factor	$\frac{2}{3}$	BrO_3^-:Malonic Acid ratio
D_u	Diffusion coefficient (activator)	1.0	Normalized
D_v	Diffusion coefficient (inhibitor)	0.6	Lower than D_u

Table 2: Summary of model parameters and calculation notes

C Reaction Mechanism

The key reactions governing the BZ system based on the Field–Körös–Noyes (FKN) mechanism:

- A (BrO_3^-) + B (Malonic Acid) $\rightarrow$ $HBrO_2$ (Activator)
- $HBrO_2 + Br^- \rightarrow 2HOBr$
- $HBrO_2$ + Ferroin (Fe^{2+}) $\rightarrow$ Fe^{3+} (Z) + HOBr
- Fe^{3+} + Malonic Acid $\rightarrow$ Fe^{2+} + Product
- Br^- regeneration step completes the feedback loop

These reactions give rise to feedback loops responsible for oscillations and wave propagation in the BZ medium.

D Code for Oregonator Simulation (Python)

The following Python script was used to simulate the Oregonator model using stiff ODE integration.

Listing 1: Python Code: Oregonator Simulation

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

def oregonator(t, y, epsilon, q, f):
    x, y_, z = y
    dxdt = (1/epsilon)*(x*(1 - x) - f*z*(x - q)/(x + q))
    dydt = x - y_
    dzdt = f*(z - x)
    return [dxdt, dydt, dzdt]

# Parameters
epsilon = 0.04
q = 0.08
f = 2/3

# Initial conditions
y0 = [0.2, 0.3, 0.4]
t_span = (0, 200)
t_eval = np.linspace(*t_span, 1000)

sol = solve_ivp(oregonator, t_span, y0, t_eval=t_eval, args=(epsilon, q, f))

# Plot
plt.plot(sol.t, sol.y[0], label='x (HBrO2)')
plt.plot(sol.t, sol.y[1], label='y (Br)')
plt.plot(sol.t, sol.y[2], label='z (Fe(III))')
plt.legend()
plt.xlabel('Time')
plt.ylabel('Concentration')
plt.title('Oregonator Oscillations')
plt.show()
```

E Essential MATLAB Code: RED Channel Analysis and Model Fitting

The following key sections of the MATLAB script capture the methodology used to:

- Extract RED channel intensity from the experimental video,
- Identify oscillatory valleys,
- Simulate the Oregonator model for multiple cycles,
- Estimate model accuracy by comparing experimental vs simulated RED profiles.

Listing 2: Detecting Valley Points in RED Channel

```
function valley_locs = find_major_valleys_only(signal, time_vec, ...
                    min_prominence_ratio, min_distance_sec)
    dt = mean(diff(time_vec));
    min_distance_samples = round(min_distance_sec / dt);
    ...
    for i = 2:length(signal)-1
        if signal(i) < signal(i-1) && signal(i) < signal(i+1)
            % Compute prominence
            ...
            if prominence > threshold
                valley_locs(end+1) = i;
            end
        end
    end
end
```

Listing 3: Oregonator ODE and Model Simulation

```
odefun = @(t, y) [(y(1)*(1 - y(1)) + f*y(2)*(q - y(1))/(q + y(1)))/epsilon;
                  y(1) - y(2)];
[tau, y] = ode15s(odefun, [0, T], [0.1; 0.05]);
z_model = 1 - y(:,2); % Simulated Ferroin signal
```

Listing 4: Accuracy Estimation Between Model and Experiment

```
y_fit = rescale(z_model, min(r_seg), max(r_seg));
error = mean(abs(y_fit - r_seg) ./ r_seg) * 100;
accuracy = 100 - error;
```

Note: The complete script used for video processing, RGB smoothing, model fitting and plot generation is available at: `github.com/sarthaks-23/Belousov-Zhabotinsky-Systems`

F Essential MATLAB Code: Spatio-Temporal Wavefront Analysis

This appendix outlines the core logic for detecting radial wavefront propagation in BZ reaction videos.

Listing 5: Extracting Circular Intensity Profile

```
theta = linspace(0, 2*pi, 100);
for r = 1:maxRadius
    xCirc = round(centerX + r * cos(theta));
    yCirc = round(centerY + r * sin(theta));
    idx = sub2ind([H W], yCirc, xCirc);
    intensity_row(r) = mean(grayFrame(idx));
end
spatialProfile(frameIdx, :) = intensity_row;
```

Listing 6: Detecting Peaks in Fixed Radius Signal

```
signal = spatialProfile(:, radiusIdx);
signal = signal - mean(signal);
for i = 2:length(signal)-1
    if signal(i) > signal(i-1) && signal(i) > signal(i+1) && ...
       signal(i) > mean(signal) + threshold
        peakLocs(end+1) = i;
    end
end
```

Listing 7: Wavefront Speed Estimation

```
coeffs = polyfit(waveTimes, waveRadii, 1);
waveSpeed = coeffs(1); % in pixels/second
```

For the full implementation of spatio-temporal visualization and tracking across multiple radii, see: `github.com/sarthaks-23/Belousov-Zhabotinsky-Systems`