Numerical Solution to Thermoacoustic Instability in horizontal Rijke tube including Bifurcation analysis

Student	Parikshit Sonwane
Roll no.	AE22B042
Course	AS6320: Acoustic Instabilities in Aerospace Prop
Assignment	03
Submission date	26.04.2025

Contents

List of Figures

Abstract

This study investigates the emergence of thermoacoustic instabilities in a horizontal Rijke tube using a low Mach number approximation and nonlinear modeling. The approach relies on Galerkin projection to derive a reduced-order model consisting of delay differential equations. These equations are solved numerically using a fourth-order Runge-Kutta scheme. Our analysis reveals how variations in heater power and time delay influence the stability of the system, uncovering nonlinear phenomena such as hysteresis and subcritical Hopf bifurcations. The results provide valuable insights into the underlying mechanisms that drive thermoacoustic instabilities in combustion systems.

1 Introduction

Thermoacoustic instabilities represent a complex coupling between unsteady heat release and acoustic pressure oscillations. These phenomena are particularly significant in propulsion systems, gas turbines, and rocket engines, where sustained oscillations can cause structural fatigue and eventual system failure. The Rijke tube, a classical experimental setup, offers a controlled platform to study such instabilities.

A typical Rijke tube consists of a cylindrical duct with a heat source placed at a fixed axial position. When specific conditions are met—such as adequate heat input and favorable phase alignment between heat release and pressure perturbations—the system exhibits spontaneous oscillations. This behavior is governed by the Rayleigh criterion, which states that acoustic oscillations grow if heat is added during the compression phase.

While linear stability analysis provides some initial insights, it fails to capture the full spectrum of nonlinear behaviors such as limit cycles, bifurcations, and chaotic transitions. Hence, a nonlinear time-domain modeling framework is required to explore these regimes. In this study, we employ a Galerkin-based model reduction technique to convert the governing partial differential equations into a finite set of ordinary differential equations, incorporating a time delay to simulate heat release response. This system is then solved using a fourth-order Runge-Kutta method.

2 Background and Objectives

Combustion-acoustic coupling poses a significant challenge in aerospace propulsion, often leading to thermoacoustic instabilities. The horizontal Rijke tube serves as a canonical system to explore these phenomena, especially nonlinear behaviors such as hysteresis and subcritical Hopf bifurcations, which are not captured by linear models. These instabilities are governed by the phase alignment between pressure oscillations and unsteady heat release, with acoustic damping playing a crucial role.

This study adopts a zero Mach number approximation based on models by Balasubramanian and Sujith 1, and Subramanian et al. 2, using non-dimensional linearized equations for the acoustic field. Heat release response is modeled using Heckl's correlation, capturing the effect of velocity perturbations on heat transfer from the hot wire.

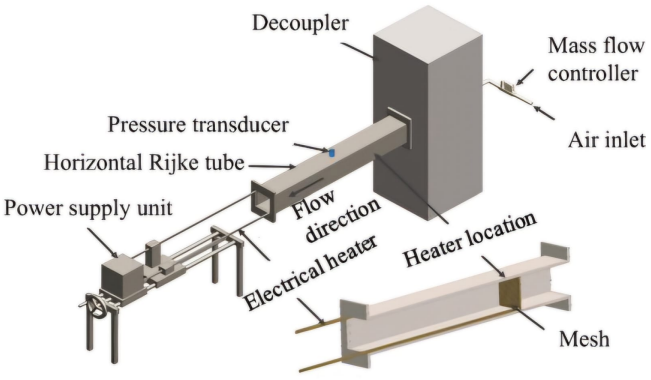


Figure 1: Rijke Tube Setup

The objectives of this work are to:

- Develop a reduced-order Galerkin model with explicit time delays.
- Capture hysteresis loops via parameter sweeps of heater power (K).
- Study the sensitivity of instability thresholds to time delay (τ).

3 Methodology

3.1 System Description and Assumptions

The system under investigation is a horizontal Rijke tube of unit length containing a wire heater located at a fixed position x_f. Air flows through the tube with a uniform mean velocity u_0.
We assume the following:

- The flow is one-dimensional, inviscid, and compressible.
- The acoustic disturbances are small (i.e., linear acoustics is valid for base equations).
- A low Mach number approximation ($M = u_0/c_0 \ll 1$) is used to decouple the mean flow and acoustic fluctuations.
- Heat release is modeled using a modified King's law with a nonlinear dependence on velocity perturbation.

3.2 Governing Equations

We begin with the momentum and energy equations for fluids, assuming the medium is ideal and inviscid. The variables x, t, p',u' and u_0 are non-dimensionalized using L_a , L_a/c_0, $\bar{p}$, u_0 and c_0 respectively where c_0 is the speed of sound, L_a is the duct length, $\bar{p}$ is the pressure of the undisturbed medium, and u_0 is the mean flow velocity.

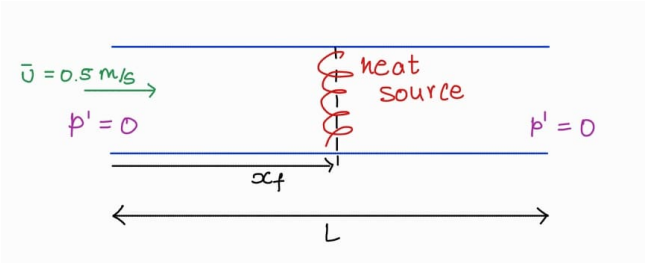


Figure 2: Rijke Tube Schematic for the defined system

The linearised non-dimensional **momentum** equation in one dimension is given by:

$$\gamma M \frac{\partial u'}{\partial t} + \frac{\partial p'}{\partial x} = 0 \tag{1}$$

and the corresponding linearized non-dimensional **energy** equation is:

$$\frac{\partial p'}{\partial t} + \gamma M \frac{\partial u'}{\partial x} = k \left[\sqrt{\left| \frac{1}{3} + u'(t-\tau) \right|} - \sqrt{\frac{1}{3}} \right] \delta(x - x_f) \tag{2}$$

where using a Heckl's modified form of Kings' law for heat release rate,

$$k = (\gamma - 1) \frac{2L_w(T_w - T)}{\gamma M S c_0 \bar{p} \sqrt{3}} \sqrt{\pi \lambda C_v \bar{\rho} \frac{d_w}{2} u_0}$$

L_w is the equivalent length of the wire, λ is the heat conductivity of air, C_v is the specific heat of air at constant volume, τ is the time lag, $\bar{\rho}$ is the mean density of air, d_w is the diameter of the wire, $T_w - T$ is the temperature difference, and S is the cross-sectional area of the duct.

The Galerkin technique is applied to reduce these equations to a system of ordinary differential equations (ODEs). The basis functions chosen are the natural acoustic modes of the duct in the absence of a heater.

We assume the solution to p'(x,t) as,

$$p'(x,t) = \sum_{j=1}^{N} a_j(t)\ sin(j\pi x)$$

where,

$$a_j(t) = -\frac{1}{j\pi} \dot{\eta}_j(t)$$

Substituting p'(x,t) = $\sum_{i=1}^{N} -\frac{1}{j\pi}\dot{\eta_j}(t)sin\,(j\pi x)$ in the non dimensionalised linearised momentum equation (1) , we get

$$u'(x,t) = \sum_{j=1}^{N} \eta_j(t)\ cos\,(j\pi x)$$

So, from these we have $\frac{\partial p'(x,t)}{\partial t} = \sum_{i=1}^{N} -\frac{1}{j\pi}\frac{\mathrm{d}\dot{\eta_j}(t)}{\mathrm{d}t}sin\,(j\pi x)$ and $\frac{\partial u'(x,t)}{\partial x} = \sum_{i=1}^{N} -j\pi\eta_j(t)\ cos\,(j\pi x)$,

which we substitute in the non dimensionalised linearised energy equation (2) ,
we get

$$\sum_{j=1}^{N} -\frac{1}{j\pi}\frac{\mathrm{d}\dot{\eta_j}(t)}{\mathrm{d}t}sin\,(j\pi x) + \sum_{i=1}^{N} -j\pi\eta_j(t)\ sin\,(j\pi x) = k\left[\sqrt{\left|\tfrac{1}{3} + u'(t-\tau)\right|} - \sqrt{\tfrac{1}{3}}\right]\delta(x-x_f)$$

Multiplying both sides by $j\pi$ and shifting the summation terms to the left hand side, we get,

$$\sum_{j=1}^{N}\left[\frac{\mathrm{d}\dot{\eta_j}(t)}{\mathrm{d}t} + (-j\pi)^2\eta_j(t)\right]sin\,(j\pi x) = -j\pi k\left[\sqrt{\left|\tfrac{1}{3} + u'(t-\tau)\right|} - \sqrt{\tfrac{1}{3}}\right]\delta(x-x_f)$$

$k_j = j\pi$: non-dimensionalised wave number

3.3 Projection Using Galerkin Modes

To project onto n-th Galerkin mode, we multiply by $sin\,(n\pi x)$ and integrate over the domain [0,1]:

$$\int_{x=0}^{x=1}\sum_{j=1}^{N}\left[\frac{\mathrm{d}\dot{\eta_j}(t)}{\mathrm{d}t} + k_j^2\eta_j(t)\right]sin\,(j\pi x)\,sin\,(n\pi x)\ dx =$$
$$\int_{x=0}^{x=1} -j\pi k\left[\sqrt{\left|\frac{1}{3} + u'(t-\tau)\right|} - \sqrt{\frac{1}{3}}\right]\delta(x-x_f)sin\,(n\pi x)\ dx$$

We know that,

$$\int_0^1 sin\,(j\pi x)\,sin\,(n\pi x)\ dx = \tfrac{1}{2}\delta_{jn}$$
$$\int_0^1 f(x)\delta(x-x_f)\,dx = f(x_f)$$

Using the orthogonality property of sine functions and the sifting property of the delta function, we derive (for $n = j$):

$$\left[\frac{\mathrm{d}\dot{\eta_j}(t)}{\mathrm{d}t} + k_j^2\eta_j(t)\right] = -j\pi k\left[\sqrt{\left|\tfrac{1}{3} + u_f'(t-\tau)\right|} - \sqrt{\tfrac{1}{3}}\right]sin\,(j\pi x_f)$$

3.4 Incorporating Damping

Adding a frequency-dependent damping term:

$$\frac{\mathrm{d}\dot{\eta_j}(t)}{\mathrm{d}t} + 2\zeta_j\omega_j\dot{\eta_j} + k_j^2\eta_j(t) = -j\pi k\left[\sqrt{\left|\frac{1}{3} + u_f'(t-\tau)\right|} - \sqrt{\frac{1}{3}}\right]sin\,(j\pi x_f) \tag{3}$$

Frequency dependent damping : $\zeta_j = \frac{1}{2\pi}\left[c_1\frac{\omega_j}{\omega_1} + c_1\sqrt{\frac{\omega_1}{\omega_j}}\right]$

and

$$\frac{\mathrm{d}\eta_j(t)}{\mathrm{d}t} = \dot{\eta_j}(t) \tag{4}$$

These are the 2 final sets of equations that we obtain which we solve numerically.

3.5 Time Integration Scheme

For a general first-order differential equation of the form

$$\frac{\mathrm{d}y(t)}{\mathrm{d}t} = f(t,y),$$

with an initial condition $y(t_0) = y_0$, we aim to evolve the solution from $t = t_0$ to $t = T$.
The Taylor series expansion of $y(t)$ about the point t_0 is:

$$y(t_0 + h) = y(t_0) + hy'(t_0) + \frac{h^2}{2!}y''(t_0) + \cdots$$

Neglecting the higher-order terms gives the Euler method:

$$y(t_0 + h) = y(t_0) + hf(t_0, y_0)$$

This can be written as a basic iterative scheme:

$$y_{n+1} = y_n + hf(t_n, y_n)$$

In our case, the governing equations (3) and (4) are rewritten as:

$$\frac{\mathrm{d}\eta_j(t)}{\mathrm{d}t} = f(t, \eta_j(t), \dot{\eta}_j(t))$$

$$\frac{\mathrm{d}\dot{\eta}_j(t)}{\mathrm{d}t} = g(t, \eta_j(t), \dot{\eta}_j(t))$$

for $j = 1, 2, \ldots, n$ Galerkin modes.
This coupled system is expanded using Taylor series as outlined previously. By retaining terms up to fourth order, we apply the classical fourth-order Runge-Kutta (RK4) method, which offers significantly improved accuracy over the Euler method.

4 Numerical Methodology

4.1 Runge-Kutta Integration with Delay

The classical fourth-order Runge-Kutta (RK4) method is employed to numerically solve the system of ordinary differential equations resulting from Galerkin projection. This method is known for its balance between accuracy and computational efficiency, making it well-suited for time-domain simulations.
Consider a general first-order ODE:

$$\frac{dy}{dt} = f(t, y), \quad y(t_0) = y_0$$

where $y(t)$ is the unknown function, $f(t, y)$ is the derivative function, and y_0 is the initial condition at $t = t_0$.
To approximate the solution at $t_{n+1} = t_n + h$, where h is the time step size, RK4 calculates four intermediate slopes as follows:

Intermediate Slopes

$$\begin{aligned}
k_1 &= f(t_n, y_n), \\
k_2 &= f\left(t_n + \frac{h}{2}, y_n + \frac{h}{2}k_1\right), \\
k_3 &= f\left(t_n + \frac{h}{2}, y_n + \frac{h}{2}k_2\right), \\
k_4 &= f\left(t_n + h, y_n + hk_3\right)
\end{aligned}$$

Final Update Step

The next value y_{n+1} is then estimated using a weighted average of these slopes:

$$y_{n+1} = y_n + \frac{h}{6}(k_1 + 2k_2 + 2k_3 + k_4)$$

5 Simulation Workflow

The procedure for solving the coupled ODEs using the Runge-Kutta (RK4) method is outlined as follows:

1. Initialize the variables η_j and $\dot{\eta}_j$ as y_1 and y_2 respectively at $t = 0$.
2. Set the constant parameters: x_f, γ, and M.
3. For each time step Δt, apply the fourth-order Runge-Kutta method to compute the updated values of η_j and $\dot{\eta}_j$. These updated values are then used to evaluate the acoustic velocity fluctuation u'. For times $t < \tau$, the heat release source term is neglected due to the time lag; for $t \geq \tau$, the source term is included.
4. Record the root mean square velocity u_{rms} for each value of the heat source amplitude K and/or time lag τ, to be used in bifurcation analysis.
5. For bifurcation plots, the final values of η_j and $\dot{\eta}_j$ from the current simulation are used as the initial conditions for the subsequent simulation with a new value of K.
6. Various quantities such as u', η_j, acoustic energy, and bifurcation measures are plotted. From these plots, the necessary results (as presented in the paper) are obtained.

5.1 MATLAB Code

The described methodology was implemented in MATLAB. The scripts listed in Codes 9.1 and 9.2 contain the algorithm used to develop and solve the system model.

6 Results and Analysis

To understand how the system evolves under different conditions, we examine the effects of varying the control parameter (K) on the nondimensional acoustic velocity u' and the Galerkin mode amplitudes η_j.

6.1 Oscillation Behavior at Different K

Figure 3 shows the time evolution of the acoustic velocity fluctuation u' for the initial condition $\eta_1(0) = 0.15$, with $x_f = 0.29$ m, $K = 0.05$, and $\tau = 0.2$ s. The system parameters used are $c_1 = 0.1$ and $c_2 = 0.06$.

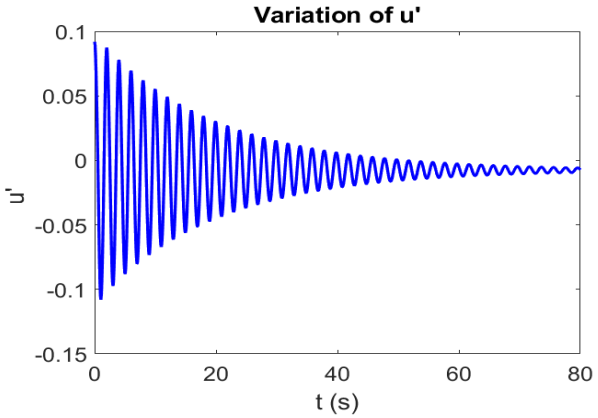
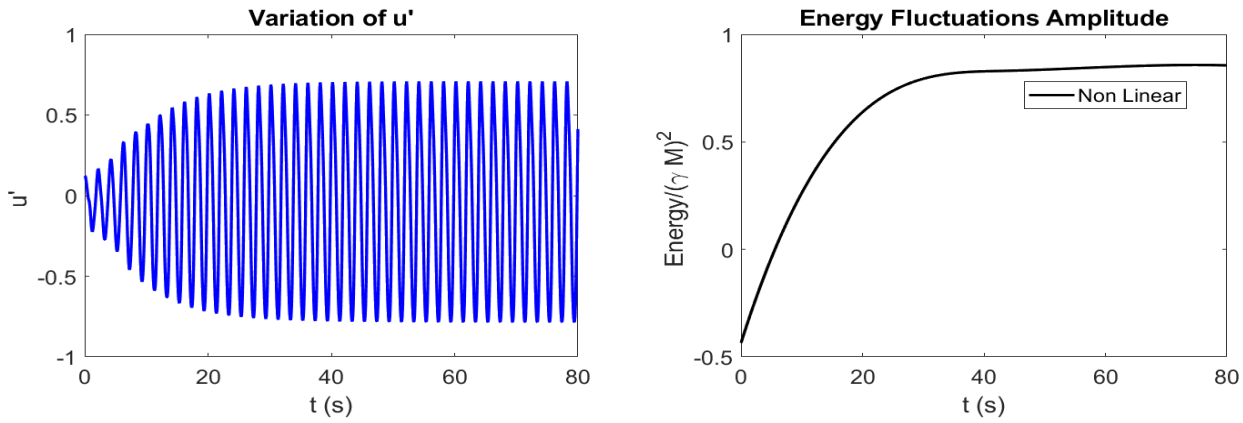


Figure 3: K = 0.05; $\tau = 0.2$; $\eta_1(0) = 0.15$, $\eta_{j\neq1}(0) = 0$, and $\dot{\eta}_j(0) = 0$

Under these conditions, the oscillations decay with time, primarily due to the damping term overpowering the driving influence of heat release. As the value of K increases, however, this balance shifts, and the system begins to exhibit persistent, large-amplitude oscillations.

Figure 4a illustrates the system response when the control parameters are modified to $K = 0.1$ and $\tau = 0.5$ s, while all other conditions are held constant. In this scenario, the heat release term becomes dominant, leading to the growth of oscillation amplitudes. Nonlinear effects inherent to the system eventually limit this growth.



(a) Velocity Fluctuations

(b) Energy Fluctuation Amplitude

Figure 4: K = 0.3; $\tau = 0.2$; $\eta_1(0) = 0.2$, $\eta_{j\neq1}(0) = 0$, and $\dot{\eta}_j(0) = 0$

In this case, the contribution of the heat source significantly amplifies the oscillatory behavior. However, the amplitude is eventually constrained by the system's nonlinear dynamics.

The non-dimensional acoustic energy is given by:

$$E = \frac{1}{(\gamma M)^2}\left(\frac{1}{2}p'^2 + \frac{1}{2}(\gamma M u')^2\right) \tag{5}$$

As the amplitude of oscillations increases in the second case, a corresponding rise in acoustic energy is observed. This trend is clearly reflected in Figure 4b.

6.2 Sensitivity to Time Delay (τ)

To gain deeper insight into the system's response, we analyze how the velocity fluctuation u' and the Galerkin modes $\eta_1(t)$, $\eta_2(t)$, and $\eta_3(t)$ evolve under varying values of K and τ. The outcomes of this sensitivity analysis are depicted in Figures 5 and 6.

The simulation results shown in Figures 5 and 6 reveal critical insights into the role of time delay τ in shaping the system's dynamical behavior.

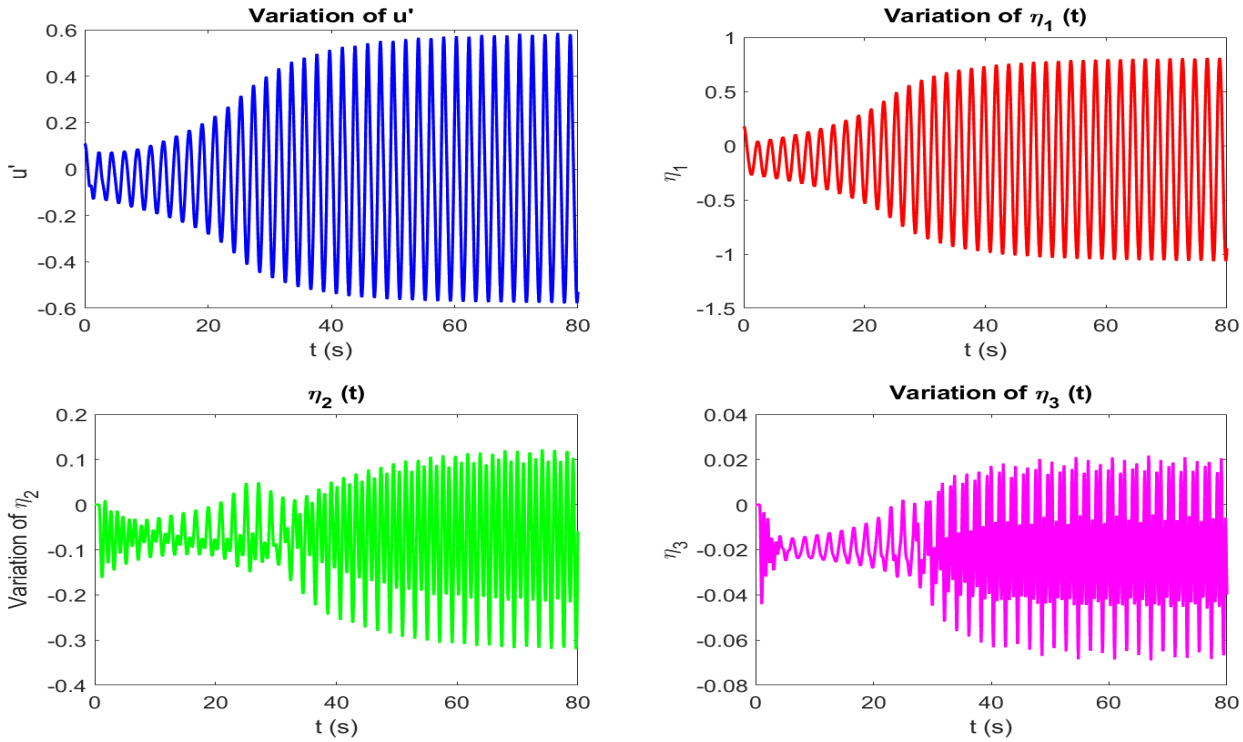


Figure 5: K = 0.25; $\tau = 0.75$; $\eta_1(0) = 0.18$, $\eta_{j\neq1}(0) = 0$, and $\dot{\eta}_j(0) = 0$

As seen in Figure 5, corresponding to $K = 0.25$ and $\tau = 0.75$, the amplitude of the acoustic velocity u' and the Galerkin modes $\eta_1(t), \eta_2(t), \eta_3(t)$ exhibits progressive growth over time. This behavior indicates that the energy input from the delayed heat release effectively amplifies acoustic oscillations, overcoming the inherent damping mechanisms. The system thus evolves towards a nonlinear limit cycle characterized by sustained large-amplitude oscillations. This energy build-up is a signature of thermoacoustic instability, where the phase relationship between the heat addition and pressure oscillations satisfies Rayleigh's criterion for positive feedback.

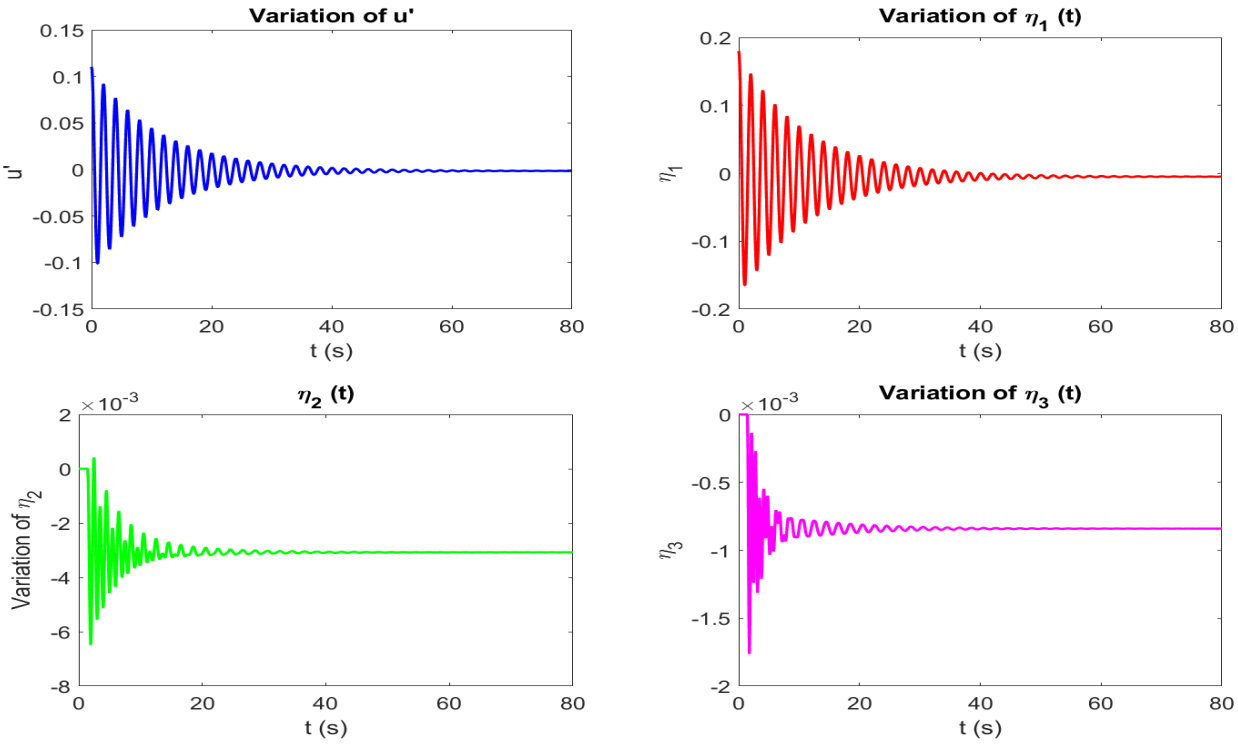


Figure 6: K = 0.01; $\tau = 0.45\pi$; $\eta_1(0) = 0.18$, $\eta_{j\neq 1}(0) = 0$, and $\dot{\eta}_j(0) = 0$

Conversely, Figure 6 illustrates the case with a much lower heater power $K = 0.01$ and a moderate delay $\tau = 0.45\pi$. Here, both the acoustic velocity and the Galerkin mode amplitudes decay steadily with time. The damping forces dominate the weak driving effect of heat release, preventing the initiation of a limit cycle. As a result, the system stabilizes to its quiescent state with negligible oscillations. This showcases the stabilizing effect of insufficient coupling strength or unfavorable phase timing in systems with delayed feedback.

Overall, the contrasting behaviors demonstrate the delicate balance between energy injection (through heat release) and energy dissipation (through damping). Small changes in either the time delay τ or the control parameter K can fundamentally alter the system's stability, emphasizing the importance of nonlinear time-delayed modeling in accurately predicting thermoacoustic instabilities.

6.3 Bifurcation Structure

The control parameter K is systematically varied from 0.2 to 1.4, while keeping $\tau = 0.2$, $c_1 = 0.1$, and $c_2 = 0.06$ constant. Figure 7 illustrates the variation of $|u'|$, representing the non-dimensional acoustic velocity for a single Galerkin mode, as a function of K.

For values of $K < 0.9$, the oscillations decay over time, indicating a stable system. Beyond this point, the amplitude begins to grow, signaling the onset of acoustic instability. This marks the occurrence of a **Hopf bifurcation**, where a sudden transition to periodic oscillations is observed. Specifically, this corresponds to a subcritical Hopf bifurcation. Moreover, as K is decreased from 1.4, the system exhibits hysteresis near the bifurcation threshold, evidenced by the green markers in Figure 7.

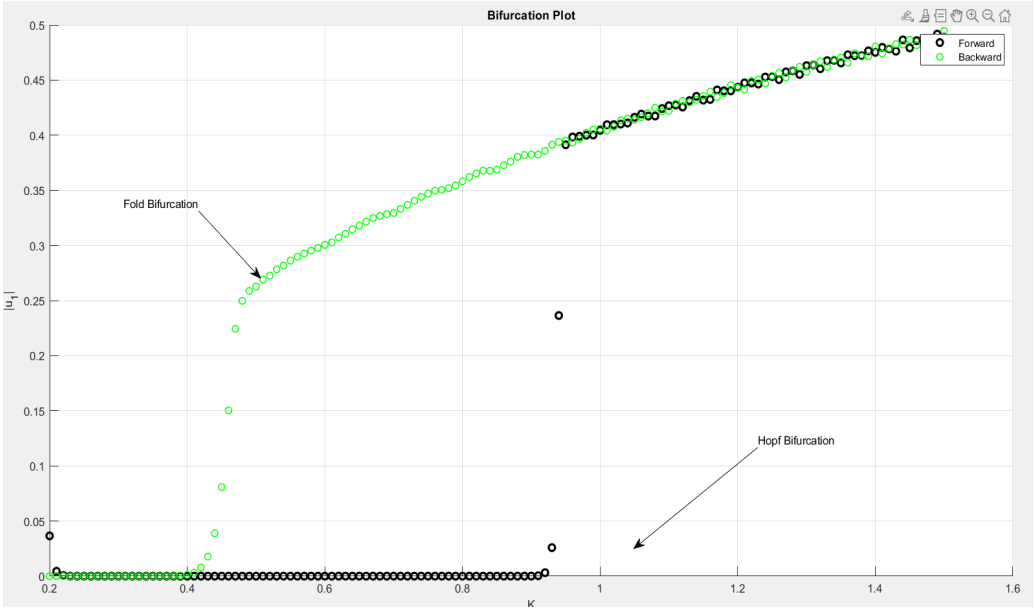


Figure 7: Bifurcation plot showing variation of non-dimensional acoustic velocity with heater power K

To further understand the system's behavior, a bifurcation diagram is also constructed by varying the time delay τ while retaining a single Galerkin mode. Figure 8 presents a 3D bifurcation plot that captures how $|u'|$ changes with both K and τ. For this analysis, the system parameters are fixed at $c_1 = 0.1$, $c_2 = 0.06$, and $x_f = 0.30\,\text{m}$.

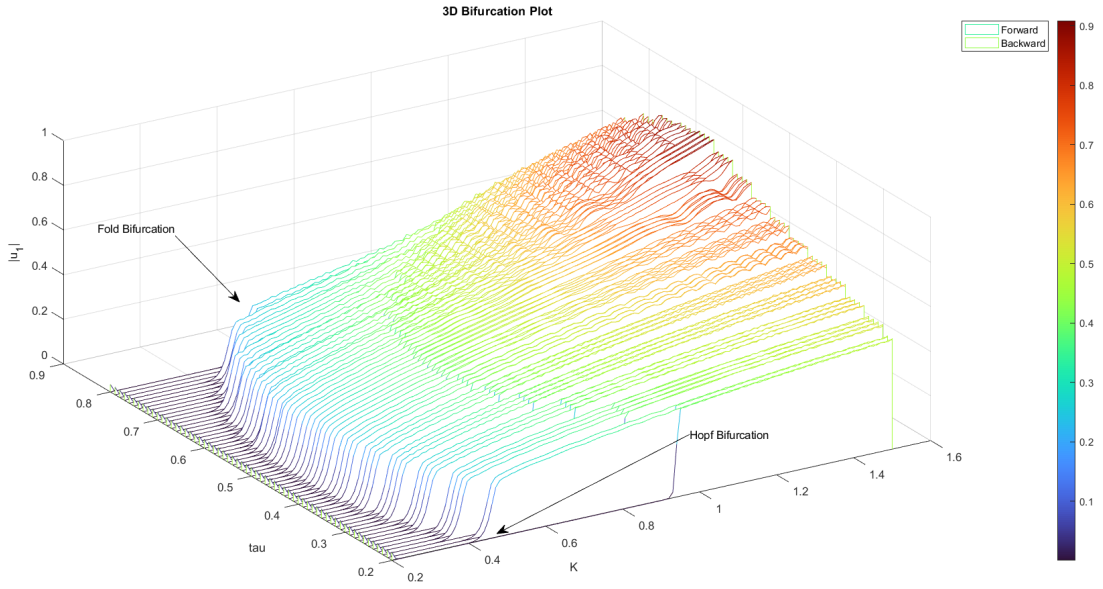


Figure 8: 3D Bifurcation plot showing dependence of $|u'|$ on K and τ

7 Conclusion

Understanding the onset of thermoacoustic instability in time-delayed systems requires tools that go beyond traditional linear analysis. One such approach—numerical continuation—was recently used to study a reduced-order model of the horizontal Rijke tube, a fundamental system in combustion dynamics.

Rather than relying on transient simulations, this method systematically tracked the evolution of steady and oscillatory solutions as system parameters were varied. The study revealed critical nonlinear features, including sudden transitions to limit cycles and hidden instability zones, governed by subcritical Hopf bifurcations. These findings highlight the strength of continuation techniques in exposing the intricate dynamics of thermoacoustic systems.

8 References

1. **Balasubramanian and Sujith (2008)**
   Thermoacoustic instability in a Rijke tube: Non-normality and nonlinearity.
   https://pubs.aip.org/aip/pof/article/20/4/044103/257507/
   Thermoacoustic-instability-in-a-Rijke-tube-Non

2. **Subramanian et al. (2010)**
   Bifurcation analysis of thermoacoustic instability in a horizontal Rijke tube.
   https://www.researchgate.net/publication/265794610_Bifurcation_Analysis_of_
   Thermoacoustic_Instability_in_a_Horizontal_Rijke_Tube

3. *Runge Kutta Integration*.
   https://www.youtube.com/watch?v=HOWJp8NV5xU
   https://www.youtube.com/watch?v=vNoFdtcPFdk

9 Appendix

9.1 Acoustic Velocity and η_j (time-dependent amplitude of the j-th Galerkin mode)

```
%% Simulation Parameters
N = 1000;       % number of timesteps
T = 80;         % total time of experiment
dt = (T/N);     % time step

%% System Parameters
g = 1.4;        % specific heat ratio
c_0 = 399.6;    % speed of sound
u_0 = 0.5;      % mean velocity of air
M = u_0/c_0;    % Mach number

%% Configurable Parameters
%% K values
%K = 0.05;       % fig4a
%K = 0.3;        % fig4b, 4d
%K = 0.25;       % fig5a, 5b, 5c, 5d
K = 0.01;       % fig6a, 6b, 6c, 6d

x_f = 0.29;     % flame position

%% Tau values
%tau = 0.2;      % fig4a
%tau = 0.5;      % fig4b, 4d,
%tau = 0.75;     % fig5a, 5b, 5c, 5d
tau = 0.45*pi;  % fig6a, 6b, 6c, 6d

%% Initial Conditions for y1 (only one of three)
%y1_init = 0.15;  % fig 4a
%y1_init = 0.2;   % fig4b, 4d
y1_init = 0.18;   % fig5a, 5b, 5c, 5d, fig6a, 6b, 6c, 6d

% Calculate delay index
n1 = round(tau/dt);

%% Initialize Arrays
J = 3;              % number of Galerkin modes
u = zeros(1, N);
p = zeros(1, N);
y1 = zeros(J, N);
y2 = zeros(J, N);

%% Set Initial Conditions
y1(:,1) = 0;
y1(1,1) = y1_init; % Use selected initial condition
y2(:,1) = 0;
u(1) = y1(1,1)*cos(pi*x_f);
p(1) = 0;

%% Precompute constants for equations
kj = (1:J)'*pi;
wj = kj;
w1 = pi;
c1 = 0.1;
c2 = 0.06;
zetaj = (1/(2*pi))*(c1*(wj/w1) + c2*sqrt(w1./wj));

%% Precompute mode shapes
cos_j_pi_x_f = cos((1:J)'*pi*x_f);
sin_j_pi_x_f = sin((1:J)'*pi*x_f);

%% Time Integration
% First part: before delay (t < tau)
for n = 1:n1
    for j = 1:J
        % RK4 integration steps
        [k1, l1] = equations1(y1(j,n), y2(j,n), j, zetaj(j));
        [k2, l2] = equations1(y1(j,n) + 0.5*dt*k1, y2(j,n) + 0.5*dt*l1, j, zetaj(j));
        [k3, l3] = equations1(y1(j,n) + 0.5*dt*k2, y2(j,n) + 0.5*dt*l2, j, zetaj(j));
```

```
        [k4, l4] = equations1(y1(j,n) + dt*k3, y2(j,n) + dt*l3, j, zetaj(j));

        y1(j,n+1) = y1(j,n) + (dt/6) * (k1 + 2*k2 + 2*k3 + k4);
        y2(j,n+1) = y2(j,n) + (dt/6) * (l1 + 2*l2 + 2*l3 + l4);
    end

    % Calculate u and p for this timestep
    u(n+1) = sum(y1(:,n+1).*cos_j_pi_x_f);
    p(n+1) = sum(y2(:,n+1).*(-g*M./kj).*sin_j_pi_x_f);
end

% Second part: after delay (t > tau)
for n = n1+1:N-1
    for j = 1:J
        % RK4 integration steps
        [k1, l1] = equations2(y1(j,n), y2(j,n), j, K, u(n-n1), x_f, zetaj(j));
        [k2, l2] = equations2(y1(j,n) + 0.5*dt*k1, y2(j,n) + 0.5*dt*l1, j, K, u(n-n1), x_f,
            zetaj(j));
        [k3, l3] = equations2(y1(j,n) + 0.5*dt*k2, y2(j,n) + 0.5*dt*l2, j, K, u(n-n1), x_f,
            zetaj(j));
        [k4, l4] = equations2(y1(j,n) + dt*k3, y2(j,n) + dt*l3, j, K, u(n-n1), x_f, zetaj(j));

        y1(j,n+1) = y1(j,n) + (dt/6) * (k1 + 2*k2 + 2*k3 + k4);
        y2(j,n+1) = y2(j,n) + (dt/6) * (l1 + 2*l2 + 2*l3 + l4);
    end

    % Calculate u and p for this timestep
    u(n+1) = sum(y1(:,n+1).*cos_j_pi_x_f);
    p(n+1) = sum(y2(:,n+1).*(-g*M./kj).*sin_j_pi_x_f);
end

%% Functions
function [f1, f2] = equations1(y1, y2, j, zetaj)
    kj = j*pi;
    wj = kj;

    f1 = y2;
    f2 = -(kj^2) * y1 - 2*zetaj*wj*y2;
end

function [f1, f2] = equations2(y1, y2, j, K, u, x_f, zetaj)
    kj = j*pi;
    wj = kj;

    f1 = y2;
    f2 = -(kj^2) * y1 - 2*zetaj*wj*y2 - ((2*j*pi*K))*(sqrt(abs((1/3) + u))/sqrt(1/3))*sin(j*pi*
        x_f);
end
```

9.2 Bifurcation

```
% Simulation Parameters
N = 1000;       % number of timesteps
T = 40;         % total time of experiment
dt = (T/N);     % time step
x_f = 0.3;       % flame location

% System Parameters
g = 1.4;
c_0 = 399.6;     % speed of sound
u_0 = 0.5;       % mean velocity of air
M = u_0/c_0;      % Mach number
J = 3;          % number of Galerkin modes

% Parameter Ranges
ks = 0.2:0.01:1.5;      % non-dimensional heater power
tau = 0.2:0.01:0.8;     % time-lag

% Preallocate Arrays
n1 = length(ks);
n3 = length(tau);
```

```
u_rms1 = zeros(n3, n1);
u_rms2 = zeros(n3, n1);
y1 = zeros(n3, J, N);
y2 = zeros(n3, J, N);

% Forward Simulation
for i = 1:n3
    y1(i, 1, 1) = 0.18;  % Initial condition
    u1 = zeros(n1, N);

    for k = 1:n1
        % Initial velocity calculation
        for j = 1:J
            u1(k, 1) = u1(k, 1) + y1(i, j, 1) * cos(j * pi * x_f);
        end

        n2 = round(tau(i) / dt);

        % Fourth-order Runge-Kutta method for t < tau
        for n = 1:n2
            for j = 1:J
                [k1, l1] = equations1(y1(i, j, n), y2(i, j, n), j);
                [k2, l2] = equations1(y1(i, j, n) + 0.5 * dt * k1, y2(i, j, n) + 0.5 * dt * l1,
                    j);
                [k3, l3] = equations1(y1(i, j, n) + 0.5 * dt * k2, y2(i, j, n) + 0.5 * dt * l2,
                    j);
                [k4, l4] = equations1(y1(i, j, n) + dt * k3, y2(i, j, n) + dt * l3, j);

                y1(i, j, n + 1) = y1(i, j, n) + (dt / 6) * (k1 + 2 * k2 + 2 * k3 + k4);
                y2(i, j, n + 1) = y2(i, j, n) + (dt / 6) * (l1 + 2 * l2 + 2 * l3 + l4);

                u1(k, n + 1) = u1(k, n + 1) + y1(i, j, n + 1) * cos(j * pi * x_f);
            end
        end

        % Fourth-order Runge-Kutta method for t > tau
        for n = n2 + 1:N - 1
            for j = 1:J
                [k1, l1] = equations2(y1(i, j, n), y2(i, j, n), j, ks(k), u1(k, n - n2), x_f);
                [k2, l2] = equations2(y1(i, j, n) + 0.5 * dt * k1, y2(i, j, n) + 0.5 * dt * l1,
                    j, ks(k), u1(k, n - n2), x_f);
                [k3, l3] = equations2(y1(i, j, n) + 0.5 * dt * k2, y2(i, j, n) + 0.5 * dt * l2,
                    j, ks(k), u1(k, n - n2), x_f);
                [k4, l4] = equations2(y1(i, j, n) + dt * k3, y2(i, j, n) + dt * l3, j, ks(k),
                    u1(k, n - n2), x_f);

                y1(i, j, n + 1) = y1(i, j, n) + (dt / 6) * (k1 + 2 * k2 + 2 * k3 + k4);
                y2(i, j, n + 1) = y2(i, j, n) + (dt / 6) * (l1 + 2 * l2 + 2 * l3 + l4);

                u1(k, n + 1) = u1(k, n + 1) + y1(i, j, n + 1) * cos(j * pi * x_f);
            end
        end

        % Update initial conditions and calculate RMS
        y1(i, :, 1) = y1(i, :, N);
        y2(i, :, 1) = y2(i, :, N);
        u_rms1(i, k) = rms(u1(k, :));
    end
end

% Backward Simulation
for i = 1:n3
    u2 = zeros(n1, N);

    for k = 1:n1
        % Initial velocity calculation (fixed typo with J instead of j)
        for j = 1:J
            u2(n1 - k + 1, 1) = u2(n1 - k + 1, 1) + y1(i, j, 1) * cos(j * pi * x_f);
        end

        n2 = round(tau(i) / dt);

        % Fourth-order Runge-Kutta method for t < tau
```

```
        for n = 1:n2
            for j = 1:J
                [k1, l1] = equations1(y1(i, j, n), y2(i, j, n), j);
                [k2, l2] = equations1(y1(i, j, n) + 0.5 * dt * k1, y2(i, j, n) + 0.5 * dt * l1,
                    j);
                [k3, l3] = equations1(y1(i, j, n) + 0.5 * dt * k2, y2(i, j, n) + 0.5 * dt * l2,
                    j);
                [k4, l4] = equations1(y1(i, j, n) + dt * k3, y2(i, j, n) + dt * l3, j);

                y1(i, j, n + 1) = y1(i, j, n) + (dt / 6) * (k1 + 2 * k2 + 2 * k3 + k4);
                y2(i, j, n + 1) = y2(i, j, n) + (dt / 6) * (l1 + 2 * l2 + 2 * l3 + l4);

                u2(n1 - k + 1, n + 1) = u2(n1 - k + 1, n + 1) + y1(i, j, n + 1) * cos(j * pi *
                    x_f);
            end
        end

        % Fourth-order Runge-Kutta method for t > tau
        for n = n2 + 1:N - 1
            for j = 1:J
                [k1, l1] = equations2(y1(i, j, n), y2(i, j, n), j, ks(n1 - k + 1), u2(n1 - k +
                    1, n - n2), x_f);
                [k2, l2] = equations2(y1(i, j, n) + 0.5 * dt * k1, y2(i, j, n) + 0.5 * dt * l1,
                    j, ks(n1 - k + 1), u2(n1 - k + 1, n - n2), x_f);
                [k3, l3] = equations2(y1(i, j, n) + 0.5 * dt * k2, y2(i, j, n) + 0.5 * dt * l2,
                    j, ks(n1 - k + 1), u2(n1 - k + 1, n - n2), x_f);
                [k4, l4] = equations2(y1(i, j, n) + dt * k3, y2(i, j, n) + dt * l3, j, ks(n1 -
                    k + 1), u2(n1 - k + 1, n - n2), x_f);

                y1(i, j, n + 1) = y1(i, j, n) + (dt / 6) * (k1 + 2 * k2 + 2 * k3 + k4);
                y2(i, j, n + 1) = y2(i, j, n) + (dt / 6) * (l1 + 2 * l2 + 2 * l3 + l4);

                u2(n1 - k + 1, n + 1) = u2(n1 - k + 1, n + 1) + y1(i, j, n + 1) * cos(j * pi *
                    x_f);
            end
        end

        % Update initial conditions and calculate RMS
        y1(i, :, 1) = y1(i, :, N);
        y2(i, :, 1) = y2(i, :, N);
        u_rms2(i, n1 - k + 1) = rms(u2(n1 - k + 1, :));
    end
end

% Plotting
% Bifurcation plot for tau = 0.2
for i = 1:n3
    if abs(tau(i) - 0.2) < 1e-6
        figure(1);
        scatter(ks, u_rms1(i, :), 'MarkerEdgeColor', 'k', 'LineWidth', 2);
        hold on;
        scatter(ks, u_rms2(i, :), 'MarkerEdgeColor', 'g', 'LineWidth', 1);
        xlabel('K');
        ylabel('|u_1|');
        legend('Forward', 'Backward')
        title('Bifurcation Plot')
        annotation('textarrow', [0.7 0.6], [0.3 0.15], 'String', 'Hopf Bifurcation')
        annotation('textarrow', [0.25 0.3], [0.65 0.55], 'String', 'Fold Bifurcation')
        grid on;
    end
end

% 3D Bifurcation plot
figure(2);
waterfall(ks, tau, u_rms1);
hold on;
waterfall(ks, tau, u_rms2);
colormap turbo
xlabel('K')
ylabel('tau');
zlabel('|u_1|')
legend('Forward', 'Backward')
title('3D Bifurcation Plot')
```

```
annotation('textarrow', [0.6 0.45], [0.3 0.15], 'String', 'Hopf Bifurcation')
annotation('textarrow', [0.2 0.25], [0.6 0.5], 'String', 'Fold Bifurcation')
colorbar

% Nested Functions
function [f1, f2] = equations1(y1, y2, j)
    kj = j * pi;
    wj = kj;
    w1 = pi;
    c1 = 0.1;
    c2 = 0.06;

    zetaj = (1 / (2 * pi)) * (c1 * (wj / w1) + c2 * sqrt(w1 / wj));

    f1 = y2;
    f2 = -2 * zetaj * wj * y2 - (kj^2) * y1;
end

function [f1, f2] = equations2(y1, y2, j, K, u, x_f)
    kj = j * pi;
    wj = kj;
    w1 = pi;
    c1 = 0.1;
    c2 = 0.06;

    zetaj = (1 / (2 * pi)) * (c1 * (wj / w1) + c2 * sqrt(w1 / wj));

    f1 = y2;
    f2 = -2 * zetaj * wj * y2 - (kj^2) * y1 - j * pi * K * (abs(sqrt((1/3) + u)) - sqrt(1/3)) *
         sin(j * pi * x_f);
end
```