Department Of Aerospace Engineering
IIT MADRAS

AS6320

AE21B001
Abel Viji George

Evolution of Non-dimensional Acoustic Velocity, Energy and Bifurcation Analysis in a Horizontal Rijke Tube

Contents

List of Figures

1 Introduction

1.1 Overview of Thermoacoustic instabilities

Constituting one of the most challenging fields of combustion research, thermoacoustic instabilities refer to undesirable, large amplitude oscillations of one or more natural acoustic modes of a system arising from resonant interaction between oscillatory flow and unsteady heat release processes. The pressure oscillations can reach up to a significant percentage of the mean static pressure. If not properly taken care of these instabilities cause a lot of problems like component vibrations, increased heat transfer rates, flame blow-off and flashback. The high amplitude pressure oscillations can even cause structural damages.

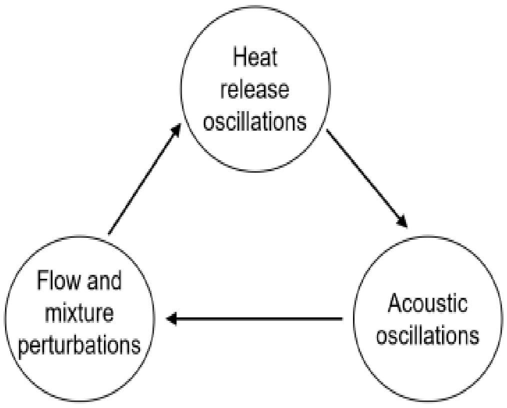


Figure 1: Feedback mechanism for Thermoacoustic Instabilities ([2])

The above figure shows the feedback mechanism for Thermoacoustic instabilities. Perturbations of the flow are generated through a driving process which produce heat release oscillations which thereby induce fluctuations in acoustic pressure and velocity. The acoustic oscillations in turn generate further flow perturbations, thus closing the feedback loop. Unsteady combustion is an efficient source of acoustic energy. Since combustors are highly resonant systems, acoustic waves generated by the unsteady combustion are reflected from the walls of the combustor to generate unsteadiness in flow near the flame.
Even complex systems like rocket motors, jet engines and gas turbines are not free from acoustic instabilities. Fluctuations in heat release arise due to acoustic fluctuations. They act as a source of acoustic fluctuations and the coupled system can reach self-sustained large amplitude oscillations. Combustion instability is caused when the fluctuating heat release rate and the acoustic field form a positive feedback loop [3]. Triggering was discovered in solid rockets and can cause a system which was linearly stable to become unstable in the presence of a finite amplitude disturbance.

1.2 Model of the Rijke Tube

The Rijke tube used resembles the one used in [3] and [1].

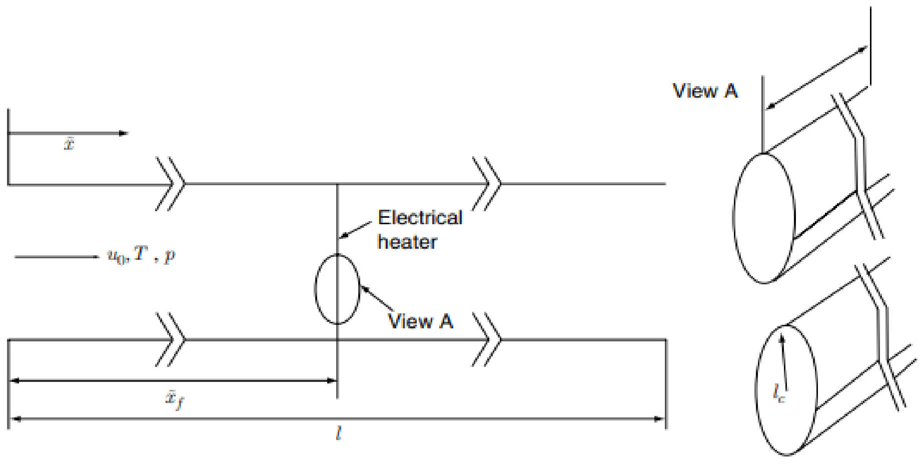


Figure 2: Configuration of a horizontal Rijke tube with an electric heater as source [3]

In this model, x is the distance along the axial direction, x_f is the location of flame in the axial direction, l is the length of the duct and t is time. The flow has a steady state velocity u_0, pressure $\bar{p}$ and temperature $\bar{T}$ with u' as acoustic velocity and p' as acoustic pressure. When γ is the ratio of specific heats of the medium, the speed of sound is calculated as c_0 and M is the Mach number of the mean flow. Additionally, ζ is the damping coefficient and $\dot{Q}'$ is the heat release rate fluctuations per unit area due to the electrical heater.

1.3 Objective

The following are the objectives for this study:

- To qualitatively capture the evolution of non-dimensional acoustic velocity with time and its dependence on parameters like Heater power, K, time lag, τ etc. How damping affects the fluctuations is also studied.
- To understand how the energy of the system respond to increasing K and how projecting the velocity onto multiple modes affect the energy distribution of the system.
- To qualitatively capture the evolution of velocity projected onto various Galerkin modes and how they respond, i.e, grow or decay by varying certain parameters.
- To obtain the hysteresis through a bifurcation plot of the RMS velocity and Heater power value by increasing and subsequently decreasing the K value and to observe the Hopf and Fold Bifurcations in the plot.

1.4 Motivation for studying the present problem

Combustion instabilities are self-sustained large amplitude oscillations of pressure and velocity in combustors with the flame acting as an acoustic actuator and the combustion chamber as an acoustic resonator. Several existing models already use linear stability analysis to capture the spontaneous pulsations arising in systems like Rijke tubes.
It is possible that a linearly stable combustor could be "triggered" by small disturbances. There could be instances of "bootstrapping" where a an initially decaying model can later grow and become unstable. These instances cannot be captured by a linear stability analysis which would indicate that the system would be stable and miss the ultimate behaviour. These reasons force us to use non-linear analysis [1].
Identifying the key parameters controlling non-linear flame dynamics is an difficult task with a wide array of possibilities and uncertainties. The difficulty is compounded due to the complex nature of combustion-acoustic interaction in flames. This led to interest in simpler thermoacoustic systems such as the Rijke tubes, like the one we used in our case.
We also see the role of introducing a damping constant in the overall behaviour of the system as used in [3] using a Galerkin type analysis. Stable, unstable and bi-stability regions are observed from bifurcation plots for varying parameters such as non-dimensional Heater power, K, time lag, τ and a few others. The results obtained from such analyses could be implemented in real life scenarios like combustors whereby we know how changing certain parameters like Heater power or time delay affect the overall behaviour of the system.

2 Methodology

2.1 Governing Equations

We will use the linearized 1-D momentum and energy equations after non-dimensionalizing them.
We use the following parameters for non-dimensionalization:

$$\tilde{x} = L_a x, \quad \tilde{t} = \frac{L_a}{c_0} t,$$
$$\tilde{u}' = u_0 u', \quad \tilde{p}' = \overline{p} p', \quad M = \frac{u_0}{c_0}.$$

The quantities with tilde (̃) are dimensional quantities and those without it are non-dimensional quantities. L_a is the duct length, c_0 is the speed of sound, $\overline{p}$ is the pressure in the undisturbed medium and u_0 is the mean flow velocity.
We also know the relation:

$$\overline{\rho} = \frac{\gamma \overline{p}}{c_0^2}$$

Linearized Momentum Equation

$$\rho\frac{\partial \tilde{u}'}{\partial \tilde{t}} + \frac{\partial \tilde{p}'}{\partial \tilde{x}} = 0$$

Non-dimensionalising:

$$\frac{\gamma \overline{p}}{c_0^2}\frac{\partial(u_0 u')}{\partial(\frac{L_a}{c_0}t)} + \frac{\partial(\overline{p}p')}{\partial(L_a x)} = 0$$

which gives:

$$\gamma M\frac{\partial u'}{\partial t} + \frac{\partial p'}{\partial x} = 0 \tag{2.1}$$

Energy Equation

$$\frac{\partial \tilde{p}'}{\partial \tilde{t}} + \gamma\overline{p}\frac{\partial \tilde{u}'}{\partial \tilde{x}} = (\gamma-1)\dot{\tilde{Q}}'$$

Non-dimensionalising:

$$\frac{\partial(\overline{p}p')}{\partial(\frac{L_a}{c_0}t)} + \gamma\overline{p}\frac{\partial(u_0 u')}{\partial(L_a x)} = (\gamma-1)\dot{\tilde{Q}}'$$

which gives,

$$\frac{\partial p'}{\partial t} + \gamma M\frac{\partial u'}{\partial x} = \frac{L_a}{c_0}\frac{(\gamma-1)\dot{\tilde{Q}}'\gamma}{\overline{\rho}c_0^2} \tag{2.2}$$

As mentioned in [1], a modified form of King's law is used to model the heat release rate. The empirical model suggested by Heckl is used where using a factor of $\frac{1}{3}$ for the u_0 seemed to work.

$$\dot{\tilde{Q}}' = \frac{2L_w(T_w-\overline{T})}{S\sqrt{3}}\sqrt{\pi\lambda C_v\overline{\rho}\frac{d_w}{2}}\left[\sqrt{\left|\frac{u_0}{3}+\tilde{u}'_f(t-\tau)\right|}-\sqrt{\frac{u_0}{3}}\right]\delta_D(\tilde{x}-\tilde{x}_f) \tag{2.3}$$

where, δ_D is the dirac delta function and we can use the following property,

$$\delta_D(\tilde{x}-\tilde{x}_f) = \delta_D(L_a(x-x_f)) = \frac{1}{L_a}\delta_D(x-x_f)$$

along with $\tilde{u_f} = u_0 u'_f$.
After adding a damping term, $\zeta p'$ and using equation (2.3) we can rewrite equation (2.2) as,

$$\frac{\partial p'}{\partial t} + \gamma M\frac{\partial u'}{\partial x} + \zeta p' = (\gamma-1)\frac{2L_w(T_w-\overline{T})}{S\sqrt{3}c_0\overline{p}}\sqrt{\pi\lambda C_v\overline{\rho}\frac{d_w}{2}u_0}\left[\sqrt{\left|\frac{1}{3}+u'_f(t-\tau)\right|}-\sqrt{\frac{1}{3}}\right]\delta_D(x-x_f) \tag{2.4}$$

We can define a constant,

$$K' = (\gamma-1)\frac{2L_w(T_w-\overline{T})}{S\sqrt{3}c_0\overline{p}}\sqrt{\pi\lambda C_v\overline{\rho}\frac{d_w}{2}u_0}$$

and rewrite equation (2.4) as,

$$\frac{\partial p'}{\partial t} + \gamma M\frac{\partial u'}{\partial x} + \zeta p' = K'\left[\sqrt{\left|\frac{1}{3}+u'_f(t-\tau)\right|}-\sqrt{\frac{1}{3}}\right]\delta_D(x-x_f) \tag{2.5}$$

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, ρ is the mean density of air, d_w is the diameter of the wire, $T_w - \overline{T}$ is the temperature difference, and S is the cross-sectional area of the duct. $L_w = 2.2m$, $d_w = 0.0005m$, $S = 1.8\times10^{-3}m^2$, $L_a = 1m$, $u_0 = 0.5m/s$, $\lambda = 0.0328W/(m.K)$, $\rho = 1.205Kg/m^3$ as is the case for a typical laboratory Rijke tube [4].

2.2 Galerkin Technique

We need to reduce equation (2.5) into ODEs in order to solve them. We use the Galerkin Technique to achieve this. The pressure field can be written in terms of the duct's natural modes,

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

As mentioned in [3], the Galerkin technique makes use of the fact that any function in a domain can be expressed as a superposition of expansion functions which form a complete basis in that domain. A sine function is chosen as the basis function since we are considering a duct open at both ends and we need pressure nodes at both ends.
We define,

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

where, $\dot{\eta}_j = \frac{d\eta_j}{dt}$.
Now we can write,

$$p' = -\sum_{j=1}^{N} \frac{\gamma M}{j\pi} sin(j\pi x)\dot{\eta}_j \tag{2.6}$$

Using equation (2.1), we can write,

$$\begin{aligned} \frac{\partial u'}{\partial t} &= -\frac{1}{\gamma M}\frac{\partial p'}{\partial x} \\ &= -\frac{1}{\gamma M}\sum_{j=1}^{N}\frac{\gamma M}{j\pi} cos(j\pi x)\dot{\eta}_j(t) \\ &= \sum_{j=1}^{N} cos(j\pi x)\dot{\eta}_j(t) \end{aligned}$$

Integrating with respect to time,

$$u' = \sum_{j=1}^{N} \eta_j(t) cos(j\pi x) \tag{2.7}$$

Substituting equations (2.6) and (2.7) into equation (2.5),

$$-\sum_{j=1}^{N}\frac{\gamma M}{j\pi} sin(j\pi x)\frac{d\dot{\eta}_j}{dt} - \gamma M\sum_{j=1}^{N} j\pi\eta_j(t) sin(j\pi x) - \zeta\sum_{j=1}^{N}\frac{\gamma M}{j\pi} sin(j\pi x)\dot{\eta}_j = K'\left[\sqrt{\left|\frac{1}{3} + u'_f(t-\tau)\right|} - \sqrt{\frac{1}{3}}\right]\delta_D(x - x_f) \tag{2.8}$$

To separate out the equation corresponding to each Galerkin Mode j, we have to take inner product with the basis function $sin(j\pi x)$.
Inner product of two functions f and g in interval $x \in [a, b]$ is defined as,

$$< f, g >= \int_a^b fg dx$$

We also use the (i) orthogonal property,

$$\int_0^1 sin(j\pi x) sin(n\pi x) dx = \frac{\delta_{jn}}{2}$$

where δ_{jn} is the Kronecker delta function defined as,

$$\delta_{jn} = \begin{cases} 1, & \text{if } j = n \\ 0, & \text{if } j \neq n \end{cases}$$

and the (ii) sifting property,

$$\int_0^1 f(x)\delta_D(x - x_f) = f(x_f)$$

Taking inner-product of all terms in equation (2.8) and using the above properties, we get,

$$-\sum_{j=1}^{N}\frac{\gamma M}{j\pi}\frac{d\dot{\eta}_j}{dt}\frac{\delta_{jn}}{2}-\gamma M\sum_{j=1}^{N}j\pi\eta_j\frac{\delta_{jn}}{2}-\zeta\sum_{j=1}^{N}\frac{\gamma M}{j\pi}\dot{\eta}_j\frac{\delta_{jn}}{2}=K'\left[\sqrt{\left|\frac{1}{3}+u'_f(t-\tau)\right|}-\sqrt{\frac{1}{3}}\right]sin(j\pi x_f)$$

We need n = j, for $\delta_{jn} = 1$. So using n = j in the above equation, we can isolate the equation for each Galerkin mode.

$$\frac{d\dot{\eta}_j}{dt}+\zeta\dot{\eta}_j+k_j^2\eta_j=-2Kj\pi\left[\sqrt{\left|\frac{1}{3}+u'_f(t-\tau)\right|}-\sqrt{\frac{1}{3}}\right]sin(j\pi x_f)$$

where, K = $\frac{K'}{\gamma M}$ and $\zeta = 2\omega_j\zeta_j$ is the frequency dependent damping and ζ_j is defined in [1] as,

$$\zeta_j=\frac{1}{2\pi}\left[c_1\frac{\omega_j}{\omega_1}+c_2\sqrt{\frac{\omega_1}{\omega_j}}\right]$$

The frequency is defined as,

$$\tilde{f}_J = j\frac{c}{2L}$$

for a time scale of $T = \frac{L}{C}$.
We can write the angular frequency, $\tilde{\omega}_J = 2\pi\tilde{f}_j$.
So, we can write,

$$\tilde{\omega}_j = 2\pi j\frac{c}{2L} = j\pi\frac{c}{L}$$

The non-dimensional angular frequency, ω_j can be written as,

$$\omega_j = \frac{L}{c}\tilde{\omega}_j = j\pi$$

k_j refers to the non-dimensional wave number and is equal to $j\pi$.

2.3 Methodology of Runge-Kutta fourth order method (RK-4)

Suppose we are given an initial value problem:

$$\frac{dy}{dx} = f(x,y) \quad with \quad y(x_0) = y_0$$

We pick the step size h>0 i.e, we divide x into small panels of width h. Then, the iterative steps are given by:

$$y_{i+1} = y_i + \frac{1}{6}(k_1 + 2k_2 + 2k_3 + k_4)$$

$$x_{i+1} = x_i + h$$

where,

- y_i is the RK4 approximation of $y(x_i)$.
- $k_1 = hf(x_i, y_i)$
- $k_2 = hf(x_i + \frac{h}{2}, y_i + \frac{k_1}{2})$
- $k_3 = hf(x_i + \frac{h}{2}, y_i + \frac{k_2}{2})$
- $k_4 = hf(x_i + h, y_i + k_3)$

The constants k_1 through k_4 mean the following:

- k_1 represents the slope at the beginning of the interval, x_i.
- k_2 represents the slope at the beginning of the interval, $x_i + \frac{h}{2}$, using the slope k_1 to estimate.
- k_3 represents the slope at the beginning of the interval, $x_i + \frac{h}{2}$, using the slope k_2 to estimate.
- k_4 represents the slope at the end of the interval, x_{i+1}, using the slope k_3 to estimate.

2.4 Numerical Solution

We have the system of first order ODEs,

$$\frac{d\eta_j}{dt} = \dot{\eta}_j$$
$$\frac{d\dot{\eta}_j}{dt} = -2\zeta_j\omega_j\dot{\eta}_j - k_j^2\eta_j - \frac{2Kj\pi}{\gamma M}\left[\sqrt{\left|\frac{1}{3} + u'_f(t-\tau)\right|} - \sqrt{\frac{1}{3}}\right] sin(j\pi x_f)$$

We would like to apply the RK-4 algorithm to this system of first order ODEs. The only issue preventing us from doing so is the term $u'_f(t-\tau)$. The time delay needs to be properly addressed.
Suppose, we have a time interval $t \in [0, T]$ and a time delay τ. We keep $u'_f(t-\tau) = 0$ for $t < \tau$.
So we have to apply RK-4 method separately to the two cases.

1. $t < \tau$:

$$\frac{d\eta_j}{dt} = \dot{\eta}_j$$
$$\frac{d\dot{\eta}_j}{dt} = -2\zeta_j\omega_j\dot{\eta}_j - k_j^2\eta_j$$

2. $t > \tau$:

$$\frac{d\eta_j}{dt} = \dot{\eta}_j$$
$$\frac{d\dot{\eta}_j}{dt} = -2\zeta_j\omega_j\dot{\eta}_j - k_j^2\eta_j - \frac{2Kj\pi}{\gamma M}\left[\sqrt{\left|\frac{1}{3} + u'_f(t-\tau)\right|} - \sqrt{\frac{1}{3}}\right] sin(j\pi x_f)$$

Using equation (2.7) at $x = x_f$,

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

We can find η and $\dot{\eta}$ for every mode at a particular time by marching in the Galerkin modes and use these to find the corresponding value of u'_f at that time instant. Then we can march in time and similarly find for all time instants.

3 Results

1. Case where damping is not considered with K = 0.01, $x_f = 0.29$, $\tau = 0.5$ s, $\eta_1(0) = 0.15$, $\eta_j(0) = 0$ for j $\neq 1$ and $\dot{\eta}_j(0) = 0$ for j = 1,2,...,N.

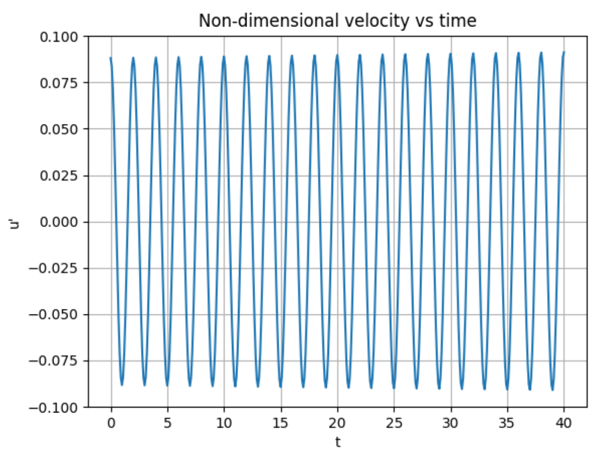
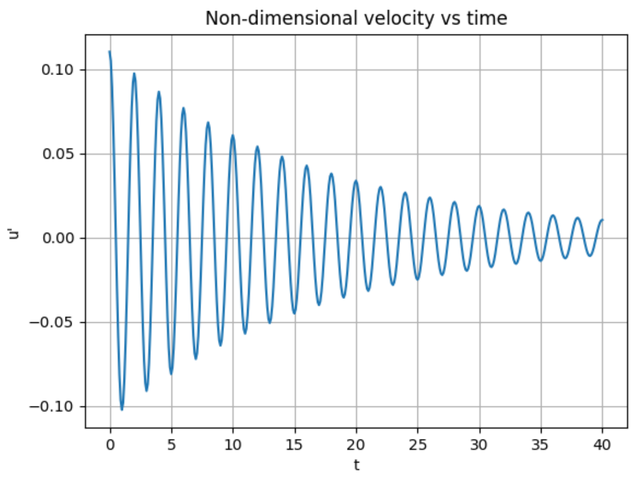
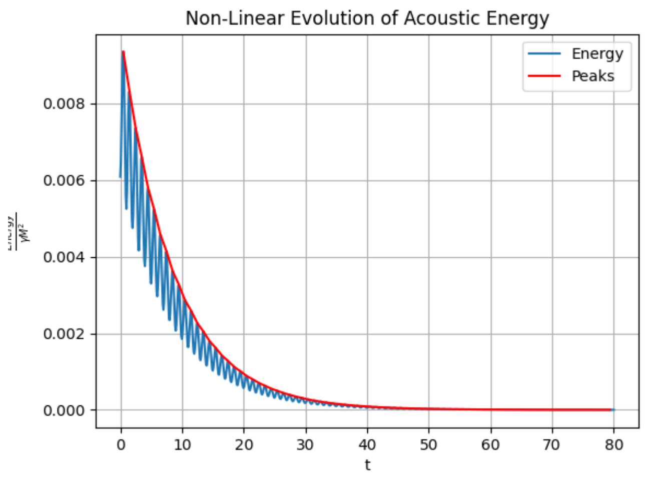


Figure 3: Evolution of non-dimensional acoustic velocity with time

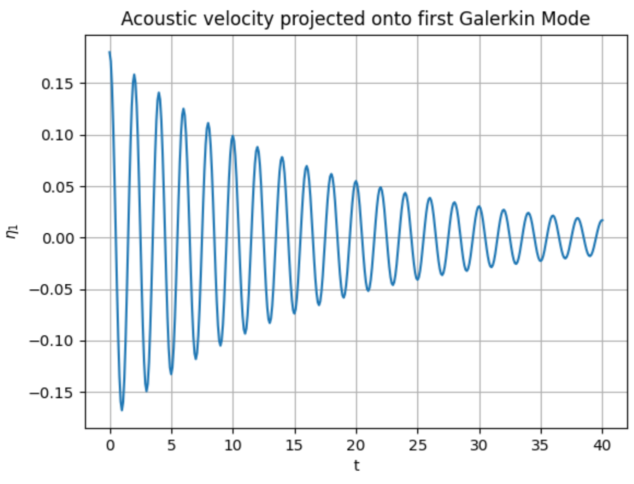
2. Case with three modes, K = 0.1, $x_f = 0.29$, $\tau = 0.5$ s, $\eta_1(0) = 0.18$, $c_1 = 0.1$, $c_2 = 0.06$, $\eta_j(0) = 0$ for j $\neq$ 1 and $\dot{\eta}_j(0) = 0$ for j = 1,2,...,N.



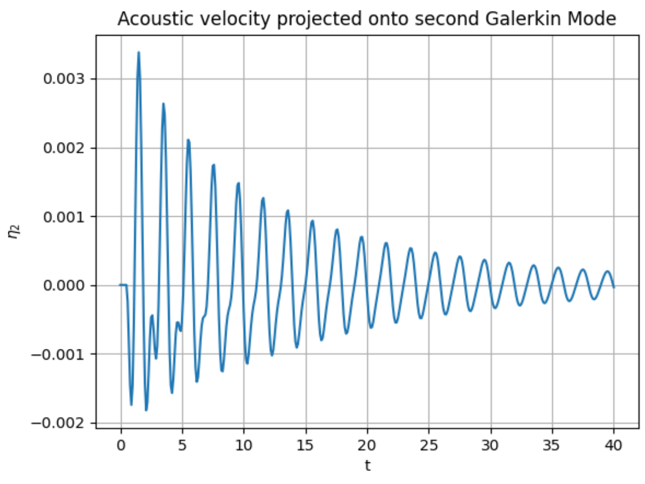
(a)



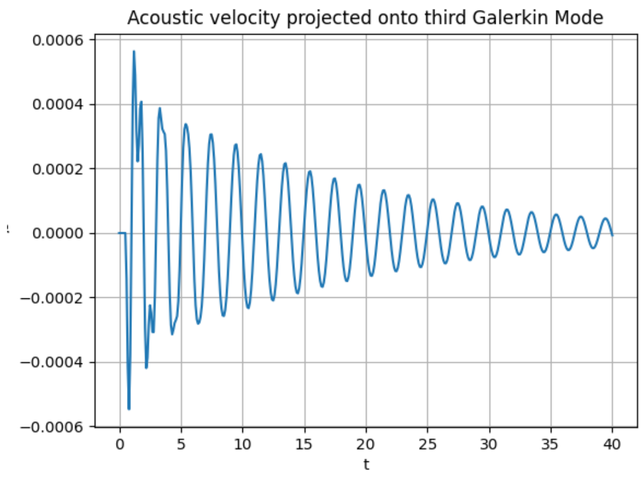
(b)



(c)



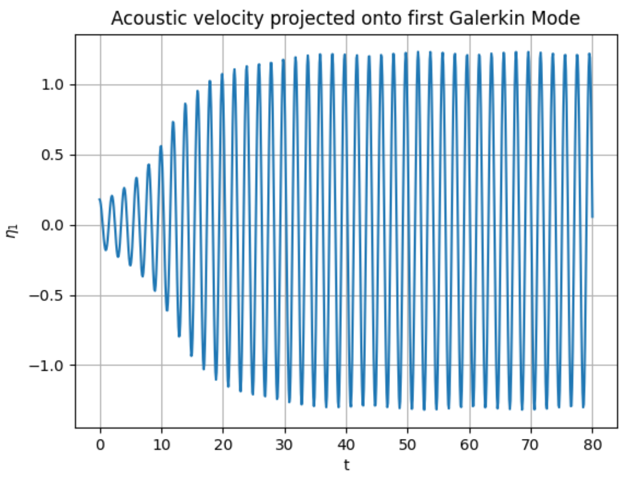
(d)



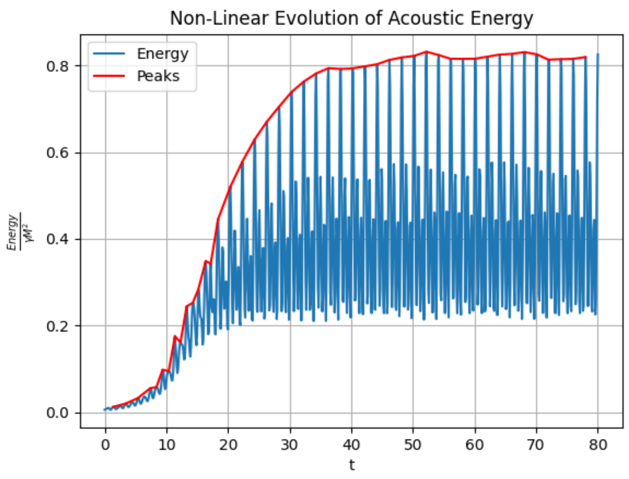
(e)

Figure 4: (a) Evolution of non-dimensional acoustic velocity, (b) Non-linear evolution of acoustic energy, (c) Evolution of acoustic velocity projected onto the first Galerkin mode, (d) Evolution of acoustic velocity projected onto the second Galerkin mode, (e) Evolution of acoustic velocity projected onto the third Galerkin mode

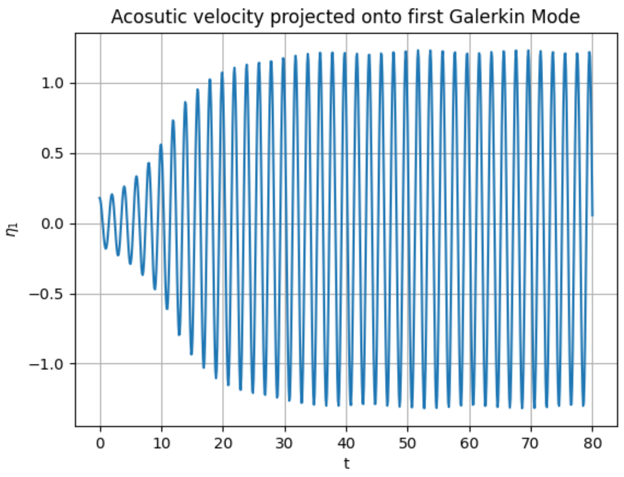
3. Case with three modes, K = 1, $x_f = 0.29$, $\tau = 0.5$ s, $\eta_1(0) = 0.18$, $c_1 = 0.1$, $c_2 = 0.06$, $\eta_j(0) = 0$ for j $\neq$ 1 and $\dot{\eta}_j(0) = 0$ for j = 1,2,...,N.



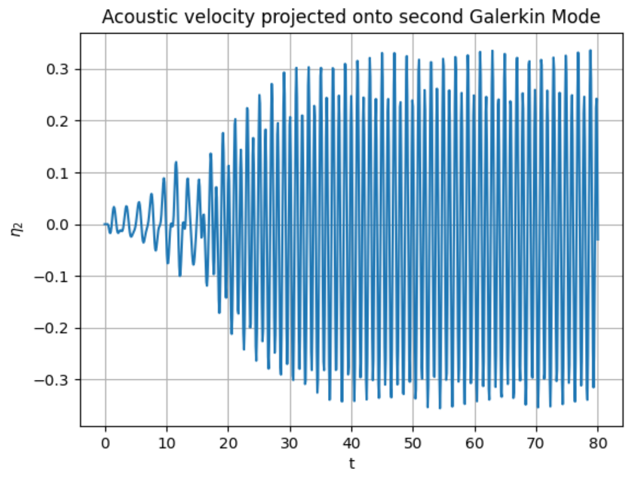
(a)



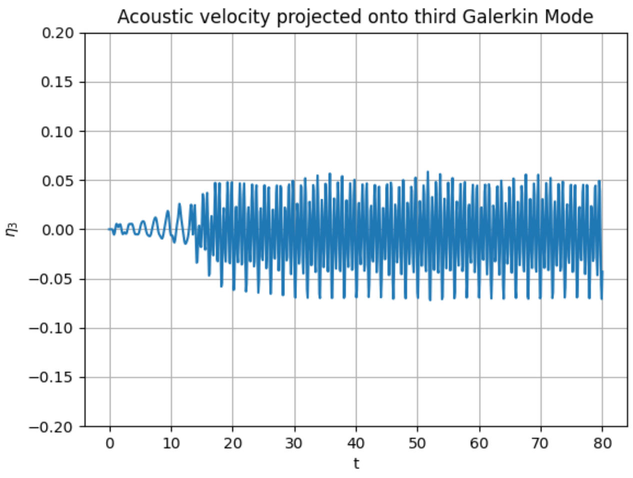
(b)



(c)



(d)



(e)

Figure 5: (a) Evolution of non-dimensional acoustic velocity, (b) Non-linear evolution of acoustic energy, (c) Evolution of acoustic velocity projected onto the first Galerkin mode, (d) Evolution of acoustic velocity projected onto the second Galerkin mode, (e) Evolution of acoustic velocity projected onto the third Galerkin mode

4. Bifurcation plot

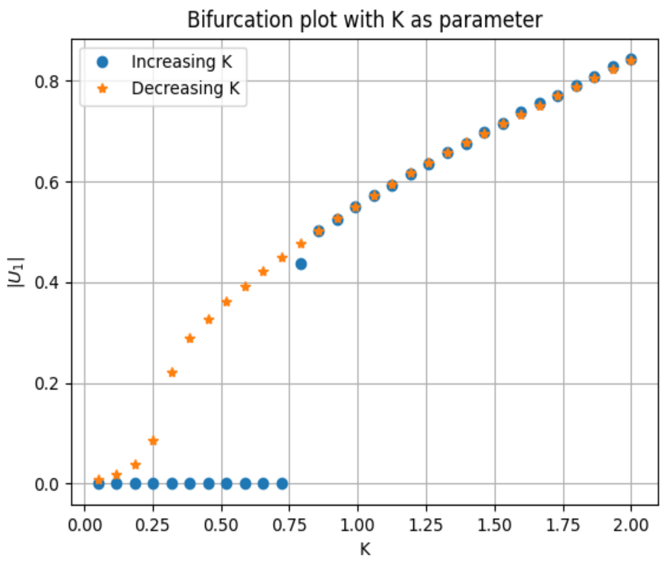


Figure 6: Bifurcation plot for variation of non-dimensional heater power K

5. 3D bifurcation plot

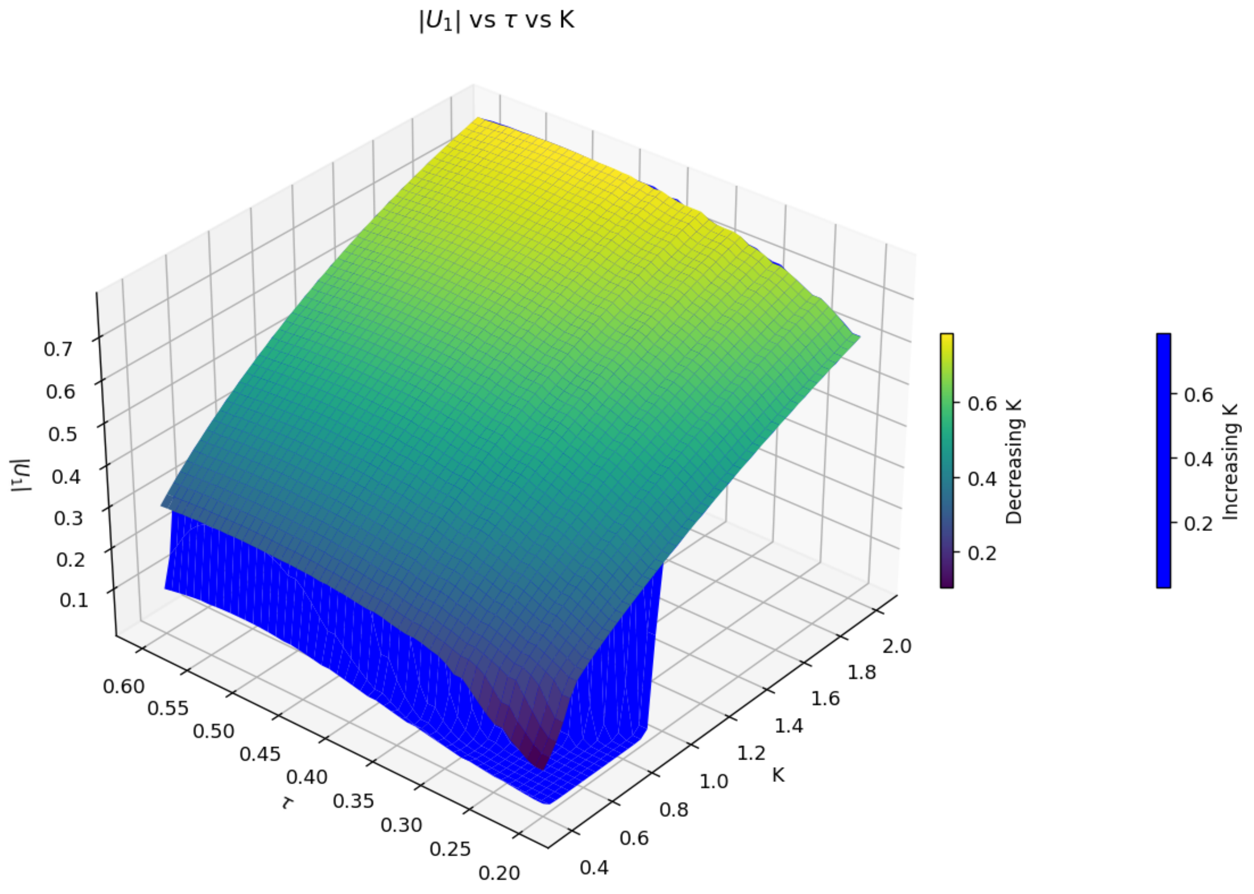


Figure 7: 3D bifurcation plot of non-dimensional heater power K for varying values of time lag

4 Conclusions

- In figure 3, we see that u' was not decaying for a given set of initial conditions for K = 0.01. This implies without damping, the fluctuations in acoustic velocity will not die down for a significantly low value of heater power.
- In figure 4, we see that adding a damping term, lead to the decay of u' for K = 0.1. This implies that with damping, the fluctuations die down for an even higher value of heater power. The energy of the system can be seen as decaying, by observing the peaks of the energy plot.
- In figure 5, we see that for a higher value of heater power K = 1, we the velocity fluctuations started to grow. The energy of the system also increased as can be seen by observing the peaks of the energy plot. Another feature is that, the magnitude of η_3 is an order smaller than other η_j which can be interpreted as the latter modes have lesser contribution towards the velocity fluctuations of the system.
- In figure 6, we are trying to observe fold and Hopf bifurcation by taking the RMS value of the velocities as the measure and the Heater power, K as the parameter. The bifurcation can be observed to a certain extent.
  We can attempt to capture the stable, unstable and bi-stable regions by observing the bifurcation plot. A vague approximation is given below by marking the regions of figure 6 after zooming in a bit.

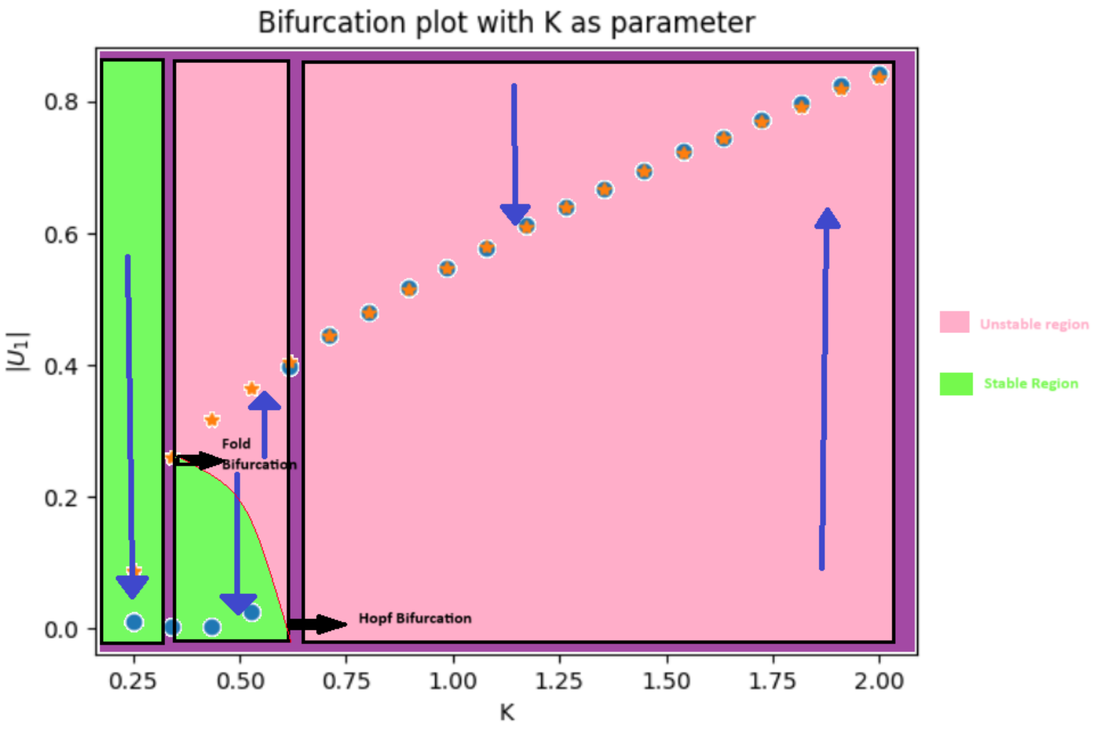
  

  Figure 8: Observing the bifurcation plot

  The above image is an attempt to replicate the images available in literature marking the stability regions and was made by marking regions of figure 6 using a simple editing software.

  We see that for $x_f = 0.3$ and $\tau = 0.5s$, increasing K from 0.25 to 2 result in a bi-stability region for around $0.35 < K < 0.61$. For lesser values of K, we have a stable region and for higher values of K, we have an unstable region.
- In figure 7, we plot a 3D variation of the RMS velocity with changing parameters K and τ. The plot is not exact and meshing is not very fine due to lack of computational power and lack of proper initial conditions. However, we can see hints of the Hopf and Fold bifurcations in the plot.

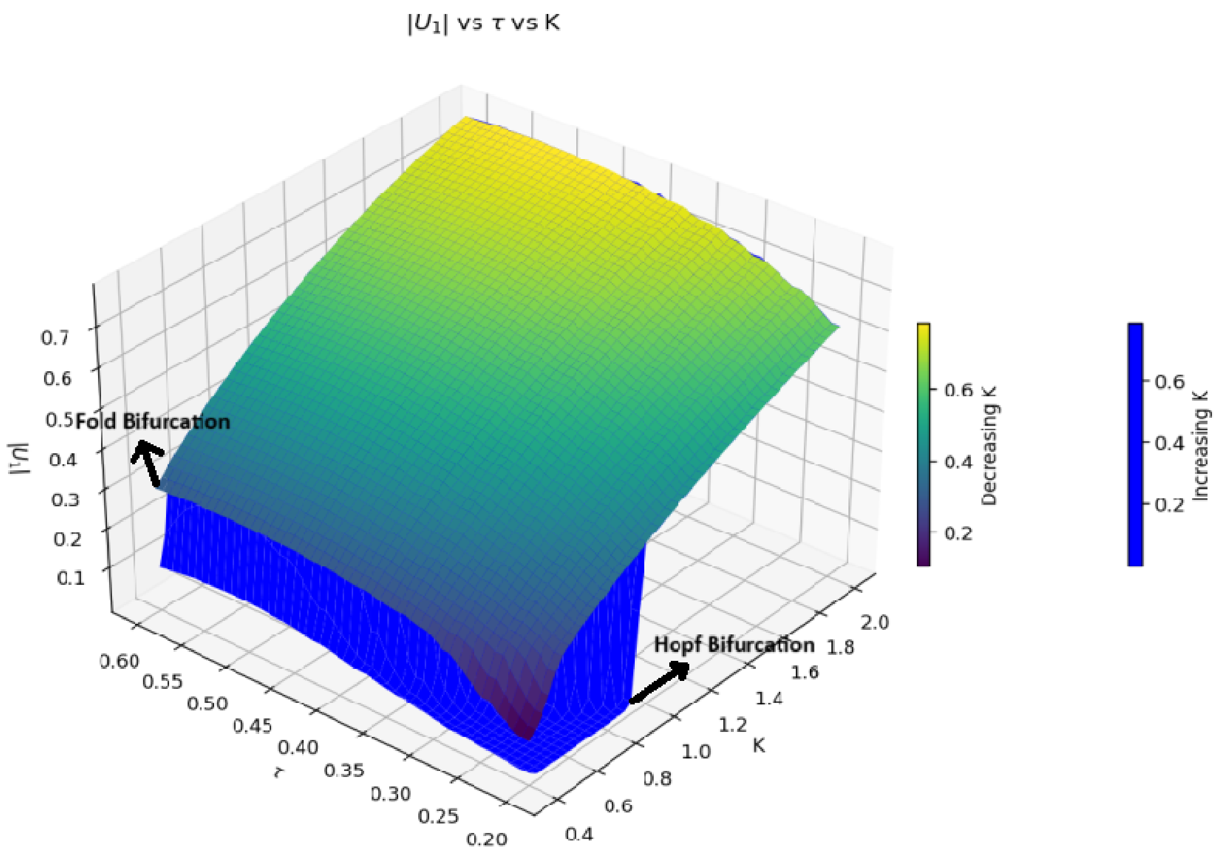


Figure 9: Observing the 3D bifurcation plot

The occurrence of Hopf and Fold bifurcations can be roughly seen here. The 3D plot is an attempt to replicate the figure 6b of [3]

5 Explanation of code

A brief explanation of the logic of the code provided in Appendix [A] is given here.

- The code has 4 functions: f1(), f2(), rk-4-1() and rk-4-2(). f1() and f2() are used to get the derivative functions needed for RK-4 algorithm for $t < \tau$ and $t > \tau$ respectively. rk-4-1() is used to find the values of η_j and $\dot{\eta}_j$ for $t < \tau$.
- Inside rk-4-2(), we call the rk-4-1() to find, η_j, $\dot{\eta}_j$ and u' values for $t < \tau$. The approach is to first march in time and then march in modes. The logic behind this is, as we march in time, the rk-4-2() will compute the values of η_j and $\dot{\eta}_j$ for a particular time and these can be used to find the value of u' at that time which is needed for a later iteration as η_j and $\dot{\eta}_j$ for a later iterating is dependent on the past value of u' through $u'_f(t-\tau)$.
- We start off by using the initial conditions for $\eta_j(0)$, $\dot{\eta}_j$ and other parameters as given in [3] and [1] and vary them according to the plot we want to replicate.
- For the bifurcation plot, we are increasing the value of the heater power, K, in steps and computing the RMS value of u' corresponding to each step. The important point is that, we are continuously increasing the value of K and so, the time is not being set back to 0 after each K. So, the $\eta_j(T)$ and $\dot{\eta}_j(T)$ computed for each K goes into the next iteration of K as the initial value. This keeps on happening even for decreasing K values.
- For the 3D plot, the only difference is adding another loop for varying τ values. The plotting part is taken care by standard functions available in Python.

6 Summary

A brief summary of what is done in this report is mentioned here.

- A brief introduction to thermoacoustic instabilities is given. The possibility of serious damage to the Aerospace engine or its components due to the growing acoustic oscillations is mentioned.
- The mode of the Rijke tube used to perform the numerical analysis is mentioned. It is a standard Rijke tube used by [3] and [1].
- The main objective of the study are to capture the evolution of acoustic velocity with time for damped and undamped systems and how they differ and to study the occurrence of hysteresis through a bifurcation plot as we vary the Heater power. The evolution of the energy of the system is also studied and we see that higher heater power leads to higher energy system and vice-versa.
- The linearized from of the governing equations, i.e, momentum and energy equations are mentioned and non-dimensionalised.
- The heat release rate, $\dot{Q}'$ is modelled using King's Law and Heckl's correlation. We end up with a partial differential equation (PDE).
- Galerkin Technique is used to express the pressure and velocity fluctuations as a superposition of expansion functions in the defined basis. This would enable converting the PDE to a second order ODE.
- We decouple the second order ODE into two first order ODE and use the Runge-Kutta fourth order (RK-4) method to numerically solve them.
- In the later sections, we show the results and interpretations from the solutions obtained.

References

[1] Kosuhik Balasubramanian and R. I. Sujith. "Thermoacoustic instability in a Rijke tube: Non-normality and nonlinearity". In: *Physics of Fluids* 2.4 (2008). Submitted: 18 June 2007 • Accepted: 07 January 2008 • Published Online: 10 April 2008, pp. 325–356.

[2] Beita et al. "Thermoacoustic Instability Considerations for High Hydrogen Combustion in Lean Premixed Gas Turbine Combustors: A Review". In: *Hydrogen* (2021). URL: `https://www.mdpi.com/2673-4141/2/1/3`.

[3] Priya Subramanian and Pankaj Wahi Sathesh Mariappan R. I. Sujith. "Bifurcation analysis of thermoacoustic instability in a horizontal Rijke tube". In: *nternational journal of spray and combustion dynamics* 2.4 (2010). Received on 5th April 2010; Revised submission on 13th August 2010, Accepted on 4th November 2010, pp. 325–356.

[4] Xiaochuan Yang, Ali Turan, and Shenghui Lei. "Thermoacoustic Instability in a Rijke Tube with a Distributed Heat Source". In: *Journal Of Thermodynamics* (2015). URL: `https://www.hindawi.com/journals/jther/2015/949384/`.

Appendix

A Code Used

```python
#Importing necessary modules
import math
import numpy as np
import matplotlib.pyplot as plt
from scipy.signal import find_peaks
from scipy.signal import argrelextrema
from mpl_toolkits.mplot3d import Axes3D
from matplotlib.lines import Line2D
from matplotlib.cm import ScalarMappable
from matplotlib.gridspec import GridSpec
from matplotlib.colors import ListedColormap

#Function to return eta and eta_dot to use in rk4 for t < t_d
def f1(y, j, c1, c2, K_i, t_d, M,x_f):
    gamma = 1.4

    a, b = y
    k_j = j*math.pi
    w_j = k_j
    #z_j = 0   # for no damping case
    z_j = (c1*w_j/math.pi + c2*math.sqrt(math.pi/w_j))/(2*math.pi)     #
   damping

    db_dt = -2*z_j*w_j*b - (k_j**2)*a

    return [b, db_dt]

#Function to return eta and eta_dot to use in rk4 for t > t_d
def f2(y, j, c1, c2, K_i, t_d, M, x_f, u_f):
    gamma = 1.4

    a, b = y
    k_j = j*math.pi
    w_j = k_j
    #z_j = 0  # for no damping case
    z_j = (c1*w_j/math.pi + c2*math.sqrt(math.pi/w_j))/(2*math.pi) # damping

    db_dt = -2*z_j*w_j*b - (k_j**2)*a - j*math.pi*K_i*(math.sqrt(abs(1/3 +
   u_f)) - math.sqrt(1/3))*math.sin(j*math.pi*x_f)

    return [b, db_dt]

# Function to find eta and eta for t < t_d
def rk_4_1(f1, y0, h, j, c1, c2, K_i, t_d, M, x_f):
    a, b = [y0[0]], [y0[1]]

    for i in range(int(t_d/h)):
        a_i, b_i  = a[-1], b[-1]
        k1 =  h*np.array(f1([a_i, b_i], j, c1, c2, K_i, t_d, M, x_f))
        k2 =  h*np.array(f1([a_i + (k1[0] / 2), b_i + (k1[1] / 2)], j, c1,
   c2, K_i, t_d, M, x_f))
        k3 =  h*np.array(f1([a_i + (k2[0] / 2), b_i + (k2[1] / 2)], j, c1,
   c2, K_i, t_d, M, x_f))
        k4 =  h*np.array(f1([a_i + k3[0], b_i + k3[1]], j, c1, c2, K_i, t_d,
    M, x_f))
        a_i_1 = a_i + (1 / 6) * (k1[0] + 2 * k2[0] + 2 * k3[0] + k4[0])
        b_i_1 = b_i + (1 / 6) * (k1[1] + 2 * k2[1] + 2 * k3[1] + k4[1])
        a.append(a_i_1)
        b.append(b_i_1)

    return a, b

# Function to find eta and eta for t > t_d
def rk_4_2(f1, f2, t, y0, h, c1, c2, K_i, t_d, M, x_f, N):

    eta = []
    eta_dot = []

    l = int(t_d/h) + 1

    for i in range(N):
        eta_j, eta_dot_j = rk_4_1(f1, y0[i], h, i+1, c1, c2, K_i, t_d, M, x_f)
        eta.append(eta_j)
        eta_dot.append(eta_dot_j)

    u1 = []
    for k in range(l):
        u_temp = 0
        for i in range(N):
            u_temp += eta[i][k]*math.cos((i+1)*math.pi*x_f)
        u1.append(u_temp)


    for i in range(len(t) - l):    # marching in time
        u_temp = 0
        for k in range(N):    # amrching in modes

            a_1 = eta[k]
            b_1 = eta_dot[k]
            a_i, b_i  = a_1[-1], b_1[-1]

            k1 =  h*np.array(f2([a_i, b_i], k+1, c1, c2, K_i, t_d, M, x_f, u1[i+1]))
            k2 =  h*np.array(f2([a_i + (k1[0] / 2), b_i + (k1[1] / 2)], k+1, c1, c2, K_i, t_d, M, x_f, u1[i+1]))
            k3 =  h*np.array(f2([a_i + (k2[0] / 2), b_i + (k2[1] / 2)], k+1, c1, c2, K_i, t_d, M, x_f, u1[i+1]))
            k4 =  h*np.array(f2([a_i + k3[0], b_i + k3[1]], k+1, c1, c2, K_i, t_d, M, x_f, u1[i+1]))
            a_i_1 = a_i + (1 / 6) * (k1[0] + 2 * k2[0] + 2 * k3[0] + k4[0])
            b_i_1 = b_i + (1 / 6) * (k1[1] + 2 * k2[1] + 2 * k3[1] + k4[1])
            u_temp += a_i_1 * math.cos((k+1)*math.pi*x_f)              # summing up eta_jcos(j*pi*x_f) at a particular time for all modes
            a_1.append(a_i_1)
            b_1.append(b_i_1)
            eta[k] = a_1
            eta_dot[k] = b_1

        u1.append(u_temp)       # appending the summed up u' for each time

    return eta, eta_dot, u1

#Initial values found from references. Change K, t_d, T and N and x_f according to requirements
gamma = 1.4

```

```python
T = 80
h = 0.1
c_0 = 399.6
L_w = 3.6
u_bar = 0.5
M = u_bar/c_0
rho_bar = 1.205
T_w = 1000
C_v = 719
lamda = 0.0328
T_bar = 300
d_w = 0.0005
S = 0.0018
p_bar = 101325
c1 = 0.1
c2 = 0.06

x_f = 0.29  # flame location
t = np.arange(0,T+h,h)
K = 1  # Heater power
#K =  (gamma - 1)*2*L_w*(T_w - T_bar)*math.sqrt(math.pi*lamda*C_v*rho_bar*u_bar*d_w/2)/(S*c_0*p_bar*math.sqrt(3))  # formula for K, gives decayng solution

t_d = 0.5   # time delay
N = 3
n = []

for i in range(1, N+1):
    n.append(i)

y0 = [[0.18, 0]] + [[0, 0]] * (N - 1)   # conditions of eta and eta_dot at t = 0

################## PART OF CODE TO REPRODUCE FIG 4, 5 AND 6 PF REFERENCE 2 ##################
eta, eta_dot, u = rk_4_2(f1, f2, t, y0, h, c1, c2, K, t_d, M, x_f, N)
p = []

for k in range(len(t)):  # 0,1,2,
        p_temp = 0
        for i in range(N):
            p_temp += (gamma*M*eta_dot[i][k]*math.sin(n[i]*math.pi*x_f))/(n[i]*math.pi)

        p.append(p_temp)

E = []

for i in range(len(u)):
    E.append((0.5*(p[i]**2) + 0.5*((gamma*M*u[i])**2))/((gamma*M)**2))

#print(eta[1])
#print(eta[2])

#print(len(u))

# Find the indices of local maxima (peaks)
peaks_idx = argrelextrema(np.array(E), np.greater, order=7)[0]

# Filter peaks to keep only the highest ones
```

```python
min_prominence = 0  # Adjust as needed
prominences = np.array([E[idx] for idx in peaks_idx])
highest_peaks_idx = peaks_idx[prominences > min_prominence]

plt.plot(t, u)
plt.title("Non-dimensional velocity vs time")
plt.xlabel("t")
plt.ylabel("u'")
#plt.ylim([-0.2,0.2])
plt.grid()
plt.show()

plt.plot(t,E, label='Energy')
plt.plot(np.array(t)[highest_peaks_idx], np.array(E)[highest_peaks_idx],
    color='red', markersize=10, label='Peaks')
plt.title("Non-Linear Evolution of Acoustic Energy")
plt.xlabel("t")
plt.ylabel("$\\frac{Energy}{\\gamma M^{2}}$")
plt.legend()
plt.grid()
plt.show()

plt.plot(t,eta[0])
plt.title("Acoustic velocity projected onto first Galerkin Mode")
plt.xlabel("t")
plt.ylabel("$\\eta_1}$")
plt.grid()
plt.show()

plt.plot(t,eta[1])
plt.title("Acoustic velocity projected onto second Galerkin Mode")
plt.xlabel("t")
plt.ylabel("$\\eta_2}$")
plt.grid()
plt.show()

plt.plot(t,eta[2])
plt.title("Acoustic velocity projected onto third Galerkin Mode")
plt.xlabel("t")
plt.ylabel("$\\eta_3}$")
plt.ylim([-0.2,0.2])
plt.grid()
plt.show()

################## PART OF CODE TO REPRODUCE FIG 6A (2D BIFURCATION PLOT) OF
     REFERENCE 1 ##################
T = 100
x_f = 0.3
t_d = 0.5
h = 0.1
t = np.arange(0,T+h,h)
K = np.linspace(0.05, 2, 30)
K_rev = np.linspace(2,0.05, 30)

U1 = []
U2 = []
y0 = [[0.18, 0]] + [[0, 0]] * (N - 1)

# Increasing K
for i in range(len(K)):
    a_2, b_2, u_f = rk_4_2(f1, f2, t, y0, h, c1, c2, K[i], t_d, M, x_f, N)
    second_half_index = len(u_f) // 2
    U1.append(np.sqrt(np.mean(np.square(u_f[second_half_index:])))) # Using
   second half of u' values to calculate Urms

    for x in range(N):
        y0[x] = [a_2[x][len(t)-1] , b_2[x][len(t)-1]]   # Using the eta_j(T)
    and eta_dot_j(T) of ith K as the initial condition for (i+1)th K.

# Decreasing K
for i in range(len(K_rev)):
    a_2_rev, b_2_rev, u_f_rev = rk_4_2(f1, f2, t, y0 , h, c1, c2, K_rev[i],
   t_d, M, x_f, N)
    second_half_index = len(u_f) // 2
    U2.append(np.sqrt(np.mean(np.square(u_f_rev[second_half_index:]))))

    for x in range(N):
        y0[x] = [a_2[x][len(t)-1] , b_2[x][len(t)-1]]


plt.plot(K,U1,'o',label = "Increasing K")
plt.plot(K_rev,U2,'*', label = "Decreasing K")
plt.xlabel('K')
plt.ylabel('$|U_{1}|$')
plt.title('Bifurcation plot with K as parameter')
plt.legend()
plt.grid()
plt.show()

################## PART OF CODE TO REPRODUCE FIG 6B (3D BIFURCATION PLOT) OF
     REFERENCE 1 ###################

# This code is computationally expensive and takes quite some time to run.
   Adjust h value and number of points of t, K and K_rev to get faster plot.

T = 80
x_f = 0.29
N = 1
h = 0.01
t = np.arange(0,T+h,h)
K = np.linspace(0.4, 3, 50)
K_rev = np.linspace(3, 0.4, 50)
t_d = np.linspace(0.2, 0.6, 50)

U1 = np.zeros((len(K), len(t_d)))
U2 = np.zeros((len(K), len(t_d)))

for k in range(len(t_d)):
    y0 = [[0.2, 0]] + [[0, 0]] * (N - 1)
    for i in range(len(K)):

        a_2, b_2, u_f = rk_4_2(f1, f2, t, y0, h, c1, c2, K[i], t_d[k], M,
   x_f, N)

        second_half_index = len(u_f) // 2
        U1[i, k] = np.sqrt(np.mean(np.array(u_f[second_half_index:]) ** 2))

        for x in range(N):
            y0[x] = [a_2[x][len(t)-1] , b_2[x][len(t)-1]]

    for i in range(len(K)):
        a_2_rev, b_2_rev, u_f_rev = rk_4_2(f1, f2, t, y0 , h, c1, c2, K_rev[
```

```python
   i], t_d[k], M, x_f, N)
        second_half_index = len(u_f) // 2
        U2[i,k] = (np.sqrt(np.mean(np.square(u_f_rev[second_half_index:]))))

        for x in range(N):
            y0[x] = [a_2[x][len(t)-1] , b_2[x][len(t)-1]]

# Creating mesh
K_mesh, t_d_mesh = np.meshgrid(K, t_d)
K_mesh_rev, t_d_mesh_rev = np.meshgrid(K_rev, t_d)

# Plotting
fig = plt.figure(figsize=(16, 10))
cmap_single_blue = ListedColormap(['blue'])
# 3D plot
ax1 = fig.add_subplot(111, projection='3d')
surf1 = ax1.plot_surface(K_mesh, t_d_mesh, U1.T, cmap = cmap_single_blue)
surf2 = ax1.plot_surface(K_mesh_rev, t_d_mesh_rev, U2.T, cmap='viridis')
ax1.set_title('$|U_{1}|$ vs $\\tau$ vs K')
ax1.set_xlabel('K')
ax1.set_ylabel('$\\tau$')
ax1.set_zlabel('$|U_{1}|$')

# Colorbars
cb1 = plt.colorbar(surf1, shrink=0.3, aspect= 20, pad=0.0)
cb1.set_label('Increasing K')
cb2 = plt.colorbar(surf2, shrink=0.3, aspect=20, pad=0.001)
cb2.set_label('Decreasing K')

plt.show()
```

Listing 1: Python code used