Model-Based Output-Only Identification of Coupled Thermoacoustic Modes

M.Sc. Thesis
M. K. Pau
15th September 2017

Supervisors: Prof. Dr. N. Noiray and Dr. E. Boujo
Department of Mechanical and Process Engineering, ETH Zürich
Declaration of originality

The signed declaration of originality is a component of every semester paper, Bachelor’s thesis, Master’s thesis and any other degree paper undertaken during the course of studies, including the respective electronic versions.

Lecturers may also require a declaration of originality for other written papers compiled for their courses.

I hereby confirm that I am the sole author of the written work here enclosed and that I have compiled it in my own words. Parts excepted are corrections of form and content by the supervisor.

Title of work (in block letters):

MODEL-BASED OUTPUT-ONLY IDENTIFICATION OF COUPLED THERMOACOUSTIC MODES

Authored by (in block letters):
For papers written by groups the names of all authors are required.

Name(s):
PAUL

First name(s):
MICHAEL KRISTIAN

With my signature I confirm that
- I have committed none of the forms of plagiarism described in the ‘Citation etiquette’ information sheet.
- I have documented all methods, data and processes truthfully.
- I have not manipulated any data.
- I have mentioned all persons who were significant facilitators of the work.

I am aware that the work may be screened electronically for plagiarism.

Place, date
15 / 09 / 2017 ZÜRICH

Signature(s)

For papers written by groups the names of all authors are required. Their signatures collectively guarantee the entire content of the written paper.
Abstract

A model-based output-only method for identifying from time-series data the parameters that determine the dynamics of a stochastically forced thermoacoustic oscillator has previously been implemented in the case of a single standing thermoacoustic mode. This thesis presents an extension of the method in question to two coupled standing modes at the same frequency.

An appropriate thermoacoustic model, as used in the previous works, is presented wherein the dynamics and probability distributions of the coupled thermoacoustic oscillator are described by a Langevin equation and its corresponding Fokker-Planck equation. The system identification method consists of fitting the analytical drift and diffusion coefficients to values that are estimated numerically by extrapolating the finite-time Kramers-Moyal coefficients to the the short-time limit. This method is shown to be functional for the case of coupled standing modes but areas of improvement remain, in particular concerning computational efficiency.

An alternative method of system identification based on iterative finite element solutions of the adjoint Fokker-Planck equation is shown to be unsuitable for coupled standing modes without significant further optimisation in the finite element step.

Finally, an exploratory investigation into the system identification of coupled rotating modes at the same frequency is conducted. It is found that the transition probabilities in this case display unusual behaviour that is unknown at this stage to be physically correct or due to error.

Acknowledgements: Thanks to Prof. Dr. Noiray and Dr. Boujo for supervising the project and the whole team at CAPS for hosting me during my exchange from KTH.
## Contents

1 Introduction 1

2 Theory 3
   2.1 Model 3
   2.2 Solution 10
   2.3 Statistics 11

3 Coupled Standing Modes 15
   3.1 Harmonic Oscillator Equations 15
   3.2 Langevin Equations 17
   3.3 Drift and Diffusion Coefficients 23
   3.4 Change of Variables 23
   3.5 Simulation 24

4 System Identification 27
   4.1 Method 27
   4.2 Results 29

5 Further Investigations 31
   5.1 AFPE-Based System Identification 31
   5.2 Coupled Rotating Modes 33

6 Conclusions 37

A Figures 39

B MATLAB Code 57
   B.1 Standing Modes: System Identification 57
   B.2 Standing Modes: AFPE Solver 69
   B.3 Rotating Modes: System Identification 71
   B.4 Rotating Modes: AFPE Solver 78

References 81
1

Introduction

A classic demonstration of heat-driven sound is the Rijke tube, which was explained by Rayleigh in his influential textbook [1]. By considering the phase relationship between acoustic pressure and heat release he formulated the Rayleigh criterion; if the product of the acoustic pressure and heat release, integrated over an oscillation cycle, is positive then in the absence of damping the oscillation will be maintained. That is, if the heat release is more in phase than out of phase with the acoustic pressure then a positive feedback loop can develop.

While the Rijke tube is primarily known as a lecture demonstration, such thermoacoustic coupling can occur in other settings. In gas turbine combustors it can lead to high amplitude self-sustained dynamic pressure oscillations. In most cases these produce marked pulsations at specific frequencies, in contrast to broadband combustion roar, which makes them fairly easy to identify by spectral analysis. They are typically detrimental because they can cause structural vibrations in the combustor which in turn can result in damage or failure. It is therefore useful to be able to characterise these oscillations. Ideally a system identification method would be able to inform in real time whether a destructive oscillation is developing in order to allow a change in operating conditions to control it.

Annular combustion chambers as are in general use [2] often exhibit thermoacoustic oscillations corresponding to azimuthal modes and these have been the subject of much study in recent years [3, 4, 5, 6, 7, 8]. These studies have found that both dominant standing and rotating modes can be found, depending on the gas turbine operating conditions and burner arrangement. Typically one finds a pair of azimuthal standing modes at the same frequency, with transitions occurring between a dominant standing mode and the combined rotating mode. This is behaviour characteristic of a weakly damped oscillator with two time scales, the oscillation frequency and relaxation time, which corresponds to the dynamics of the pressure amplitude envelope [9].

The observed modal dynamics of such combustors cannot be fully described by deterministic models alone as there is inherent stochastic forcing from turbulence and flow instabilities. Due to the ubiquity of stochastic processes in a wide range of fields, many methods have been developed for dealing with stochastic processes and in particular methods based on the Fokker-Planck equation have found applications from medicine [10] to eco-
nomics [11, 12].

A model-based output-only method for identifying the parameters that determine the dynamics of a stochastically forced thermoacoustic oscillator has previously been implemented for the case of a single mode by Noiray and Boujo [13, 14]. We will present the basis of the model used and extend the system identification method to coupled standing modes. The model allows the system to be described by a Langevin equation and its corresponding Fokker-Planck equation. The system identification method is then based on fitting the model analytical drift and diffusion coefficients to values that are estimated numerically from time-series data by extrapolating the finite-time Kramers-Moyal coefficients.

An alternative method based on iterative solutions to the adjoint Fokker-Planck equation, that has also been implemented by Noiray and Boujo [14] for the single-mode case, is investigated for the coupled-mode case. Additionally the case of coupled rotating modes is briefly explored.
2

Theory

We begin with the necessary theoretical background; first providing a complete derivation of the thermoacoustic model to be used; second discussing the form of solution to be used; third presenting the statistical basis of the system identification methods.

2.1 Model

To date there exists no complete theory for the generation of sound in aeroacoustic flows. However, there is a powerful tool, Lighthill’s aeroacoustic analogy [15, 16], that can be used for practical analyses. This analogy manipulates the equations of fluid dynamics into a non-homogeneous differential equation whose associated homogeneous equation is the classical wave equation and whose remaining parts are interpreted as source terms. The solution is split into two parts: the acoustic field which satisfies the homogeneous equation, and the source field which satisfies the non-homogeneous equation. The primary limitation of Lighthill’s analogy is the assumption that the source field is independent to the acoustic field, which means that the source field must be prescribed based on measurements or another theory.

Reaching our thermoacoustic model consists of three stages; first derivation of the Lighthill equation in terms of pressure, following [17]; second an expansion of the source terms to account for combustion, following [18]; and third a prescribed model for the combustion source [9, 13, 19].

We start by considering the fundamental equations of fluid mechanics, conservation of mass and momentum.

Conservation of Mass

Conservation of mass in an arbitrary material control volume $V$ requires that the rate of change of mass within $V$ is equal to the mass flow out of $V$ plus the mass produced inside $V$. In integral form this can be stated as

$$\iiint_V \frac{\partial \rho}{\partial t} dV = -\iint_S \rho \mathbf{u} \cdot \mathbf{n} dS + \iiint_V m dV, \quad (2.1)$$

where $\rho$ is the fluid density, $\mathbf{u}$ is the fluid velocity field and $m$ is the rate of mass production per unit volume.
Given a vector \( \mathbf{A} \), Gauss’s theorem states that

\[
\iiint_S \mathbf{A} \cdot \mathbf{n} dS = \iiint_V \nabla_i \mathbf{A}_i dV,
\]
(2.2)

which when applied to Eq. 2.1 gives

\[
\iiint_V \left( \frac{\partial \mathbf{p}}{\partial t} + \nabla_i (\mathbf{p} \mathbf{u}_i) - \mathbf{m} \right) dV = 0.
\]
(2.3)

As the control volume is arbitrary and the integrand is continuous the
continuity equation in differential form can be read directly as

\[
\frac{\partial \mathbf{p}}{\partial t} + \nabla_i (\mathbf{p} \mathbf{u}_i) = \mathbf{m}.
\]
(2.4)

Conservation of Momentum

Newton’s second law applied to the same fixed control volume \( V \) states that
the rate of change of momentum within \( V \) plus the flow of momentum out of
\( V \) is equal to the sum of external and internal forces, where \( \mathbf{f}_V \) are the external
or volume forces and \( \mathbf{f}_S \) are the internal or surface forces. In integral form
this reads

\[
\iiint_V \frac{\partial}{\partial t} (\mathbf{p} \mathbf{u}_i) dV + \iint_S \rho \mathbf{u}_i (\mathbf{u}_j \mathbf{n}_j) dS = \iiint_V \mathbf{f}_V dV + \iint_S \mathbf{f}_S dS.
\]
(2.5)

Assuming that the fluid behaves as a Newtonian fluid it adheres to the
constitutive relation

\[
\mathbf{f}_S = -p \mathbf{n} + \tau_{ij} \mathbf{n}_j,
\]
(2.6)

where \( p \) is the thermodynamic pressure and

\[
\tau_{ij} = \eta (\nabla_j \mathbf{u}_i + \nabla_i \mathbf{u}_j) - \frac{2}{3} \eta (\nabla_k \mathbf{u}_k) \delta_{ij}
\]
(2.7)

is the viscous stress tensor, \( \eta \) is the dynamic viscosity and \( \delta_{ij} \) is Kronecker’s
delta.

Applying Gauss’s theorem to Eq. 2.1 as was done to Eq. 2.5 results in the
differential expression of momentum conservation,

\[
\frac{\partial}{\partial t} (\rho \mathbf{u}_i) + \nabla_j (\rho \mathbf{u}_i \mathbf{u}_j) = -\nabla_i p + \nabla_j \tau_{ij} + \mathbf{f}_V.
\]
(2.8)
Lighthill’s Analogy

Differentiating Eq. 2.4 with respect to time and subtracting the divergence of Eq. 2.8 gives

\[
\frac{\partial^2 \rho}{\partial t^2} - \nabla_i \nabla_i \rho = \frac{\partial m}{\partial t} - \nabla_i f_{Vi} + \nabla_i \nabla_j (\rho u_i u_j - \tau_{ij}) .
\]  

(2.9)

The fluid field is split into two parts, the acoustic field satisfying the homogeneous wave equation and the source field satisfying the inhomogeneous equation. The acoustic field encompasses the source field and is characterised in the absence of a mean flow by the mean values \( p_0, \rho_0 \) and the sound speed \( c_0 = \sqrt{\gamma p_0 / \rho_0} \), where \( \gamma \) is the specific heat ratio. The total pressure and density can be expressed as a sum of mean and fluctuating parts, \( p = p_0 + p' \) and \( \rho = \rho_0 + \rho' \).

Addition of \( 1/c_0^2 \partial^2 p' / \partial t^2 \) to both sides of Eq. 2.9 gives Lighthill’s equation in terms of pressure,

\[
\frac{1}{c_0^2} \frac{\partial^2 p'}{\partial t^2} - \nabla_i \nabla_i p' = \frac{\partial m}{\partial t} - \nabla_i f_{Vi} + \nabla_i \nabla_j (\rho u_i u_j - \tau_{ij}) - \frac{\partial^2 \rho_e}{\partial t^2} .
\]  

(2.10)

where

\[
\rho_e = \left( \frac{c_0^2 \rho' - p'}{c_0^2} \right)
\]  

(2.11)

is the excess density which vanishes in the far field but is nonzero in the source field. The excess density term in Eq. 2.10 represents non-adiabatic changes of state such as during heat release from combustion.

To simplify the final equation for the thermoacoustic model we assume that the excess density term is dominant, in which case Eq. 2.10 reduces to

\[
\frac{1}{c_0^2} \frac{\partial^2 p'}{\partial t^2} - \nabla_i \nabla_i p' = -\frac{\partial^2 \rho_e}{\partial t^2} .
\]  

(2.12)

Combustion Source

To expand the excess density source we must consider the thermodynamics of the source region. The first law of thermodynamics states that the change in internal energy equals the work done, pressure and viscous, minus the heat lost

\[
\rho \frac{D_e}{Dt} = -\rho \nabla_i u_i + (\nabla_j u_i - \tau_{ij}) - \nabla_i q_i ,
\]  

(2.13)

where \( e \) is the internal energy per unit mass and \( q \) is the heat flux.

The fundamental thermodynamic relation in terms of energy states that
\[ dc = T ds + \frac{p}{\rho^2} d\rho + \sum_n \frac{\mu_n}{W_n} dY_n, \quad (2.14) \]

where \( T \) is absolute temperature, \( s \) is entropy per unit mass and we have substituted \( \rho \) for volume per unit mass. \( \mu_n \) is the chemical potential of species \( n \) with molecular weight \( W_n \) and mass fraction \( Y_n \).

Substitution of Eq. 2.14 into Eq. 2.13 gives

\[ \rho T \frac{DS}{Dt} + \frac{p}{\rho} \frac{D\rho}{Dt} + \sum_n \frac{\mu_n}{W_n} \rho \frac{DY_n}{Dt} = -p \nabla_i u_i + (\nabla_j u_i) \tau_{ij} - \nabla_i q_i. \quad (2.15) \]

In Eq. 2.12 we assumed that the source term \( \partial m/\partial t \) could be neglected and we extend that assumption by neglecting mass injection altogether, that is \( m = 0 \). Using Eq. 2.4, the second term on the left and first term on the right of Eq. 2.15 can then be eliminated,

\[ \frac{p}{\rho} \frac{D\rho}{Dt} + p \nabla_i u_i = \frac{p}{\rho} \left( \frac{\partial p}{\partial \rho} \bigg|_{\rho,Y_n} \nabla_i \rho + \rho \nabla_i u_i \right) \\
= \frac{p}{\rho} \left( \frac{\partial p}{\partial \rho} + \nabla_i (\rho u_i) \right) = 0. \quad (2.16) \]

This simplifies Eq. 2.15 to

\[ \frac{DS}{Dt} = -\frac{1}{\rho T} \sum_n \frac{\mu_n}{W_n} \rho \frac{DY_n}{Dt} + \frac{1}{\rho T} (\nabla_j u_i) \tau_{ij} - \frac{1}{\rho T} \nabla_i q_i. \quad (2.17) \]

Using the chain rule and the general definition \( c = \sqrt{\left(\frac{\partial p}{\partial \rho}\right)_s} \) provides another relationship,

\[ \frac{DP}{Dt} = \frac{\partial p}{\partial \rho} \bigg|_{s,Y_n} \frac{D\rho}{Dt} + \frac{\partial p}{\partial s} \bigg|_{\rho,Y_n} \frac{DS}{Dt} + \sum_n \frac{\partial p}{\partial Y_n} \bigg|_{s,\rho,Y_n} \frac{DY_n}{Dt} \quad (2.18) \]

which when rearranged for the total derivative of density is

\[ \frac{DP}{Dt} = \frac{1}{c^2} \frac{D\rho}{Dt} + \frac{1}{c^2} \frac{\partial p}{\partial s} \bigg|_{\rho,Y_n} \frac{DS}{Dt} - \sum_n \frac{\partial p}{\partial Y_n} \bigg|_{s,\rho,Y_n} \frac{DY_n}{Dt}. \quad (2.19) \]

Substituting 2.17 for \( DS/Dt \) into 2.19 gives

\[ \frac{DP}{Dt} = \frac{1}{c^2} \frac{D\rho}{Dt} + \frac{1}{c^2 T} \frac{\partial p}{\partial s} \bigg|_{\rho,Y_n} \left( \nabla_i q_i - (\nabla_j u_i) \tau_{ij} \right) \\
+ \sum_n \frac{1}{c^2} \left( \frac{1}{T} \frac{\mu_n}{W_n} \frac{\partial p}{\partial s} \bigg|_{\rho,Y_n} - \frac{\partial p}{\partial Y_n} \bigg|_{s,\rho,Y_n} \right) \frac{DY_n}{Dt}. \quad (2.20) \]
Eq. 2.14 restated in terms of enthalpy per unit mass, \( h \), is

\[
dh = T \, ds + \frac{1}{\rho} \, dp + \sum_n \frac{\mu_n}{W_n} \, dY_n,
\]

which with repeated use of the chain rule again yields two identities,

\[
\frac{1}{\rho T c^2} \frac{\partial p}{\partial s} \bigg|_{\rho,Y_n} = \frac{\alpha}{c_p}
\]

and

\[
\frac{1}{c^2} \left( \frac{1}{T W_n} \frac{\partial p}{\partial s} \bigg|_{\rho,Y_n} - \frac{\partial p}{\partial Y_n} \bigg|_{s,p,Y_m} \right) = \frac{\rho \alpha}{c_p} \frac{\partial h}{\partial Y_n} \bigg|_{\rho,p,Y_m}
\]

where \( \alpha \) is the volumetric coefficient of thermal expansion and \( c_p \) is the specific heat capacity at constant pressure.

Eq. 2.20 then simplifies to

\[
\frac{D \rho}{D t} = \frac{1}{c^2} \frac{D p}{D t} + \frac{\alpha}{c_p} \left( \sum_n \frac{\partial h}{\partial Y_n} \bigg|_{\rho,p,Y_m} \frac{D Y_n}{D t} + \nabla_i q_i - (\nabla_j u_i) \tau_{ij} \right),
\]

Using the chain rule on enthalpy, we can form the expression

\[
\frac{\partial h}{\partial Y_n} \bigg|_{T,p,Y_m} = \frac{\partial h}{\partial Y_n} \bigg|_{\rho,p,Y_m} + \frac{\partial h}{\partial \rho} \bigg|_{p,Y_n} \frac{\partial \rho}{\partial Y_n} \bigg|_{T,p,Y_m},
\]

from which it follows that

\[
\sum_n \frac{\partial h}{\partial Y_n} \bigg|_{\rho,p,Y_m} \frac{D Y_n}{D t} = \sum_n \frac{\partial h}{\partial Y_n} \bigg|_{T,p,Y_m} \frac{D Y_n}{D t} - \frac{\partial h}{\partial \rho} \bigg|_{p,Y_n} \left( \sum_n \frac{\partial \rho}{\partial Y_n} \bigg|_{T,p,Y_m} \frac{D Y_n}{D t} \right),
\]

where the left hand side appears in Eq. 2.24.

Conservation of species demands that

\[
\rho \frac{D Y_n}{D t} = w_n - \nabla_i J_{ni},
\]

where \( w_n \) is the production rate per unit volume of species \( n \) by reaction and \( J_n \) is the flux of species \( n \) by diffusion.

When diffusion of species is neglected,

\[
\rho \frac{D Y_n}{D t} = w_n,
\]
then the first term on the right hand side of Eq. 2.26 is equal to $-Q$, where $Q$ is the heat release rate per unit volume due to changes in species concentration.

The final term of Eq. 2.26 concerns the change in density due to changes in species concentration. If we assume that the average molecular weight is constant then

$$
\sum_n \frac{\partial \rho}{\partial Y_n} \left. \frac{DY_n}{Dt} \right|_{T,p,Y_m} = 0 \quad (2.29)
$$

and Eq. 2.26 becomes

$$
\sum_n \frac{\partial h}{\partial Y_n} \left. \rho \frac{DY_n}{Dt} \right|_{\rho,p,Y_m} = -Q. \quad (2.30)
$$

Using this in Eq. 2.24 leads to

$$
\frac{D\rho}{Dt} = \frac{1}{c^2} \frac{Dp}{Dt} + \frac{\alpha}{c_p} \left( -Q + \nabla_i q_i - (\nabla_j u_i) \tau_{ij} \right), \quad (2.31)
$$

which shows that the density of a material particle changes due to both pressure variations and thermal expansion.

Returning to the excess density, we can write

$$
\frac{D\rho_e}{Dt} = \frac{\partial \rho_e}{\partial t} + u_i \nabla_i \rho_e = \frac{\partial \rho_e}{\partial t} + \nabla_i (\rho_e u_i) - \rho_e \nabla_i u_i \quad (2.32)
$$

or

$$
\frac{\partial \rho_e}{\partial t} = \frac{D\rho_e}{Dt} - \frac{\rho_e}{\rho} \frac{D\rho}{Dt} - \nabla_i (\rho_e u_i), \quad (2.33)
$$

where from Eq. 2.4 we have used

$$
\nabla_i u_i = -\frac{1}{\rho} \frac{D\rho}{Dt}. \quad (2.34)
$$

Recalling the definition $\rho_e = \rho' - p'/c_0^2 = \rho - \rho_0 - (p-p_0)/c_0^2$, Eq. 2.33 becomes
\[
\frac{\partial \rho_e}{\partial t} = \left( \frac{D \rho}{Dt} - \frac{D \rho_0}{Dt} - \frac{1}{c_0^2} \frac{D p}{Dt} + \frac{1}{c_0^2} \frac{D \rho_0}{Dt} \right)
- \frac{(\rho - \rho_0 - p/c_0^2 + p_0/c_0^2) D \rho}{\rho} - \nabla_i (\rho_e \mathbf{u}_i)
- \frac{1}{c_0^2} \frac{D p}{Dt} + \frac{\rho_0}{\rho} \frac{D \rho}{Dt} + \frac{p}{\rho c_0^2} \frac{D \rho}{Dt} - \frac{p_0}{\rho c_0^2} \frac{D \rho}{Dt} - \nabla_i (\rho_e \mathbf{u}_i) \quad (2.35)
\]

and substitution of Eq. 2.31 for \(D \rho/Dt\) then leads to

\[
\frac{\partial \rho_e}{\partial t} = \frac{\alpha \rho_0}{c_p \rho} \left( -Q + \nabla_i q_i - (\nabla_j \mathbf{u}_i) \tau_{ij} \right) - \nabla_i (\rho_e \mathbf{u}_i)
- \frac{1}{c_0^2} \left( \left( 1 - \frac{\rho_0 c_0^2}{\rho c^2} \right) \frac{D p}{Dt} - \frac{(p - p_0) D \rho}{\rho} \right). \quad (2.36)
\]

The wave equation with expanded excess density source terms is then

\[
\frac{1}{c_0^2} \frac{\partial^2 p'}{\partial t^2} - \nabla_i \nabla_i p' = -\frac{\partial}{\partial t} \left( \frac{\alpha \rho_0}{c_p \rho} \left( -Q + \nabla_i q_i - (\nabla_j \mathbf{u}_i) \tau_{ij} \right) \right)
+ \frac{\partial}{\partial t} \nabla_i \left( \frac{(\rho - \rho_0)c_0^2 - (p - p_0)}{c_0^2} \mathbf{u}_i \right)
+ \frac{\partial}{\partial t} \frac{1}{c_0^2} \left( \left( 1 - \frac{\rho_0 c_0^2}{\rho c^2} \right) \frac{D p}{Dt} - \frac{(p - p_0) D \rho}{\rho} \right). \quad (2.37)
\]

All non-adiabatic sources are represented in the above equation. The first term represents the monopole sound generated by irreversible flow processes and is the one of interest to us; in combustion, \(Q \neq 0\) and this term dominates.

\[
\frac{1}{c_0^2} \frac{\partial^2 p'}{\partial t^2} - \nabla_i \nabla_i p' = \frac{\partial}{\partial t} \left( \frac{\alpha \rho_0}{c_p \rho} Q \right) \quad (2.38)
\]

In an ideal gas \(\alpha/c_p = (\gamma - 1)/c^2\) and if we assume that the combustion occurs at ambient pressure then \(\rho c^2 = \gamma p_0 = \rho_0 c_0^2\) and

\[
\frac{1}{c_0^2} \frac{\partial^2 p'}{\partial t^2} - \nabla_i \nabla_i p' = \frac{\partial}{\partial t} \left( \frac{\gamma - 1}{c_0^2} Q \right). \quad (2.39)
\]
Or, defining a normalised heat release rate \( Q' = (\gamma - 1)Q \),
\[
\frac{\partial^2 p'}{\partial t^2} - c_0^2 \nabla_i \nabla_i p' = \frac{\partial Q'}{\partial t}.
\] (2.40)

### Additional Considerations

From here on we will drop the ’ notation used so far as it is no longer needed for clarity. We have no indication from Eq. 2.40 of how the combustion source term depends on the pressure field. As was brought up at the beginning of the chapter we must prescribe a model to describe the acoustic-flame coupling, i.e. how \( Q = Q(p) \). In general the coupling has both a general behaviour that can be described deterministically and a random element due to stochastic forcing from turbulence.

\[
Q = Q_{\text{deterministic}} + Q_{\text{stochastic}}
\] (2.41)

It has been shown that the deterministic part can be approximated by a hyperbolic tangent function, which in turn can be approximated by a third order Taylor expansion in the low amplitude range [19]. The additional approximation makes for easier handling.

\[
Q = \beta p - \kappa p^3.
\] (2.42)

We also include a term for acoustic damping, \( \alpha (\partial p/\partial t) \), which although simple has been shown to be adequate [19]. This gives the deterministic equation for the thermoacoustic model as
\[
\frac{\partial^2 p}{\partial t^2} - c_0^2 \nabla_i \nabla_i p = (2\nu - 3\kappa p^2) \frac{\partial p}{\partial t},
\] (2.43)

where we have introduced the growth rate \( \nu = (\beta - \alpha) / 2 \).

Finally, we will only be considering azimuthal modes. In polar coordinates and having neglected any axial or radial dependence the equation is
\[
\frac{\partial^2 p}{\partial t^2} - \omega^2 \frac{\partial^2 p}{\partial \theta^2} = (2\nu - 3\kappa p^2) \frac{\partial p}{\partial t},
\] (2.44)

where \( \omega = c_0^2 / R^2 \) is the fundamental frequency and \( R \) is the mean radius of the annular combustor.

### 2.2 Solution

As we are looking for an acoustic solution it is natural to assume that the solution can be decomposed as a Fourier series where the coefficients are spatial wave-functions,
\[ p(r, \theta, z, t) = \sum_n \Psi_n(r, \theta, z) \cos(\omega_n t + \phi_n). \quad (2.45) \]

Given that \( p \) is an acoustic pressure, the above must be a zero mean solution and there is no zeroth component, \( \Psi_0 \). We assume that the \( \Psi_n \) are separable in spatial coordinates,

\[ \Psi_n = A_n f_n(r, z) \Theta_n(\theta), \quad (2.46) \]

and that there is only azimuthal spatial dependence, \( f_n(r, z) = 1 \), because we are interested in azimuthal modes only.

\[ p(\theta, t) = \sum_n A_n \Theta_n(\theta) \cos(\omega_n t + \phi_n) \quad (2.47) \]

In general, the amplitudes \( A_n \) and phases \( \phi_n \) can be time dependent if they vary slowly with respect to the oscillation timescale and the simplest form for \( \Theta(\theta) \) is \( \cos(n\theta) \) or \( \sin(n\theta) \). This gives a Fourier decomposition of

\[ p(\theta, t) = \sum_n A_n(t) \cos(n\theta) \cos(\omega_n t + \phi_{nA}(t)) \]

\[ + B_n(t) \sin(n\theta) \cos(\omega_n t + \phi_{nB}(t)), \quad (2.48) \]

or

\[ p(\theta, t) = \sum_n \eta_{nA}(t) \cos(n\theta) + \eta_{nB}(t) \sin(n\theta) \quad (2.49) \]

with

\[ \eta_{nA} = A_n(t) \cos(\omega_n t + \phi_{nA}(t)) \]

\[ \eta_{nB} = B_n(t) \cos(\omega_n t + \phi_{nB}(t)). \quad (2.50) \]

### 2.3 Statistics

The dynamics of a continuous Markovian system of \( n \) macroscopic variables, \( \{X_k(t)\} \) \((k = 1, ..., n)\), is governed by a system of Langevin equations

\[ \frac{d}{dt} X_i(t) = h_i(\{X_k(t)\}, t) + \sum_j g_{ij}(\{X_k(t)\}, t) \xi_j(t), \quad (2.51) \]

where the fluctuating Langevin forces \( \xi \) are \( \delta \)-correlated, zero mean noise functions with intensities \( \Gamma \),
\[ \langle \xi_i(t) \rangle = 0 \quad (2.52) \]
\[ \langle \xi_i(t + \tau) \xi_j(t) \rangle = \sqrt{\Gamma_i \Gamma_j} \delta_{ij} \delta(\tau), \]

and \( h \) and \( g \) are the Langevin coefficients [20]. If \( g \) is constant then the noise is referred to as additive, if not it is referred to as multiplicative. We will deal only with additive noise.

The Markovian condition requires that the system at the next time, \( t + \tau \) depends only on the present state of the system at \( t \). That is to say, a Markovian process does not depend on its history. If we define the conditional probability density, \( P \), as

\[ P(\{X_k\}_n, t_n | \{X_k\}_{n-1}, t_{n-1}; \ldots; \{X_k\}_1, t_1) \]

\[ = \frac{W(\{X_k\}_n, t_n; \ldots; \{X_k\}_1, t_1)}{W(\{X_k\}_{n-1}, t_{n-1}; \ldots; \{X_k\}_1, t_1)} \quad (2.53) \]

where \( \{X_k\}_n \) denotes the set of values for \( \{X_k\} \) at a specific time \( t_n \) and

\[ \int W(\{X_k\}_n, t_n; \ldots; \{X_k\}_1, t_1) \prod_k dX_{ki} \quad (2.54) \]

is the probability that the system is in the state \( \{X_k\}_i \ldots \{X_k\}_i + d\{X_k\}_i \) at time \( t_i \), then the Markovian property is

\[ P(\{X_k\}_n, t_n | \{X_k\}_{n-1}, t_{n-1}; \ldots; \{X_k\}_1, t_1) = P(\{X_k\}_n, t_n | \{X_k\}_{n-1}, t_{n-1}) \]

\[ t_n > t_{n-1} > \ldots > t_1, \quad (2.55) \]

where \( P(\{X_k\}_n, t_n | \{X_k\}_{n-1}, t_{n-1}) \) is the transition probability density [20]. We will write this as \( P(\{X_k\}_{t+\tau} | \{X_k\}_t) \).

The Fokker-Planck equation describes the time evolution of the probability densities of a system governed by Eq. 2.51 [20],

\[ \frac{\partial}{\partial t} W(\{X_k(t)\}, t) = \mathcal{L} [W(\{X_k(t)\}, t)] \quad (2.56) \]

where the Fokker-Planck operator \( \mathcal{L} \) is defined as

\[ \mathcal{L} = -\sum_i \frac{\partial}{\partial X_i} D_i^{(1)}(\{X_k(t)\}, t) + \sum_i \sum_j \frac{\partial^2}{\partial X_i \partial X_j} D_{ij}^{(2)}(\{X_k(t)\}, t). \quad (2.57) \]
The coefficients \( D^{(1)} \) and \( D^{(2)} \) are the first and second Kramers-Moyal coefficients, the drift and diffusion coefficients respectively, and are defined as

\[
D^{(1)}_i(\{X_k(t)\}, t) = D_i(\{X_k(t)\}, t) = \lim_{\tau \to 0} \frac{1}{\tau} \langle x_i(t + \tau) - X_i(t) \rangle \bigg|_{X_k(t) = X_k(t)}
\]

\[
= \lim_{\tau \to 0} \frac{1}{\tau} \int_{-\infty}^{\infty} (x_i(t + \tau) - X_i(t)) P(\{x_k\}_{t+\tau} | \{X_k\}_t) \prod_k dx_k \quad (2.58)
\]

and

\[
D^{(2)}_{ij}(\{X_k(t)\}, t) = D_{ij}(\{X_k(t)\}, t) = \frac{1}{2} \sum_m g_{im}(\{X_k\}, t) g_{jm}(\{X_k\}, t) \quad (2.60)
\]

\[
\times P(\{x_k\}_{t+\tau} | \{X_k\}_t) \prod_k dx_k. \quad (2.59)
\]

The drift coefficient can be thought of as describing how the peak of the probability distribution moves with time and the diffusion coefficient corresponds to how the shape of the distribution diffuses with time.

If the system of Eq. 2.51 is interpreted in the Itô sense, with the \( \xi \) as standard zero-mean unit intensity \( \delta \)-correlated Gaussian white noises such that their derivatives are standard Wiener processes then the Kramers-Moyal coefficients can be read directly [20],

\[
D_i(\{X_k(t)\}, t) = h_i(\{X_k(t)\}, t)
\]

\[
D_{ij}(\{X_k(t)\}, t) = \frac{1}{2} \sum_m g_{im}(\{X_k\}, t) g_{jm}(\{X_k\}, t) \quad (2.60)
\]

To recapitulate, a Markovian system with stochastic white noise forcing is described by a Langevin equation, Eq. 2.51, and a Fokker-Planck equation, Eq. 2.56. The dynamics of the system are determined by the Langevin coefficients. The Langevin coefficients, and any parameters they depend on,
are related to the Kramers-Moyal coefficients which can be calculated at finite times $\tau$ directly from measured data with Eq. 2.58 and 2.59 where the required conditional probability distributions can be calculated numerically with histograms.
3

Coupled Standing Modes

If two standing modes at the same frequency are dominant then the form of
the solution in terms of Eq. 2.49 is

\[ p(\theta, t) \approx \eta_A(t) \cos(\theta) + \eta_B(t) \sin(\theta) \] (3.1)

with time dependent pressure components

\[ \eta_A(t) = A(t) \cos(\omega t + \phi_A(t)) \]
\[ \eta_B(t) = B(t) \cos(\omega t + \phi_B(t)), \] (3.2)

or in full

\[ p(\theta, t) \approx A(t) \cos(\omega t + \phi_A(t)) \cos(\theta) + B(t) \cos(\omega t + \phi_B(t)) \sin(\theta). \] (3.3)

3.1 Harmonic Oscillator Equations

With Eq. 3.1 as the assumed solution, the model Eq. 2.44 can be developed
into a set of coupled harmonic oscillator equations for the modal pressure
amplitudes \( \eta_A \) and \( \eta_B \). Substituting Eq. 3.1 into the model Eq. 2.44 yields

\[
\frac{d^2 \eta_A}{dt^2} \cos(\theta) + \frac{d^2 \eta_B}{dt^2} \sin(\theta) + \omega^2 (\eta_A \cos(\theta) + \eta_B \sin(\theta)) \\
= 2\nu \left( \frac{d\eta_A}{dt} \cos(\theta) + \frac{d\eta_B}{dt} \sin(\theta) \right) \\
- 3\kappa \left( \frac{d\eta_A}{dt} \eta_A^2 \cos(\theta) + \frac{d\eta_A}{dt} \eta_B^2 \cos(\theta) \sin^2(\theta) + 2 \frac{d\eta_A}{dt} \eta_A \eta_B \cos^2(\theta) \sin(\theta) \right) \\
+ \frac{d\eta_B}{dt} \eta_A^2 \cos(\theta) \sin(\theta) + \frac{d\eta_B}{dt} \eta_B^2 \sin^3(\theta) + 2 \frac{d\eta_B}{dt} \eta_A \eta_B \cos(\theta) \sin^2(\theta) \right), \] (3.4)
which can be separated by using the orthogonality of the spatial wavefunctions.

For $\eta_A$ we multiply by $\cos(\theta)$ and integrate over a full revolution of $\theta$,

$$
\int_0^{2\pi} \left[ \frac{d^2\eta_A}{dt^2} \cos^2(\theta) + \frac{d^2\eta_B}{dt^2} \cos(\theta) \sin(\theta) + \omega^2 (\eta_A \cos^2(\theta) + \eta_B \cos(\theta) \sin(\theta)) \right] d\theta
$$

\begin{align*}
&= \int_0^{2\pi} \left[ 2\nu \left( \frac{d\eta_A}{dt} \cos^2(\theta) + \frac{d\eta_B}{dt} \cos(\theta) \sin(\theta) \right) \\
&- 3\kappa \left( \frac{d\eta_A}{dt} \eta_A^2 \cos^4(\theta) + \frac{d\eta_B}{dt} \eta_B^2 \cos^2(\theta) \sin^2(\theta) + 2 \frac{d\eta_A}{dt} \eta_A \eta_B \cos^3(\theta) \sin(\theta) \\
&+ \frac{d\eta_B}{dt} \eta_A \cos^3(\theta) \sin(\theta) + \frac{d\eta_B}{dt} \eta_B^2 \cos^3(\theta) \sin^2(\theta) + 2 \frac{d\eta_B}{dt} \eta_A \eta_B \cos(\theta) \sin^2(\theta) \right) \right] d\theta,
\end{align*}

(3.5)

to give

$$
\pi \frac{d^2\eta_A}{dt^2} + \pi \omega^2 \eta_A
$$

\begin{align*}
&= 2\nu \frac{d\eta_A}{dt} - 3\kappa \left( \frac{3\pi}{4} \frac{d\eta_A}{dt} \eta_A^2 + \frac{\pi}{4} \frac{d\eta_A}{dt} \eta_B^2 + \frac{\pi}{2} \frac{d\eta_B}{dt} \eta_A \eta_B \right),
\end{align*}

(3.6)

which reduces to

$$
\frac{d^2\eta_A}{dt^2} + \omega^2 \eta_A = 2\nu \frac{d\eta_A}{dt} - 3\kappa \left( \frac{d\eta_A}{dt} (3\eta_A^2 + \eta_B^2) + 2 \frac{d\eta_B}{dt} \eta_A \eta_B \right).
$$

(3.7)

For $\eta_B$ we repeat the process but multiply by $\sin(\theta)$ to give

$$
\frac{d^2\eta_B}{dt^2} + \omega^2 \eta_B = 2\nu \frac{d\eta_B}{dt} - 3\kappa \left( \frac{d\eta_B}{dt} (3\eta_B^2 + \eta_A^2) + 2 \frac{d\eta_A}{dt} \eta_B \eta_A \right).
$$

(3.8)

At this stage we introduce an additive noise term $\xi$ to each equation to account for the stochastic forcing of the heat release due to turbulence. $\xi$ is a delta-correlated Gaussian white noise with intensity $\Gamma$. This is a heuristic approach because we have not explicitly dealt with the stochastic term of the heat release rate up to now. The use of this simple noise term is justified by its
success in describing the single mode \[13, 14\] and previous work which shows it to be additive \[21\]. We then have a set of coupled oscillator equations for the time dependent pressure components of the modes at frequency $\omega$ that are determined by the parameters $\nu$, $\kappa$ and $\Gamma$.

$$
\begin{align*}
\begin{align*}
\frac{d^2 \eta_A}{dt^2} + \omega^2 \eta_A &= 2\nu \frac{d\eta_A}{dt} - \frac{3\kappa}{4} \left( \frac{d\eta_A}{dt} \left(3\eta_A^2 + \eta_B^2\right) + 2\frac{d\eta_B}{dt} \eta_A \eta_B\right) + \xi \\
\frac{d^2 \eta_B}{dt^2} + \omega^2 \eta_B &= 2\nu \frac{d\eta_B}{dt} - \frac{3\kappa}{4} \left( \frac{d\eta_B}{dt} \left(3\eta_B^2 + \eta_A^2\right) + 2\frac{d\eta_A}{dt} \eta_B \eta_A\right) + \xi
\end{align*}
\end{align*}
\tag{3.9}
$$

### 3.2 Langevin Equations

We now shift from the description of $\eta_A$ and $\eta_B$ to the amplitudes and phases $A$, $B$, $\phi_A$ and $\phi_B$ by applying deterministic \[22\] and stochastic \[23, 24\] averaging procedures.

When the Fourier solution was first introduced we mentioned that there were two timescales, the oscillation timescale $2\pi/\omega$ and the slower timescale of the amplitudes and phases. We assume that the amplitudes and phases can be treated as constant over an oscillation period. Differentiating $\eta_A$ and $\eta_B$ with respect to the oscillation timescale gives

\[
\begin{align*}
\eta_A &= A \cos(\omega t + \phi_A) \quad \rightarrow \quad \frac{d\eta_A}{dt} = -A\omega \sin(\omega t + \phi_A) \\
\eta_B &= B \cos(\omega t + \phi_B) \quad \rightarrow \quad \frac{d\eta_B}{dt} = -B\omega \sin(\omega t + \phi_B).
\end{align*}
\tag{3.10}
\]

From this we can generate the relations

\[
\begin{align*}
A &= \sqrt{\eta_A^2 + \left(\frac{1}{\omega} \frac{d\eta_A}{dt}\right)^2} \\
B &= \sqrt{\eta_B^2 + \left(\frac{1}{\omega} \frac{d\eta_B}{dt}\right)^2} \\
\phi_A &= -\arctan\left(\frac{1}{\omega \eta_A} \frac{d\eta_A}{dt}\right) - \omega t \\
\phi_B &= -\arctan\left(\frac{1}{\omega \eta_B} \frac{d\eta_B}{dt}\right) - \omega t.
\end{align*}
\tag{3.11}
\]

When differentiated, these lead to
\[
\frac{dA}{dt} = \frac{1}{A\omega^2} \frac{d\eta_A}{dt} \left( \frac{d^2\eta_A}{dt^2} + \omega^2\eta_A \right) \quad \frac{dB}{dt} = \frac{1}{B\omega^2} \frac{d\eta_B}{dt} \left( \frac{d^2\eta_B}{dt^2} + \omega^2\eta_B \right)
\]
\[
\frac{d\phi_A}{dt} = -\frac{\eta_A}{A^2\omega} \left( \frac{d^2\eta_A}{dt^2} + \omega^2\eta_A \right) \quad \frac{d\phi_B}{dt} = -\frac{\eta_B}{B^2\omega} \left( \frac{d^2\eta_B}{dt^2} + \omega^2\eta_B \right),
\]
from which we get
\[
\frac{dA}{dt} = \frac{1}{A\omega^2} \frac{d\eta_A}{dt} \left( 2\nu \frac{d\eta_A}{dt} - \frac{3\kappa}{4} \left( \frac{d\eta_A}{dt} (3\eta_A^2 + \eta_B^2) + 2\frac{d\eta_B}{dt} \eta_A \eta_B \right) \right) + \frac{1}{A\omega^2} \frac{d\eta_A}{dt} \xi, \quad (3.13)
\]
\[
\frac{dB}{dt} = \frac{1}{B\omega^2} \frac{d\eta_B}{dt} \left( 2\nu \frac{d\eta_B}{dt} - \frac{3\kappa}{4} \left( \frac{d\eta_B}{dt} (3\eta_B^2 + \eta_A^2) + 2\frac{d\eta_A}{dt} \eta_B \eta_A \right) \right) + \frac{1}{B\omega^2} \frac{d\eta_B}{dt} \xi, \quad (3.14)
\]
\[
\frac{d\phi_A}{dt} = -\frac{\eta_A}{A^2\omega} \left( 2\nu \frac{d\eta_A}{dt} - \frac{3\kappa}{4} \left( \frac{d\eta_A}{dt} (3\eta_A^2 + \eta_B^2) + 2\frac{d\eta_B}{dt} \eta_A \eta_B \right) \right) - \frac{\xi}{A^2\omega}, \quad (3.15)
\]
\[
\frac{d\phi_B}{dt} = -\frac{\eta_B}{B^2\omega} \left( 2\nu \frac{d\eta_B}{dt} - \frac{3\kappa}{4} \left( \frac{d\eta_B}{dt} (3\eta_B^2 + \eta_A^2) + 2\frac{d\eta_A}{dt} \eta_B \eta_A \right) \right) - \frac{\xi}{B^2\omega}. \quad (3.16)
\]
Considering only Eq. 3.13 for \(dA/dt\) and substituting Eq. 3.2 and 3.10,
\[
\frac{dA}{dt} = 2\nu \sin^2(\omega t + \phi_A) \\
- \frac{3\kappa}{4} (A \sin^2(\omega t + \phi_A) (3A^2 \cos^2(\omega t + \phi_A) + B^2 \cos^2(\omega t + \phi_B))) \\
+ 2AB^2 \sin(\omega t + \phi_A) \sin(\omega t + \phi_B) \cos(\omega t + \phi_A) \cos(\omega t + \phi_B)) \\
- \frac{\sin(\omega t + \phi_A)\xi}{\omega}. 
\] (3.17)

Through some trigonometric manipulation it is possible to separate the fast oscillation terms to give

\[
\frac{dA}{dt} = \left[\nu A - \frac{3\kappa}{32} (3A^2 + (2 + \cos(2\Phi)) B^2) A \right] \\
- \nu A \cos(2\omega t + 2\phi_A) + \frac{3\kappa}{32} A (3A^2 \cos(4\omega t + 4\phi_A) \\
+ 6B^2 \cos(4\omega t + 2(\phi_A + \phi_B)) \\
+ 4B^2 (\cos(2\omega t + 2\phi_A) - \cos(2\omega t + 2\phi_B))) \\
+ \left[\frac{-\sin(\omega t + \phi_A)\xi}{\omega}\right], 
\] (3.18)

where \( \Phi = \phi_A - \phi_B \). Expressed in this way it is apparent that when averaged over an oscillation period, all of the deterministic oscillating terms vanish leaving only

\[
\frac{dA}{dt} = \nu A - \frac{3\kappa}{32} (3A^2 + (2 + \cos(2\Phi)) B^2) A + \left[\frac{-\sin(\omega t + \phi_A)\xi}{\omega}\right]. 
\] (3.19)

The same steps are applied to \( B, \phi_A \) and \( \phi_B \).

Following the stochastic averaging procedure \[23, 24\], we define a new stochastic variable

\[
\zeta = -\sin(\omega t + \phi_A)\xi. 
\] (3.20)
Considering a time interval $\Delta$, where $\tau_\xi << \Delta << \tau_A$, $\tau_{\phi A}$ and $\tau_\xi$ is the correlation time of the noise source $\xi$, which in reality is quasi- not $\delta$-correlated,

$$\zeta = -\sin(\omega t + \phi_A)\xi \approx -\sin(\omega t + \phi_\Delta)\xi - \cos(\omega t + \phi_\Delta)\xi \Delta \phi_A, \quad (3.21)$$

where

$$\phi_\Delta = \phi_A(t - \Delta) \quad (3.22)$$

and

$$\Delta \phi_A = \phi_A - \phi_\Delta. \quad (3.23)$$

$\phi_\Delta$ is not correlated with $\xi_A$ since $\tau_\xi << \Delta << \tau_{\phi A}$, therefore

$$<\zeta> \approx -\cos(\omega t + \phi_A) <\xi_A \Delta \phi_A>. \quad (3.24)$$

Integrating the phase, where as for the amplitude the fast oscillation terms have been averaged out,

$$\Delta \phi_A = \int_{t-\Delta}^t -\frac{1}{A\omega^2} \xi \cos(\omega t + \phi_A)$$

$$= \int_{-\Delta}^0 -\frac{1}{\omega A(t + \tau)} \xi(t + \tau) \cos(\omega(t + \tau) + \phi_A(t + \tau))d\tau. \quad (3.25)$$

As $\xi$ is stationary and $\Delta << \tau_A, \tau_{\phi A}$

$$<\xi \Delta \phi_A> = -\frac{1}{\omega A} \int_{-\Delta}^0 <\xi \xi_\tau> \cos(\omega(t + \tau) + \phi_A)d\tau \quad (3.26)$$

and

$$<\zeta> = \frac{1}{\omega A} \int_{-\Delta}^0 <\xi \xi_\tau> \cos(\omega t + \phi_A) \cos(\omega(t + \tau) + \phi_A)d\tau. \quad (3.27)$$

Expanding the products of cosines and omitting rapidly oscillating terms then gives

$$<\zeta> = \frac{1}{2\omega A} \int_{-\infty}^0 <\xi \xi_\tau> \cos(\omega \tau)d\tau = \frac{\pi S_{\xi \xi}(\omega)}{2\omega A} = \frac{\Gamma}{4\omega A}. \quad (3.28)$$

We now intend to make a substitution of $\zeta$ with the zero-mean process
\[ \zeta' = \zeta - \langle \zeta \rangle \approx -\xi \sin(\omega t + \phi - \Delta) \] (3.29)

so that our equations can be interpreted as Langevin equations. Assuming \( \delta \)-correlation,

\[
\langle \zeta' \zeta' \rangle = \int_{-\infty}^{\infty} <\zeta' \zeta' > d\theta \delta(\tau) = \int_{-\infty}^{\infty} <\xi \sin(\omega t + \phi - \Delta) \xi \sin(\omega(t + \theta) + \phi - \Delta) > d\theta \delta(\tau). \quad (3.30)
\]

\[
< \zeta' \zeta' > = \frac{1}{2} \int_{-\infty}^{\infty} <\xi \xi > \cos(\omega \theta) d\theta \delta(\tau) = \pi S_{\xi \xi}(\omega) \delta(\tau) = \frac{\Gamma}{2} \delta(\tau) \quad (3.31)
\]

We can then write

\[
-\sin(\omega t + \phi A) \xi = \frac{\zeta}{\omega} = \frac{1}{\omega}(\zeta' + <\zeta>) = \frac{\Gamma}{4\omega^2 A} + \chi, \quad (3.32)
\]

where we have defined \( \chi = \zeta' / \omega \), with

\[
< \chi \chi > = \frac{\Gamma}{2\omega^2} \delta(\tau) \quad (3.33)
\]

\[
< \chi > = 0.
\]

We then have a Langevin equation for the amplitude \( A \),

\[
\frac{dA}{dt} = \nu A - \frac{3\kappa}{32} [3A^2 + (2 + \cos(2\Phi)) B^2] A + \frac{\Gamma}{4\omega^2 A} + \chi. \quad (3.34)
\]

Repeating the process for \( B, \phi_A \) and \( \phi_B \) yields further Langevin equations which form a system of coupled equations as in Eq. 2.51. The system can be simplified by taking the difference \( \Phi = \phi_A - \phi_B \) and reducing the number of explicit variables to three.

\[
(A, B, \Phi) \left\{ \begin{array}{l}
\frac{dA}{dt} = \nu A - \frac{3\kappa}{32} [3A^2 + (2 + \cos(2\Phi)) B^2] A + \frac{\Gamma}{4\omega^2 A} + \chi \\
\frac{dB}{dt} = \nu B - \frac{3\kappa}{32} [3B^2 + (2 + \cos(2\Phi)) A^2] B + \frac{\Gamma}{4\omega^2 B} + \chi \\
\frac{d\Phi}{dt} = \frac{3\kappa}{32} (A^2 + B^2) \sin(2\Phi) + (\frac{A + B}{AB}) \chi
\end{array} \right. \quad (3.35)
\]
We now introduce an amendment whereby we state that the modes can have different parameters while satisfying the same equation \([19]\). In practice this would be due to any irregularities in the spatial symmetry, causing orthogonal modes to ‘see’ different surroundings. The system of equations is then

\[
\begin{align*}
\frac{dA}{dt} &= \nu_A A - \frac{3\kappa_A}{32} (3A^2 + (2 + \cos(2\Phi))B^2)A + \frac{\Gamma_A}{4\omega^2 A} + \chi_A \\
\frac{dB}{dt} &= \nu_B B - \frac{3\kappa_B}{32} (3B^2 + (2 + \cos(2\Phi))A^2)B + \frac{\Gamma_B}{4\omega^2 B} + \chi_B \\
\frac{d\Phi}{dt} &= \frac{3}{32} (\kappa_A A^2 + \kappa_B B^2) \sin(2\Phi) + \left(\frac{A + B}{AB}\right) \chi_\Phi,
\end{align*}
\]  

(3.36)

and is determined by the set of parameters \(\{\nu\}, \{\kappa\} \text{ and } \{\Gamma\}\), with independent \(\delta\)-correlated, zero mean white noises

\[\langle \chi_i \rangle = 0\]  

(3.37)

and

\[\langle \chi_i \tau \chi_j \rangle = \langle \chi_i(t + \tau) \chi_j(t) \rangle = \frac{\sqrt{\Gamma_i \Gamma_j}}{2\omega^2} \delta_{ij} \delta(\tau).\]  

(3.38)

With the stochastic terms defined this way the requirements are satisfied to express the equations in the Itô sense.

\[
\frac{d}{dt} \begin{pmatrix} A \\ B \\ \Phi \end{pmatrix} = \begin{pmatrix} \nu_A A - \frac{3\kappa_A}{32} (3A^2 + (2 + \cos(2\Phi))B^2)A + \frac{\Gamma_A}{4\omega^2 A} \\ \nu_B B - \frac{3\kappa_B}{32} (3B^2 + (2 + \cos(2\Phi))A^2)B + \frac{\Gamma_B}{4\omega^2 B} \\ \frac{3\kappa_\Phi}{32} (A^2 + B^2) \sin(2\Phi) \end{pmatrix} + \begin{pmatrix} \sqrt{\frac{\Gamma_A}{2\omega^2}} & 0 & 0 \\ 0 & \sqrt{\frac{\Gamma_B}{2\omega^2}} & 0 \\ 0 & 0 & \left(\frac{A + B}{AB}\right) \sqrt{\frac{\Gamma_\Phi}{2\omega^2}} \end{pmatrix} \begin{pmatrix} \hat{\chi}_A \\ \hat{\chi}_B \\ \hat{\chi}_\Phi \end{pmatrix},
\]  

(3.39)
where the $\chi$ have been rephrased in terms of unit intensity uncorrelated white noises $\hat{\chi}$.

\[
< \hat{\chi}_i(t + \tau)\hat{\chi}_j(t) > = \delta_{ij}\delta(\tau) \tag{3.40}
\]

\[
\chi_i = \sqrt{\frac{\Gamma_i}{2\omega^2}} \hat{\chi}_i \tag{3.41}
\]

\[
< \chi_i(t + \tau)\chi_j(t) > = \sqrt{\frac{\Gamma_i \Gamma_j}{2\omega^2}} < \hat{\chi}_i(t + \tau)\hat{\chi}_j(t) >
= \sqrt{\frac{\Gamma_i \Gamma_j}{2\omega^2}} \delta_{ij}\delta(\tau) \tag{3.42}
\]

### 3.3 Drift and Diffusion Coefficients

The drift and diffusion coefficients are read directly according to Eq. 2.60 as

\[
D^{(1)} = \begin{pmatrix}
D_{AA}^{(1)} \\
D_{AB}^{(1)} \\
D_{\Phi A}^{(1)} \\
D_{\Phi B}^{(1)}
\end{pmatrix} = \begin{pmatrix}
\nu_A A - \frac{3\kappa_A}{32} (3A^2 + (2 + \cos(2\Phi))B^2)A + \frac{\Gamma_A}{4\omega^2}A \\
\nu_B B - \frac{3\kappa_B}{32} (3B^2 + (2 + \cos(2\Phi))A^2)B + \frac{\Gamma_B}{4\omega^2}B \\
\frac{3\kappa_\Phi}{32} (A^2 + B^2) \sin(2\Phi)
\end{pmatrix}
\tag{3.43}
\]

and

\[
D^{(2)} = \begin{pmatrix}
D_{AA}^{(2)} & D_{AB}^{(2)} & D_{\Phi A}^{(2)} \\
D_{BA}^{(2)} & D_{BB}^{(2)} & D_{\Phi B}^{(2)} \\
D_{\Phi A}^{(2)} & D_{\Phi B}^{(2)} & D_{\Phi\Phi}^{(2)}
\end{pmatrix} = \begin{pmatrix}
\Gamma_A & 0 & 0 \\
0 & \frac{\Gamma_B}{4\omega^2} & 0 \\
0 & 0 & \left(\frac{A + B}{AB}\right)^2 \frac{\Gamma}{4\omega^2}
\end{pmatrix} \tag{3.44}
\]

### 3.4 Change of Variables

We finally implement a change of variables, to make the phase difference implicit and ease the coming calculations.
{A, B, Φ} → {A, ̂B} or { ̂A, B} \hspace{1cm} (3.45)

with

\[ ̂A = (2 + \cos(2\Phi))^{1/2}A \]
\[ ̂B = (2 + \cos(2\Phi))^{1/2}B \] \hspace{1cm} (3.46)

to give

\[ \frac{dA}{dt}(A, ̂B) = \nu_A A - \frac{3\kappa_A}{32}(3A^2 + ̂B^2)A + \frac{\Gamma_A}{4\omega^2A} + \sqrt{\frac{\Gamma_A}{2\omega^2}}\chi_A \]
\[ \frac{dB}{dt}( ̂A, B) = \nu_B B - \frac{3\kappa_B}{32}(3B^2 + ̂A^2)B + \frac{\Gamma_B}{4\omega^2B} + \sqrt{\frac{\Gamma_B}{2\omega^2}}\chi_B. \] \hspace{1cm} (3.47)

3.5 Simulation

The pressure signal is simulated in SIMULINK with the coupled oscillator equations, Eq. 3.9, by rearranging and replacing the time derivatives with the Laplacian \( s^2 = i\omega \) to give

\[ \eta_A = \frac{s}{s^2 - 2\nu_A + \omega^2} \left( \frac{\xi_A}{s} - \frac{3\kappa_A}{4} \left( \eta_A^3 + \eta_A\eta_B^2 \right) \right) \] \hspace{1cm} (3.48)

and

\[ \eta_B = \frac{s}{s^2 - 2\nu_B + \omega^2} \left( \frac{\xi_B}{s} - \frac{3\kappa_B}{4} \left( \eta_B^3 + \eta_B\eta_A^2 \right) \right) \] \hspace{1cm} (3.49)

The block diagram is given in Fig. 3.5.
Figure 3.1: SIMULINK block diagram for standing modal pressures $\eta_A$ and $\eta_B$. 
4

System Identification

4.1 Method

Based on Eq. 3.47, the parameters that determine the dynamics of the coupled thermoacoustic modes are $\nu_A$ and $\nu_B$, the growth rates, $\kappa_A$ and $\kappa_B$, the non-linear flame coefficients and $\Gamma_A$ and $\Gamma_B$, the noise intensities.

The algorithm that has previously been used to estimate these system parameters in the case of the single mode [14, 25] is, when applied to the coupled-modes, is as follows,

1. Simulate the system for a set time, $T$, with a given sampling frequency $f_S$. The simulation is initialised in a resting-state and excited by the noises. A transient signal therefore occurs first and this is discarded.

2. Extract the time-series data of the relevant degrees of freedom. For us this is the amplitudes $A$ and $B$ and the phase difference $\Phi$. They are extracted with the Hilbert transform of the pressure signals.

$$A = |\mathcal{H}(\eta_A)|$$ (4.1)

$$B = |\mathcal{H}(\eta_B)|$$ (4.2)

$$\Phi = \phi_A - \phi_B = \text{Im}[\ln \mathcal{H}(\eta_A)] - \text{Im}[\ln \mathcal{H}(\eta_B)].$$ (4.3)

$\hat{A}$ and $\hat{B}$ are calculated from these with Eq. 3.46.

3. Choose time shifts $\tau$ and bin resolutions $\Delta A$, $\Delta B$, $\Delta \hat{A}$ and $\Delta \hat{B}$.

4. Data bin the signals and calculate the transition probability densities with histograms according to

$$P(\{a, \hat{b}\}_{t+\tau} | \{A, \hat{B}\}_t) = \frac{W(\{a, \hat{b}\}_{t+\tau}, \{A, \hat{B}\}_t)}{W(\{A, \hat{B}\}_t)} \bigg|_{\{a, \hat{b}\}_t = \{A, \hat{B}\}_t}$$ (4.4)

and
\begin{equation}
P(\{\hat{a},b\}_{t+\tau}|\{\hat{A},B\}_t) = \frac{W(\{\hat{a},b\}_{t+\tau},\{\hat{A},B\}_t)}{W(\{\hat{A},B\}_t)}|_{\{\hat{a},b\}_t=\{\hat{A},B\}_t}.
\end{equation}

Note that we use the notation of majuscule letters for the original signals and minuscule letters for the time shifted signals. MATLAB has no built-in function for probability distributions of more than two dimensions so the four dimensional joint probability distributions are calculated by taking two dimensional slices.

5. Calculate the finite-time Kramers-Moyal coefficients, Eq. 2.58 and 2.59 without the limit \( \tau \to 0 \),

\begin{equation}
D^{(1)}_A(A,\hat{B},\tau) = \frac{1}{\tau} \int_{-\infty}^{\infty} (a - A) P(\{a,\hat{b}\}_{t+\tau}|\{A,\hat{B}\}_t)dad\hat{b} \tag{4.6}
\end{equation}

\begin{equation}
D^{(1)}_B(\hat{A},B,\tau) = \frac{1}{\tau} \int_{-\infty}^{\infty} (b - B) P(\{\hat{a},b\}_{t+\tau}|\{\hat{A},B\}_t) d\hat{a}db \tag{4.7}
\end{equation}

\begin{equation}
D^{(2)}_{AA}(A,\hat{B},\tau) = \frac{1}{2\tau} \int_{-\infty}^{\infty} (a - A)^2 P(\{a,\hat{b}\}_{t+\tau}|\{A,\hat{B}\}_t)dad\hat{b} \tag{4.8}
\end{equation}

\begin{equation}
D^{(2)}_{BB}(\hat{A},B,\tau) = \frac{1}{2\tau} \int_{-\infty}^{\infty} (b - B)^2 P(\{\hat{a},b\}_{t+\tau}|\{\hat{A},B\}_t)d\hat{a}db. \tag{4.9}
\end{equation}

6. Extrapolate the finite-time Kramers-Moyal coefficients to the limit \( \tau = 0 \) to estimate the drift and diffusion coefficients. The extrapolation is done element by element according to

\begin{equation}
D(A_i,\hat{B}_j,\tau) = D(A_i,\hat{B}_j,0)e^{k_{ij}\tau} \tag{4.10}
\end{equation}

and a least squares fit. This has no particular analytical basis as there is no known analytical relation, instead it is justified by the fact that it works for the single-mode system identification. The range of \( \tau \) values to include in the fit must be chosen.

7. Fit the extrapolated drift and diffusion coefficients to the analytical expressions, Eq. 3.43 and 3.44, to extract the parameters. This is again done on a least squares basis.

We first fit the diffusion coefficients to extract \( \Gamma_A \) and \( \Gamma_B \) then use these values in the equations for the fitting of the drift coefficients. The amplitude ranges to be included in the fitting must, as for \( \tau \), be chosen.
Note that, because we have used the change of variables to make the phase difference implicit, steps 4. and 5. must essentially be done twice to account for the different transition probabilities. This is less computationally intensive and easier to manage than the three variable case with the explicit phase difference.

The above algorithm is realised through MATLAB and the code can be found in Appendix B.1. Graphical examples of each step are given in Appendix A Fig. A.1 through A.11 where the following parameters have been used.

- $\nu_A = 5$ Hz and $\nu_B = 2$ Hz
- $\kappa_A = \kappa_B = 3$ Hz Pa$^{-2}$
- $\Gamma_A/4\omega^2 = \Gamma_B/4\omega^2 = 1$
- $\omega = 2\pi \times 150$ Hz
- $\Delta A = \Delta B = \Delta \hat{A} = \Delta \hat{B} = 0.1$ Pa (approximately 30-50 bins per amplitude)
- $n_r = 50$ with $\Delta \tau = 0.005$ s
- $T = 500$ s
- $f_S = 4000$ Hz $\gg \omega$
- Transient cut-off at 0.5 s

4.2 Results

Table 4.1 gives the results of the system identification for a sample of tests. We can see that overall the agreement is good, with the parameters at least predicted to the right order of magnitude and often fairly accurately. If one mode is dominant, with a higher growth rate, then the results for that mode are particularly accurate. If one mode has a positive growth rate and the other a negative growth rate then physically the negative mode is not experiencing a positive feedback loop (naturally if the growth rate is negative it is not ‘growing’) and is barely present in the signal, which is reflected in the results. This is satisfactory because the system identification need only be able to identify rather broadly whether there is a mode that requires attention. Also worth noting is that if the growth rate is particularly small then the estimation of the non-linear heat coefficient is off because the non linear term is secondary to the linear term and if the linear term itself is small then the nonlinear term will be negligible. Additionally, the growth rates chosen here are approaching the limit of smallness considering that, for example, the ratio $\nu/\omega$ with $\nu = 5$ Hz and $\omega = 150$ Hz is around 3% so these results also show applicability at this extreme.
The implementation of the system identification is not perfect and the extrapolation and fitting process can be optimised further with many parameters to vary:

- Simulation time $T$
- Sampling frequency $f_S$
- Amplitude resolutions $\Delta A$, $\Delta B$, $\Delta \dot{A}$ and $\Delta \dot{B}$
- Type of fitting (e.g. least squares)
- The equation to fit the $\tau$ dependence of the finite-time Kramers-Moyal coefficients
- The range of $\tau$ values to include in the fit
- The range of amplitudes to fit

At this stage the code also requires further refinement to improve the computational efficiency. In particular the nested for-loops used to calculate the transition probabilities and finite-time Kramers-Moyal coefficients are the primary time sinks, with the run-time of these calculations taking approximately one hour with between 30 and 50 bins per amplitude and 50 time shifts. Further investigation of the suggested parameters above in a timely manner would require this problem to be solved first.
Further Investigations

5.1 AFPE-Based System Identification

A second method of system identification based on iterative solutions of the adjoint Fokker-Planck equation that does not require extrapolation has been suggested and implemented for the single-mode oscillator by Boujo and Noiray [14].

The adjoint Fokker-Planck equation is

\[
\frac{\partial}{\partial t}W^\dagger(\{x_k(t)\}, t) = L^\dagger W^\dagger(\{x_k(t)\}, t),
\]

where the adjoint Fokker-Planck operator \(L^\dagger\) is defined as

\[
L^\dagger = -\sum_i^N D^{(1)}_i(\{x_k(t)\}) \frac{\partial}{\partial x_i} + \sum_i^N \sum_j^N D^{(2)}_{ij}(\{x_k(t)\}) \frac{\partial^2}{\partial x_i \partial x_j}
\]

and it can be shown that the solution of the adjoint Fokker-Planck equation with a specific initial condition is related to the finite-time Kramers-Moyal coefficients [14, 26, 27],

\[
P^\dagger(a, b, \phi, 0) = (a - A)^1 \rightarrow D^{(1)}_A(A, b, \phi, \tau) = \frac{1}{\tau}P^\dagger(A, b, \phi, \tau)
\]

\[
P^\dagger(a, b, \phi, 0) = (a - A)^2 \rightarrow D^{(2)}_{AA}(A, b, \phi, \tau) = \frac{1}{2\tau}P^\dagger(A, b, \phi, \tau)
\]

\[
P^\dagger(a, b, \phi, 0) = (b - B)^1 \rightarrow D^{(1)}_B(a, B, \phi, \tau) = \frac{1}{\tau}P^\dagger(a, B, \phi, \tau)
\]

\[
P^\dagger(a, b, \phi, 0) = (b - B)^2 \rightarrow D^{(2)}_{BB}(a, B, \phi, \tau) = \frac{1}{2\tau}P^\dagger(a, B, \phi, \tau).
\]

The algorithm for system identification is then:
1. - 5. As for the extrapolation based method.

6. Set initial estimates of the values $\nu_A$, $\nu_B$, $\kappa_A$, $\kappa_B$, $\Gamma_A$ and $\Gamma_B$.

7. Using the current parameter estimates compute the finite time Kramers-Moyal coefficients by solution of the adjoint Fokker-Planck equation according to Eq. 5.3 through 5.6.

8. Evaluate the error between the results of 5. and 7.

9. If the error is within the desired tolerance then the parameters have been found, else update 7. and reiterate.

This method rests on the ability to solve the adjoint Fokker-Planck equation efficiently such that it can be solved iteratively to reach a solution.

The partial differential equation toolbox in MATLAB has finite element capability for solving parabolic partial differential equations of the form

$$m \frac{\partial^2 x}{\partial t^2} + d \frac{\partial x}{\partial t} - \nabla \cdot (c \nabla x) + ax = f,$$

(5.7)

and we can write the adjoint Fokker-Planck equation to match this,

$$\frac{\partial W^\dagger}{\partial t} = \left( D_A \frac{\partial W^\dagger}{\partial A} + D_B \frac{\partial W^\dagger}{\partial B} + D_\Phi \frac{\partial W^\dagger}{\partial \Phi} \right)$$

$$+ \left( D_{AA} \frac{\partial^2 W^\dagger}{\partial A^2} + D_{BB} \frac{\partial^2 W^\dagger}{\partial B^2} + D_{\Phi\Phi} \frac{\partial^2 W^\dagger}{\partial \Phi^2} \right),$$

(5.8)

$$0 \cdot \frac{\partial^2 W^\dagger}{\partial t^2} + 1 \cdot \frac{\partial W^\dagger}{\partial t} - \nabla \cdot (D^{(2)} \nabla W^\dagger) + 0 \cdot W^\dagger = D^{(1)} \cdot \nabla W^\dagger.$$

(5.9)

This is readily implemented in MATLAB, as shown in Appendix B.2, however a single solution for a given initial condition takes hours to run. This makes it incompatible with the iterative procedure and means that a single software solution with MATLAB is unlikely to be possible. A joint solution with another finite element software such as COMSOL may be possible. Alternatively, if we consider a system of reduced complexity with a fixed phase difference then the number of variables is reduced from three to two and this is less computationally demanding.
5.2 Coupled Rotating Modes

If two rotating modes, rather than standing waves, at the same frequency are dominant then the form of Eq. 2.49 is

\[
p(\theta, t) = F(t) \cos(\omega t + \phi_F + \theta) + G(t) \cos(\omega t + \phi_G - \theta)
\]

\[
= F(t) \cos(\omega t + \phi_F) \cos(\theta) - F(t) \sin(\omega t + \phi_F) \sin(\theta)
\]

\[
+ G(t) \cos(\omega t + \phi_G) \cos(\theta) + G(t) \sin(\omega t + \phi_G) \sin(\theta).
\]

Using the same methods as for the standing modes one can reach a set of coupled oscillator equations and Langevin equations [28]. Inspection of the second equality of Eq. 5.10 reveals that it is not possible to separate the modal pressures by orthogonality as was done for the standing modes because both \( F \) and \( G \) share spatial wave-functions when \( \theta \) is separated. The pressure can therefore not be directly simulated in SIMULINK and the amplitudes must be simulated directly from the Langevin equations,

\[
\begin{align*}
\frac{dF}{dt} &= 2\nu_F F - 3\kappa_F \left( \frac{1}{2} F^3 + FG^2 \right) + \frac{\Gamma_F}{4\omega^2 F} + \sqrt{\frac{\Gamma_F}{2\omega^2}} \chi_F, \\
\frac{dG}{dt} &= 2\nu_G G - 3\kappa_G \left( \frac{1}{2} G^3 + GF^2 \right) + \frac{\Gamma_G}{4\omega^2 G} + \sqrt{\frac{\Gamma_G}{2\omega^2}} \chi_G,
\end{align*}
\]

where the phase is naturally uncoupled from the amplitudes and we have retained our definition of the growth rate including a factor of a half. The block diagram is given in Fig. 5.2.

An issue arises with this approach to the simulation because both positive and negative values for \( F \) and \( G \) are mathematically permitted but only positive values of the pressure envelope amplitudes are physical. We have been unable to find a consistent method to prevent crossing into the negative potential well and have instead used a ‘cheat’ of taking the absolute values to keep them physical.

The system identification methods are then as for the standing modes. However, in adapting the code for the rotating modes, see Appendix B.3, we have encountered unusual results in the intermediate step of calculating the transition probabilities.

While the joint probability density \( W(F,G) \) appears correct compared to the analytical solution, the transition probabilities \( P(\{f,g\}_{t+\tau}|\{F,G\}_t) \) and joint probability densities \( P(\{f,g\}_{t+\tau}, \{F,G\}_t) \) lack distinct central maxima and instead show rings around areas of zero probability. This effect is more
pronounced at small time shifts $\tau$ and the missing portion of the probability distribution fills out as it diffuses. This can be seen in Appendix A Fig. A.14 through A.16. This seems to suggest that $F$ and $G$ cannot hold the same values when measured a short time later. At present it is not understood whether this is physically correct or due to error, possibly due to fault in the simulation.

As such the system identification methods have not been applied to the rotating modes. To test them we must be sure that we are simulating data that is correctly described by the model because the system identification methods are specifically tuned to the model equations.
Figure 5.1: SIMULINK block diagram for rotating mode amplitudes $F$ and $G$. 
6

Conclusions

A model-based output-only extrapolation based system identification method previously used for single-mode thermoacoustic oscillators has been extended to the case of two coupled standing modes. The code behind the system identification is functional, with the initial results proving to be reasonable. However, further improvement would be beneficial.

There are many parameters in the simulation, extrapolation and fitting processes that can be varied:

- Simulation time $T$. Higher simulation time will yield more data and therefore should provide better statistics but is counterbalanced by increased computation time.

- Simulated frequency $\omega$ and sampling frequency $f_S$. $f_S$ must be much greater than $\omega$ for the fast time-scale oscillations to be accurately represented but fewer sampling points would provide less data and improve the speed of the system identification methods.

- Amplitude resolutions $\Delta_A$, $\Delta_B$, $\Delta\hat{A}$ and $\Delta\hat{B}$. Finer resolution should provide more accurate estimates of the transition probabilities and finite-time Kramers-Moyal coefficients but comes at the cost of increased computation time.

- Type of fitting. Least squares has been used but more complex fitting procedures are possible, see for example [29].

- The equation to fit the $\tau$ dependence of the finite-time Kramers-Moyal coefficients.

- The range of $\tau$ values to include in the fit. Typically, the estimated finite-time Kramers-Moyal coefficients are not useful at extremely small or large values of $\tau$.

- The range of amplitudes to include in the fit. Our code provides good estimates of the drift and diffusion coefficients within a limited range of amplitudes, typically diverging from the analytical values at high amplitudes.
Thorough investigation of these parameters would require further optimisation of the code to reduce computation time. The calculation of transition probabilities and finite-time Kramers-Moyal coefficients is the primary time-sink due to the inefficiency of the nested for-loops used. The computation time also limits applicability of the solution in its present state to real-time system identification of experimental data.

A second recently proposed method based on iterative solutions to the adjoint Fokker-Planck equation has also been explored but a viable single software solution was not found in MATLAB due to the time required to solve the three dimensional adjoint Fokker-Planck equation. A combined software solution approach may be possible with an alternative finite element software such as COMSOL.

Finally, the case of coupled rotating modes has also been briefly explored but peculiar results for the transition probabilities warrant further investigation before a system identification method can be tested.
A

Figures
Figure A.1: Simulated pressure signals $\eta_A$ and $\eta_B$, with the transient signals (dashed) and Hilbert amplitude envelopes (black). $\omega = 2\pi \times 150$ Hz, $\nu_A = 5$ Hz, $\nu_B = 2$ Hz, $\kappa_A = \kappa_B = 3$ Hz Pa$^{-2}$, $\Gamma_A/4\omega^2 = \Gamma_B/4\omega^2 = 1$. 

40
Figure A.2: Fourier transforms of the simulated pressure signals $\eta_A$ and $\eta_B$, showing the correct pronounced frequency peak at $f = 150$ Hz. $\omega = 2\pi \times 150$ Hz, $\nu_A = 5$ Hz, $\nu_B = 2$ Hz, $\kappa_A = \kappa_B = 3$Hz Pa$^{-2}$, $\Gamma_A/4\omega^2 = \Gamma_B/4\omega^2 = 1$. 
Figure A.3: Joint probability distribution of the amplitudes $A$ and $B$, $W(A, B)$. Black: deterministic values. Blue: mean values. $\omega = 2\pi \times 150$ Hz. $\nu_A = 5$ Hz, $\nu_B = 2$ Hz, $\kappa_A = \kappa_B = 3\text{Hz Pa}^{-2}$, $\Gamma_A/4\omega^2 = \Gamma_B/4\omega^2 = 1$. As $\nu_A > \nu_B$, the $A$ mode dominates and the distribution is skewed towards higher $A$ values than $B$ values.
Figure A.4: Example slice of the joint probability distribution $W(\{a, b\}_{t+\tau}, \{A, B\}_{t})$ at $(A, B, \tau) = (1.75, 2.25, 0.015)$, $\omega = 2\pi \times 150$ Hz. $\nu_A = 5$ Hz, $\nu_B = 2$ Hz, $\kappa_A = \kappa_B = 3$ Hz Pa$^{-2}$, $\Gamma_A / 4\omega^2 = \Gamma_B / 4\omega^2 = 1$. 

43
Figure A.5: Example slice of the conditional probability distribution $P(\{a,b\}_{t+\tau}\mid\{A,B\}_t)$ at $(A,B,\tau) = (1.75, 2.25, 0.015)$. $\omega = 2\pi \times 150$ Hz. $\nu_A = 5$ Hz, $\nu_B = 2$ Hz, $\kappa_A = \kappa_B = 3$ Hz Pa$^{-2}$, $\Gamma_A/4\omega^2 = \Gamma_B/4\omega^2 = 1$. 
Figure A.6: Example slice of the joint probability distribution $W(\{\hat{a}, b\}_{t+\tau}, \{\hat{A}, B\}_t)$ at $(\hat{A}, B, \tau) = (3.05, 1.35, 0.015)$, $\omega = 2\pi \times 150$ Hz. $\nu_A = 5$ Hz, $\nu_B = 2$ Hz, $\kappa_A = \kappa_B = 3$ Hz Pa$^{-2}$, $\Gamma_A/4\omega^2 = \Gamma_B/4\omega^2 = 1$. 
Figure A.7: Example slice of the conditional probability distribution $P(\{\hat{a}, b\}_t|\{\hat{A}, B\}_t)$ at $(\hat{A}, B, \tau) = (3.05, 1.35, 0.015)$. $\omega = 2\pi \times 150$ Hz. $\nu_A = 5$ Hz, $\nu_B = 2$ Hz, $\kappa_A = \kappa_B = 3$ Hz Pa$^{-2}$, $\Gamma_A/4\omega^2 = \Gamma_B/4\omega^2 = 1$. 
Figure A.8: Example of fitting $D_A^{(1)}$ to $\tau$ with $(A_i, B_j) = (2.04, 0.45)$. The analytical drift value is marked with a black asterisk at $\tau = 0$. $\omega = 2\pi \times 150$ Hz. $\nu_A = 5$ Hz, $\nu_B = 2$ Hz, $\kappa_A = \kappa_B = 3$ Hz Pa$^{-2}$, $\Gamma_A/4\omega^2 = \Gamma_B/4\omega^2 = 1$. 

47
Figure A.9: Example of fitting $D_A^{(2)}$ to $\tau$ with $(A_i, \hat{B}_j) = (2.04, 0.45)$. The analytical diffusion value is marked with a black asterisk at $\tau = 0$. $\omega = 2\pi \times 150$ Hz. $\nu_A = 5$ Hz, $\nu_B = 2$ Hz, $\kappa_A = \kappa_B = 3$ Hz Pa$^{-2}$, $\Gamma_A/4\omega^2 = \Gamma_B/4\omega^2 = 1$. 
Figure A.10: Example fit of the extrapolated $D_A^{(1)}$ (mesh) to the analytical expression (solid transparent surface). The fitted surface is the solid opaque surface. $\omega = 2\pi \times 150$ Hz. $\nu_A = 5$ Hz, $\nu_B = 2$ Hz, $\kappa_A = \kappa_B = 3$ Hz Pa$^{-2}$, $\Gamma_A/4\omega^2 = \Gamma_B/4\omega^2 = 1$. The fitted values are $\nu_A = 4.33$ Hz, $\kappa_A = 2.45$ Hz and $\Gamma_A/4\omega^2 = 1.06$. 
Figure A.11: Example fit of the extrapolated $D_A^{(2)}$ (mesh) to the analytical expression (solid transparent surface). The fitted surface is the solid opaque surface. $\omega = 2\pi \times 150$ Hz, $\nu_A = 5$ Hz, $\nu_B = 2$ Hz, $\kappa_A = \kappa_B = 3$ Hz Pa$^{-2}$, $\Gamma_A/4\omega^2 = \Gamma_B/4\omega^2 = 1$. The fitted value is $\Gamma_A/4\omega^2 = 1.06$. 
Figure A.12: Calculated probability distribution $W(F, G)$. Black: mean amplitudes. $\omega = 2\pi \times 150$ Hz. $\nu_F = 2$ Hz, $\nu_G = 5$ Hz, $\kappa_F = \kappa_F = 3$ Hz Pa$^{-2}$, $\Gamma_F/4\omega^2 = \Gamma_F/4\omega^2 = 1$. 
Figure A.13: Analytical probability distribution $W(F,G)$. Black: mean amplitudes. $\omega = 2\pi \times 150$ Hz. $\nu_F = 2$ Hz, $\nu_G = 5$ Hz, $\kappa_F = \kappa_G = 3$ Hz Pa$^{-2}$, $\Gamma_F/4\omega^2 = \Gamma_F/4\omega^2 = 1$. 
Figure A.14: Example slice of $W(f,t|\{g\})$ at $\tau = 0.0003$ s showing the ring behaviour. $\omega = 2\pi \times 150$ Hz, $\nu_F = 2$ Hz, $\nu_G = 5$ Hz, $\kappa_F = \kappa_F = 3$ Hz Pa, $\Gamma_F/4\omega^2 = \Gamma_F/4\omega^2 = 1$. 
Figure A.15: Example slice of $W(\{ f, g \}_{t+\tau}, \{ F, G \}_t)$ at $\tau = 0.001$ s showing the ring filling as the distribution diffuses. $\omega = 2\pi \times 150$ Hz. $\nu_F = 2$ Hz, $\nu_G = 5$ Hz, $\kappa_F = \kappa_F = 3$Hz $Pa^{-2}$, $\Gamma_F/4\omega^2 = \Gamma_F/4\omega^2 = 1$. 
Figure A.16: Example slice of $W([f, g]_{t+\tau}, [F, G]_t)$ at $\tau = 0.002$ s. The ring is still visible but is no longer a zero probability zone. $\omega = 2\pi \times 150$ Hz. $\nu_F = 2$ Hz, $\nu_G = 5$ Hz, $\kappa_F = \kappa_F = 3$ Hz Pa$^{-2}$, $\Gamma_F/4\omega^2 = \Gamma_F/4\omega^2 = 1$. 
B

MATLAB Code

B.1 Standing Modes: System Identification

```matlab
% (A, B): STANDING MODES
1 close all, clear all
2 startScript = tic;
3 set (groot, 'defaultTextInterpreter', 'latex');
4 set (groot, 'defaultLegendInterpreter', 'latex');
5 set (groot, 'defaultAxesTickLabelInterpreter', 'latex');

% PARAMETERS
% . . . SIMULATION
11 timeSimulation = 500; % (s) Simulation duration
12 freqSampling = 4000; % (Hz) Sampling frequency
13 periodSampling = 1/freqSampling; % (s) Sampling period
14 freqModal = 150; % (Hz) Modal frequency
15 angfreqModal = 2*pi*freqModal; % (rad/s) Angular modal frequency
16 periodModal = 1/freqModal; % (s) Modal period

% . . . A-PARAMETERS
23 nuA = 5; % (rad/s) Growth rate = (beta - alpha)/2
24 kappaA = 3; % (rad/Pa3s) Non-linear source term
25 gammmaxA = 1; % Normalised noise intensity (/4w2)
26 gammaA = gammmaxA*4*angfreqModal^2; % Noise intensity
27 seedA = floor(1e3*rand(1)); % Random noise seed

% . . . B-PARAMETERS
34 nuB = 5; % (rad/s) Growth rate = (beta - alpha)/2
35 kappaB = 3; % (rad/Pa3s) Non-linear source term
36 gammmaxB = 1; % Normalised noise intensity (/4w2)
37 gammaB = gammmaxB*4*angfreqModal^2; % Noise intensity
38 seedB = floor(1e3*rand(1)); % Random noise seed

% . . . PROCESSING
45 timePlotRange = 2; % (s) Signal plot range
46 timeTransientCutoff = 0.5; % (s) Transient cut-off point
47 freqSearchRange = [30 200]; % (Hz) Range for peak search
48 dA = 0.1; dB = 0.1; % (Pa) Amplitude resolutions
49 dC = 0.1; dD = 0.1; % (C = A^ and D = B^)

% SIMULATION
54 startSimulation = tic;
55 disp('Simulation...
');
56 sim('simABvFINAL');
57 timeRaw = simout.time;
58 etaARaw = simout.signals.values(:, 1);
```

57
etaBraw = simout.signals.values(:, 2);
elapsedSimulation = toc(startSimulation);
disp(['... num2str(elapsedSimulation) ' ' s'])

%%% PROCESSING
startProcessing = tic;
disp('Processing...')

%%% t-DOMAIN

% Process timeCut and etaAcut, etaBcut

timeCut = timeRaw(timeRaw>timeTransientCutoff);
etAcut = etaAraw(timeRaw>timeTransientCutoff);
etBcut = etaBraw(timeRaw>timeTransientCutoff);
nSamples = length(timeCut);

%%% f-DOMAIN

freq = linspace(0, 1, nSamples)*freqSampling;
etaAff = fft(etaAcut)/nSamples;
etaBfft = fft(etaBcut)/nSamples;

% Find peak frequencies and corresponding time points
indPeakSearchLo = find(freq>=freqSearchRange(1), 1, 'first');
indPeakSearchHi = find(freq<=freqSearchRange(2), 1, 'last');

indFftPeakA = find(etaAff(indPeakSearchLo:indPeakSearchHi)) ... = max(etaAff(indPeakSearchLo:indPeakSearchHi));
indFftPeakB = find(etaBfft(indPeakSearchLo:indPeakSearchHi)) ... = max(etaBfft(indPeakSearchLo:indPeakSearchHi));

freqPeakA = freq(indPeakSearchLo + indFftPeakA - 1);
freqPeakB = freq(indPeakSearchLo + indFftPeakB - 1);
angfreqPeakA = 2*pi*freqPeakA;
angfreqPeakB = 2*pi*freqPeakB;

% Process timeProcessed and etaAprocessed, etaBprocessed

timeProcessed = timeCut;
etAProcessed = etaAcut;
etAfftProcessed = etaAff;
freqAProcessed = freq;
etAprocessed = etaBcut;
etBfftProcessed = etaBfft;

%%% AMPLITUDE & PHASE

etaAhilbert = hilbert(etaAprocessed);
etahilbert = hilbert(etaBprocessed);

A = abs(etaAhilbert); meanA = mean(A);
B = abs(etaBhilbert); meanB = mean(B);

phiA = imag(log(etaAhilbert)) - angfreqPeakA*timeProcessed;
phiB = imag(log(etaBhilbert)) - angfreqPeakB*timeProcessed;

phiDifference = phiA - phiB;

C = sqrt(2 + cos(2*phiDifference)).*A; meanC = mean(C); % C = A
D = sqrt(2 + cos(2*phiDifference)).*B; meanD = mean(D); % D = B

elapsedProcess = toc(startProcessing);
disp(['... num2str(elapsedProcess) ' ' s'])

%%% PDFs

startPdfs = tic;
disp('Calculating PDFs...')

%%% PDF(etaA)

HEtaA = floor(min(etaAprocessed)/dA)*dA;
uuEtaA = ceil(max(etaAprocessed)/dA)*dA;
edgesEtaA = [HEtaA (edgesEtaA(1:end-1)+dA/2) uuEtaA];
nEtaA = length(binsEtaA);
countEtaA = zeros(nEtaA, 1);

countEtaA(2:end-1) = histcounts(etaAprocessed, edgesEtaA);

pdfEtaA = countEtaA / trapz(binsEtaA, countEtaA);

% . . . PDF(etaB)

llEtaB = floor(min(etaBprocessed)/dB) * dB;

uuEtaB = ceil(max(etaBprocessed)/dB) * dB;

edgesEtaB = llEtaB:dB:uuEtaB;

binsEtaB = [llEtaB (edgesEtaB(1:end-1)+dB/2) uuEtaB];

nEtaB = length(binsEtaB);

countEtaB = zeros(nEtaB, 1);

countEtaB(2:end-1) = histcounts(etaBprocessed, edgesEtaB);

pdfEtaB = countEtaB / trapz(binsEtaB, countEtaB);

% . . . PDF(A)

uuA = ceil(max(A)/dA) * dA;

edgesA = 0:dA:uuA;

binsA = [0 (edgesA(1:end-1)+dA/2) uuA];

nA = length(binsA);

countA = zeros(nA, 1);

countA(2:end-1) = histcounts(A, edgesA);

pdfA = countA / trapz(binsA, countA);

% . . . PDF(B)

uuB = ceil(max(B)/dB) * dB;

edgesB = 0:dB:uuB;

binsB = [0 (edgesB(1:end-1)+dB/2) uuB];

nB = length(binsB);

countB = zeros(nB, 1);

countB(2:end-1) = histcounts(B, edgesB);

pdfB = countB / trapz(binsB, countB);

% . . . PDF(C)

uuC = ceil(max(C)/dC) * dC;

edgesC = 0:dC:uuC;

binsC = [0 (edgesC(1:end-1)+dC/2) uuC];

nC = length(binsC);

countC = zeros(nC, 1);

countC(2:end-1) = histcounts(C, edgesC);

pdfC = countC / trapz(binsC, countC);

% . . . PDF(D)

uuD = ceil(max(D)/dD) * dD;

edgesD = 0:dD:uuD;

binsD = [0 (edgesD(1:end-1)+dD/2) uuD];

nD = length(binsD);

countD = zeros(nD, 1);

countD(2:end-1) = histcounts(D, edgesD);

pdfD = countD / trapz(binsD, countD);

% . . . PDF(A, B)

countAB = zeros(nA, nB);

countAB(2:end-1, 2:end-1) = histcounts2(A, B, edgesA, edgesB);

pdfAB = countAB / trapz(binsA, trapz(binsB, countAB, 2), 1);

% . . . PDF(A, D)

countAD = zeros(nA, nD);

countAD(2:end-1, 2:end-1) = histcounts2(A, D, edgesA, edgesD);

pdfAD = countAD / trapz(binsA, trapz(binsD, countAD, 2), 1);

% . . . PDF(C, B)

countCB = zeros(nC, nB);

countCB(2:end-1, 2:end-1) = histcounts2(C, B, edgesC, edgesB);

pdfCB = countCB / trapz(binsC, trapz(binsB, countCB, 2), 1);

% . . . PDF(C, D)

countCD = zeros(nC, nD);

countCD(2:end-1, 2:end-1) = histcounts2(C, D, edgesC, edgesD);

pdfCD = countCD / trapz(binsC, trapz(binsD, countCD, 2), 1);
countCB = zeros(nC, nB);
countCB(2:end-1, 2:end-1) = histcounts2(C, B, edgesC, edgesB);
jpdfCB = countCB/trapz(binsC, trapz(binsB, countCB, 2), 1);
elapsedPdfs = toc(startPdfs);
disp(['  elapsedPdfs ' num2str(elapsedPdfs) ' s'])

% % SYSTEM ID

startSystemId = tic;
disp('System ID...')

nShift = 20;

nTau = 50;

indShift = linspace(nShift, nShift+nTau, nTau);
tau = indShift*periodSampling;

% % PDFs

countADAD = zeros(nTau, nA, nD, nA, nD);

for i = 1:nTau
    iA = A(1:(nSamples - indShift(i))); % Resize A to same size as a
    iD = D(1:(nSamples - indShift(i))); % ( no zero-padding)

    % Calculate joint counts of (a, d) cycling over (A, D) = (jA, kD)
    for j = 1:(nA - 2)
        for k = 1:(nD - 2)
            ind = find(((edgesA(j) <= iA) & (iA < edgesA(j+1))) & ...
                ((edgesD(k) <= iD) & (iD < edgesD(k+1))));
            AAD = A(ind + indShift(i));
            DAD = D(ind + indShift(i));
            countADAD(i, 2:end-1, 2:end-1, j+1, k+1) = ...
                histcounts2(AAD, DAD, edgesA, edgesD);
        end
    end
end

jpdfADAD = zeros(nTau, nA, nD, nA, nD);

for i = 1:nTau
        / trapz(binsA, ...
        trapz(binsD, ...
        trapz(binsA, ...
        trapz(binsD, countADAD(i, : , : , : , 5), 4), 3), 2);
end
jpdfADAD(jpdfADAD < 1e-2) = 0; % Remove outliers

cpdfADAD = zeros(nTau, nA, nD, nA, nD);

for i = 1:nTau
    for j = 1:nA
        for k = 1:nD
            for l = 1:nA
                for m = 1:nD
                    cpdfADAD(i, j, k, l, m) = ...
                    jpdfADAD(i, j, k, l, m)/jpdfADAD(l, m);
                end
            end
        end
    end
end

cpdfADAD(isnan(cpdfADAD)) = 0; % Because of /0

% % (C, B) = (A*, B)

countCBCB = zeros(nTau, nC, nB, nC, nB);

for i = 1:nTau
    iC = C(1:(nSamples - indShift(i)));
    iB = B(1:(nSamples - indShift(i)));
end
for j = 1:(nC - 2)
    for k = 1:(nB - 2)
        ind = find(((edgesC(j) <= iC) & (iC < edgesC(j+1))) & ... ((edgesB(k) <= iB) & (iB < edgesB(k+1))));
        CCB = C(ind + indShift(i));
        BCB = B(ind + indShift(i));
        countCBCB(i, 2:end-1, 2:end-1, j+1, k+1) = ...
        histcounts2(CCB, BCB, edgesC, edgesB);
    end
end

jpdfCBCB = zeros(nTau, nC, nB, nC, nB);
for i = 1:nTau
    jpdfCBCB(i, j, k, l, m) = ... 
    /trapz(binsC, ... 
    trapz(binsB, ... 
    trapz(binsC, ... 
    jpdfCBCB(i, j, k, l, m)/jpdfCB(l, m);
end

jpdfCBCB(i-pdfCBCB < 1e-2) = 0;

cpdfCBCB = zeros(nTau, nC, nB, nC, nB);
for i = 1:nTau
    for j = 1:nC
        for k = 1:nB
            for l = 1:nC
                for m = 1:nB
                    cpdfCBCB(i, j, k, l, m) = ...
                    jpdfCBCB(i, j, k, l, m)/jpdfCB(l, m);
                    end
                end
            end
        end
    end
end

cpdfCBCB(isnan(cpdfCBCB)) = 0;

integrandDA = zeros(nTau, nA, nD, nA, nD); % Drift integrands
integrandDB = zeros(nTau, nC, nB, nC, nB);
integrandDAA = zeros(nTau, nA, nD, nA, nD); % Diffusion integrands
integrandDBB = zeros(nTau, nC, nB, nC, nB);
for i = 1:nTau
    for j = 1:nA
        for k = 1:nA
            integrandDA(i, j, k) = (binsA(j) - binsA(k))* ... 
            cpdfDAD(i, j, k) / tau(i);
        end
    end
end

for j = 1:nB
    for k = 1:nB
        integrandDB(i, j, k) = (binsB(j) - binsB(k))* ... 
        cpdfDJB(i, j, k) / tau(i);
    end
end

for j = 1:nA
    for k = 1:nA
        integrandDAA(i, j, k) = ((binsA(j) - binsA(k))^2)* ... 
        cpdfDAD(i, j, k) / (2*tau(i));
    end
end

for j = 1:nB
    for k = 1:nB
        integrandDBB(i, j, k) = ((binsB(j) - binsB(k))^2)* ... 
        cpdfDJB(i, j, k) / (2*tau(i));
    end
end

DA = zeros(nTau, nA, nD); % Drift coefficients
DB = zeros(nTau, nC, nB);
DAA = zeros(nTau, nA, nD); % Diffusion coefficients
DBB = zeros(nTau, nC, nB);
for i = 1:nTau
    for j = 1:nA
        for k = 1:nD
            DA(i, j, k) = trapz(binsA, ... 
            trapz(binsD, integrandDA(i, j, k), 3), 2);
            end
        end
    end
end

DAA(i, j, k) = trapz(binsA, ...
351 \text{trapz}(\text{binsD}, \text{integrandDAA}(i, :, :, j, k), 3), 2); \\
352 \text{end} \\
353 \text{end} \\
354 \text{for } j = 1 : nC \\
355 \text{for } k = 1 : nB \\
356 DB(i, j, k) = \text{trapz}(\text{binsC}, \ldots \\
357 \text{trapz}(\text{binsB}, \text{integrandDB}(i, :, :, j, k), 3), 2); \\
358 \text{DBB}(i, j, k) = \text{trapz}(\text{binsC}, \ldots \\
359 \text{trapz}(\text{binsB}, \text{integrandDBB}(i, :, :, j, k), 3), 2); \\
360 \text{end} \\
361 \text{end} \\
362 \text{end} \\
363 \text{elapsedSystemId} = \text{toc}(\text{startSystemId}); \\
364 \text{disp(['\ldots ' num2str(elapsedSystemId) ' s'])} \\
365 \\
366 \%\% FITTING \\
367 \\
368 \text{startFitting} = \text{tic}; \\
369 \text{disp('Fitting\ldots')} \\
370 \%\ldots v. TAU \\
371 \text{indTau1} = \text{round}(nTau/10); \\
372 \text{tau1} = \text{tau(indTau1)}; \% \text{<<<<<<<<<<<<<<} \\
373 \text{indTau2} = nTau - \text{round}(nTau/10); \\
374 \text{tau2} = \text{tau(indTau2)}; \% \text{<<<<<<<<<<<<<<} \\
375 \text{DAtfitTauCoeffs} = \text{zeros}(nA, nD, 2); \text{DAtfitTau0} = \text{zeros}(nA, nD); \\
376 \text{DAfitTauCoeffs} = \text{zeros}(nA, nD, 2); \text{DAfitTau0} = \text{zeros}(nA, nD); \\
377 \text{for } i = 1 : nA \\
378 \text{for } j = 1 : nD \\
379 p = \text{fit}(\text{tau(indTau1:indTau2)}, \text{DA(indTau1:indTau2, i, j)}, '\text{exp1}'); \\
380 q = \text{fit}(\text{tau(indTau1:indTau2)}, \text{DAA(indTau1:indTau2, i, j)}, '\text{exp1}'); \\
381 pCoeffs = \text{coeffvalues}(p); \\
382 qCoeffs = \text{coeffvalues}(q); \\
383 \text{for } k = 1 : 2 \\
384 \text{DAfitTauCoeffs}(i, j, k) = pCoeffs(k); \\
385 \text{DAfitTauCoeffs(i, j, k) = qCoeffs(k);} \\
386 \text{end} \\
387 \text{DAfitTau0(i, j) = pCoeffs(1);} \\
388 \text{DAfitTau0(i, j) = qCoeffs(1);} \\
389 \text{end} \\
390 \text{end} \\
391 \text{end} \\
392 \text{DAtfitTau} = \text{zeros}(nTau+1, nA, nD); \\
393 \text{DAfitTau} = \text{zeros}(nTau+1, nA, nD); \\
394 \text{for } i = 1 : nA \\
395 \text{for } j = 1 : nD \\
396 \text{DAtfitTau(:, i, j) = DAfitTauCoeffs(i, j, 1)* \ldots \\
397 \text{exp}[\text{DAfitTauCoeffs(i, j, 2)*[0 tau]}]; \\
398 \text{DAfitTau(:, i, j) = DAfitTauCoeffs(i, j, 1)* \ldots \\
399 \text{exp}[\text{DAfitTauCoeffs(i, j, 2)*[0 tau]}]; \\
400 \text{end} \\
401 \text{end} \\
402 \text{DBfitTauCoeffs} = \text{zeros}(nC, nB, 2); \text{DBfitTau0} = \text{zeros}(nC, nB); \\
403 \text{DBfitTauCoeffs} = \text{zeros}(nC, nB, 2); \text{DBfitTau0} = \text{zeros}(nC, nB); \\
404 \text{for } i = 1 : nC \\
405 \text{for } j = 1 : nB \\
406 p = \text{fit}(\text{tau(indTau1:indTau2)}, \text{DB(indTau1:indTau2, i, j)}, '\text{exp1}'); \\
407 q = \text{fit}(\text{tau(indTau1:indTau2)}, \text{DBB(indTau1:indTau2, i, j)}, '\text{exp1}'); \\
408 pCoeffs = \text{coeffvalues}(p); \\
409 qCoeffs = \text{coeffvalues}(q); \\
410 \text{for } k = 1 : 2 \\
411 \text{DBfitTauCoeffs(i, j, k) = pCoeffs(k);} \\
412 \text{DBfitTauCoeffs(i, j, k) = qCoeffs(k);} \\
413 \text{end} \\
414 \text{DBfitTau0(i, j) = pCoeffs(1);} \\
415 \text{DBfitTau0(i, j) = qCoeffs(1);} \\
416 \text{end} \\
417 \text{end}
for i = 1 : nC
    for j = 1 : nB
        DBfitTau (: , i , j) = DBfitTauCoeffs ( i , j , 1) * ... 
        exp ( DBfitTauCoeffs ( i , j , 2) * [0 tau] ) ;
        DBBfitTau (: , i , j) = DBBfitTauCoeffs ( i , j , 1) * ... 
        exp ( DBBfitTauCoeffs ( i , j , 2) * [0 tau] ) ;
    end
end

% % . . . v . (A, D)

indAfit = ( pdfA >= 0.1 * max( pdfA ) ) ; Afit = binsA ( indAfit ) ;
%maxpdf = 0.1

indDfit = ( pdfD >= 0.1 * max( pdfD ) ) ; Dfit = binsD ( indDfit ) ;

[ AfitGrid , DfitGrid ] = meshgrid ( Afit , Dfit ) ;

[ x , y , z ] = ... 
prepareSurfaceData ( AfitGrid , DfitGrid , DAAfitTau0 ( indAfit , indDfit ) .') ;
DAAfitADfunction = fittypedef (@ ( gammaxAfit , x , y ) ... 
    gammaxAfit * ( x .^ 0 ) .* ( y .^ 0 ) , ... 
    'coefficients' , { 'gammaMaxAfit' } , ... 
    'independent' , { x , y } ) ;
DAAfitAD = fit ( [ x , y ] , z , DAAfitADfunction ) ;
DAAfitADcoeffs = coeffvalues ( DAAfitAD ) ;

gammaMaxAfit = DAAfitADcoeffs (1) ; disp ( [ 'gammaMaxAfit = ' num2str ( gammaMaxAfit ) ] )

% . . . v . (B, C)

indCfit = ( pdfC >= maxpdf * max( pdfC ) ) ; Cfit = binsC ( indCfit ) ;

indDfit = ( pdfD >= maxpdf * max( pdfD ) ) ; Bfit = binsD ( indBfit ) ;

[CfitGrid, BfitGrid] = meshgrid(Cfit, Bfit);

[x, y, z] = ... 
prepareSurfaceData(CfitGrid, BfitGrid, DBBfitTau0(indCfit, indBfit).') ;
DBBfitCBfunction = fittypedef (@ ( nuBfit, kappaBfit, x, y ) ... 
    nuBfit * y - (3 * kappaBfit / 32) * (3 * y .^ 2 + x .^ 2) .* y + gammaMaxBfit ./ y , ... 
    'coefficients', [ 'nuBfit', 'kappaBfit' ] , ... 
    'independent', { x , y } , ... 
    'dependent', { z } ) ;
DBBfitCB = fit ([ x , y ] , z , DBBfitCBfunction) ;
DBBfitCBcoeffs = coeffvalues(DBBfitCB) ;
gammaMaxBfit = DBBfitCBcoeffs (1) ; disp ( [ 'gammaMaxBfit = ' num2str ( gammaMaxBfit ) ] )
elapsedFitting = toc(startFitting);
disp([... num2str(elapsedFitting) ' s']);

% % ANALYTICAL

coupledDeterministicA = 2*sqrt((2*nuA)/(3*kappaA));
coupledDeterministicB = 2*sqrt((2*nuB)/(3*kappaB));

driftAfunction = @(nu, kappa, gammax, a, d) ... 
nu*a - (3*kappa/32)*(3*a.'*2 + d.*2).*a + gammax./a;
driftBfunction = @(nu, kappa, gammax, c, b) ... 
nu*b - (3*kappa/32)*(3*b.*2 + c.*2).*b + gammax./b;

[binGridA, binGridD] = meshgrid(binsA, binsD);
[binGridB, binGridC] = meshgrid(binsB, binsC);

DAanalytical = driftAfunction(nuA, kappaA, gammaxA, binGridA, binGridD);
DBanalytical = driftBfunction(nuB, kappaB, gammaxB, binGridC, binGridB);

% % PLOTTING

figSignal = figure;

subplot(2, 1, 1), hold on, grid on
plot(timeRaw(timeRaw<timePlotRange), etaAraw(timeRaw<timePlotRange), 'b--')
plot(timeCut(timeCut<timePlotRange), etaAcut(timeCut<timePlotRange), 'b-')
plot(timeProcessed(timeProcessed<timePlotRange), A(timeProcessed<timePlotRange), 'k-')
xlabel('$t$ (s)')
ylabel('$\eta_A$')

legend({'Transient'; 'Full'; 'Hilbert'});

subplot(2, 1, 2), hold on, grid on
plot(timeRaw(timeRaw<timePlotRange), etaBraw(timeRaw<timePlotRange), 'b--')
plot(timeCut(timeCut<timePlotRange), etaBcut(timeCut<timePlotRange), 'b-')
plot(timeProcessed(timeProcessed<timePlotRange), B(timeProcessed<timePlotRange), 'k-')
xlabel('$f$ (Hz)')
ylabel('FFT($\eta_B$)')

% % FFT

Nyquist–Shannon: max resolvable frequency = freqSampling/2

figFFT = figure;

subplot(2, 1, 1), hold on, grid on
set(gca, 'yscale', 'log')
plot(freq(1:round(nSamples/2)), abs(etaAfft(1:round(nSamples/2))), 'b-')
plot([1 1]*freqPeakA, get(gca, ylim), 'k-')
xlabel('f (Hz)')
ylabel('FFT($\eta_A$)')

legend({'FFT'; '$f_{Peak}$'});

subplot(2, 1, 2), hold on, grid on
set(gca, 'yscale', 'log')
plot(freq(1:round(nSamples/2)), abs(etaBfft(1:round(nSamples/2))), 'b-')
plot([1 1]*freqPeakB, get(gca, ylim), 'k-')

% % % AMPLITUDE & PHASE

figAmplitude = figure;

subplot(2, 2, 1), hold on, grid on
set(gca, 'yscale', 'log')
plot(freq(1:round(nSamples/2)), abs(etaAamplitude(1:round(nSamples/2))), 'b-')
plot([1 1]*freqPeakA, get(gca, ylim), 'k-')
xlabel('f (Hz)')
ylabel('FFT($\eta_A$)')

subplot(2, 2, 2), hold on, grid on
set(gca, 'yscale', 'log')
plot(freq(1:round(nSamples/2)), abs(etaBamplitude(1:round(nSamples/2))), 'b-')
plot([1 1]*freqPeakB, get(gca, ylim), 'k-')
xlabel('f (Hz)')
ylabel('FFT($\eta_B$)')
plot(timeProcessed(timeProcessed<timePlotRange), ...) B(timeProcessed(timeProcessed<timePlotRange), 'k-')
xlabel('$t$ (s)')
ylabel('$B$')

subplot(2, 2, 3), hold on, grid on plot(timeProcessed(timeProcessed<timePlotRange), ...) C(timeProcessed(timeProcessed<timePlotRange), 'k-')
xlabel('$t$ (s)')
ylabel('$\hat{A}$')

subplot(2, 2, 4), hold on, grid on plot(timeProcessed(timeProcessed<timePlotRange), ...) D(timeProcessed(timeProcessed<timePlotRange), 'k-')
xlabel('$t$ (s)')
ylabel('$\hat{B}$')

figPhase = figure;

subplot(3, 1, 1) , hold on, grid on wrapTo2Pi(phiA(timeProcessed<timePlotRange)) , 'k-' )
xlabel('$t$ (s)')
ylabel('$\phi_A$')

subplot(3, 1, 2) , hold on, grid on wrapTo2Pi(phiB(timeProcessed<timePlotRange)) , 'k-' )
xlabel('$t$ (s)')
ylabel('$\phi_B$')

subplot(3, 1, 3) , hold on, grid on wrapTo2Pi(phiDifference(timeProcessed<timePlotRange)) , 'k-' )
xlabel('$t$ (s)')
ylabel('Phi = $\phi_A - \phi_B$')

% % . . . PDFs

figPDFetaAandA = figure; hold on, grid on plot(binsEtaA, pdfEtaA, 'or-')
plot(binsA, pdfA, 'ob-')
plot([1 1]*meanA, get(gca, 'ylim'), 'b-')
plot([1 1]*coupledDeterministicA, get(gca, 'ylim'), 'k-')
xlabel('$\eta_A$')
ylabel('PDF')
legend('{\$\eta_A$}', '{\$A$}', '{\$\bar{A}$}', '{\$A_{\text{det}}$}', 'location', 'northwest')

figPDFetaBandB = figure; hold on, grid on, box on plot(binsEtaB, pdfEtaB, 'or-')
plot(binsB, pdfB, 'ob-')
plot([1 1]*meanB, get(gca, 'ylim'), 'b-')
plot([1 1]*coupledDeterministicB, get(gca, 'ylim'), 'k-')
xlabel('$\eta_B$')
ylabel('PDF')
legend('{\$\eta_B$}', '{\$B$}', '{\$\bar{B}$}', '{\$B_{\text{det}}$}', 'location', 'northwest')

figPDFAB = figure; hold on, grid on, box on contourf(binsA, binsB, jpdfAB, colormap(flipud(hot)))
plot([1 1]*meanA, get(gca, 'ylim'), 'b-')
plot(get(gca, 'xlim'), [1 1]*meanB, 'b-')
plot([1 1]*coupledDeterministicA, get(gca, 'ylim'), 'k-')
plot(get(gca, 'xlim'), [1 1]*coupledDeterministicB, 'k-')
xlabel('$A$')
ylabel('$B$')

figPDFC = figure; hold on, grid on, box on plot(binsC, pdfC, 'ob-')

65
plot([1 1]*meanC, get(gca, 'ylim'), 'b--')
xlabel('$$\hat{A}$$')
ylabel('PDF')

figPDFD = figure; hold on, grid on, box on
plot(binsD, pdfD, 'ob--')
xlabel('$$\hat{B}$$')
ylabel('PDF')

figPDFAD = figure; hold on, grid on, box on
contourf(binsA, binsD, jpdfAD.')
colormap(flipud(hot))
plot([1 1]*meanA, get(gca, 'ylim'), 'b--')
plot(get(gca, 'xlim'), [1 1]*meanD, 'b--')
xlabel('$$A$$')
ylabel('$$\hat{B}$$')

figPDFCB = figure; hold on, grid on, box on
contourf(binsC, binsB, jpdfCB.')
colormap(flipud(hot))
plot([1 1]*meanC, get(gca, 'ylim'), 'b--')
plot(get(gca, 'xlim'), [1 1]*meanB, 'b--')
xlabel('$$\hat{A}$$')
ylabel('$$B$$')

% % ... SYSTEM ID

indTauPlot = 3; tauPlot = tau(indTauPlot);
dmAplot = round(nA/2); Aplot = binsA(indmAplot);
dmCplot = round(nC/2); Cplot = binsC(indmCplot);
dmDplot = round(nD/2); Dplot = binsD(indmDplot);

figJpdfADAD = figure; hold on, grid on, box on
contourf(binsA, binsD, squeeze(jpdfADAD(indTauPlot, :, :, indmAplot, indmDplot))'.')
colormap(flipud(hot))
xlabel('$$a$$')
ylabel('$$\hat{b}$$')
title([' $$JPDF( a , \hat{b} , A = $$' num2str(Aplot) ' , B = $$' num2str(Dplot) ' @ (\tau = $$' num2str(tauPlot) ')$]')

figJpdfCBCB = figure; hold on, grid on, box on
contourf(binsC, binsB, squeeze(jpdfCBCB(indTauPlot, :, :, indmCplot, indmBplot))'.')
colormap(flipud(hot))
xlabel('$$\hat{a}$$')
ylabel('$$b$$')
title([' $$JPDF( \hat{a} , b , A = $$' num2str(Cplot) ' , B = $$' num2str(Bplot) ' @ (\tau = $$' num2str(tauPlot) ')$]')

figCPdfADAD = figure; hold on, grid on, box on
contourf(binsA, binsD, squeeze(cpdfADAD(indTauPlot, :, :, indmAplot, indmDplot))'.')
colormap(flipud(hot))
xlabel('$$\hat{a}$$')
ylabel('$$\hat{b}$$')
title([' $$CPDF( a , \hat{b} | A = $$' num2str(Aplot) ' , B = $$' num2str(Bplot) ' @ (\tau = $$' num2str(tauPlot) ')$]')

figCPdfCBCB = figure; hold on, grid on, box on
contourf(binsC, binsB, squeeze(cpdfCBCB(indTauPlot, :, :, indmCplot, indmBplot))'.')
colormap(flipud(hot))
\begin{verbatim}
xlabel('\$\tau \hat{A}\$')
ylabel('SD' '(2) \{B, \tau\} (\hat{A}, B)$')
leg = ['\{ Calculated \}', '\{ Exponential Fit \}']

\&
\end{verbatim}
B.2 Standing Modes: AFPE Solver

function afpeSolve3D
% AFFESOLVE3D Solve AFPE in (a, b, phi)

% PARAMETERS:
nu = 5; kappa = 3; gammmax = 1;

% Domain limits: x = A, y = B, z = Phi
xMin = 0; xMax = 10;
yMin = 0; yMax = 10;
zMin = 0; zMax = 2*pi;

m = 0.5; % Maximum mesh element size
tList = 0:0.001:0.1; % Times to solve at

% Langevin Equations:
da/dt = nuA - (3kappa/32)(3A^2 + (2 + cos2phi)B^2)A + gammmax/(4wn^2A) + zetaA
db/dt = nuB - (3kappa/32)(3B^2 + (2 + cos2phi)A^2)B + gammmax/(4wn^2B) + zetaB
dphi/dt = (3kappa/32)(A^2 + B^2) sin2phi + (1/A + 1/B) zetaPhi

% Kramers–Moyal Coefficients:
D1A = nuA - (3kappa/32)(3A^2 + (2 + cos2phi)B^2)A
D1B = nuB - (3kappa/32)(3B^2 + (2 + cos2phi)A^2)B
D1P = (3kappa/32)(A^2 + B^2) sin2phi

D2AA = gammmax
D2BB = gammmax
D2PP = gammmax/(1/A + 1/B)^2

% Adjoint Fokker–Planck Equation:

% PDE Toolbox solves scalar equations of the form:
m[d2/dt2]u + d[d/dt]u + div(cgrad(u)) + au = f

% Adjoint Fokker–Planck Equation:

% PDE Toolbox solves scalar equations of the form:
m[d2/dt2]u + d[d/dt]u + div(cgrad(u)) + au = f

% % % %
tList = 0:0.05:0.1;
model = createpde();

% GEOMETRY:
[xg, yg, zg] = meshgrid(xMin:xMax, yMin:yMax, zMin:zMax);
xg = xg(:); yg = yg(:); zg = zg(:);
shp = alphaShape(xg, yg, zg);
[elements, nodes] = boundaryFacets(shp);
nodes = nodes';
elements = elements';
geometryFromMesh(model, nodes, elements);

degplot(model, 'FaceLabels', 'on', 'FaceAlpha', 0.5)
axis equal, axis([xMin xMax yMin yMax zMin zMax])
drawnow

% BCS: None

% PDE COEFFICIENTS:
m = 0;
d = 1;
a = 0;

function cmatrix = ccoeffunction(region, ~)
nl = 3;
nr = numel(region.x);
cmatrix = zeros(nl,nr);
cmatrix(1,:) = gammax*ones(1,nr);
cmatrix(2,:) = gammax*ones(1,nr);
cmatrix(3,:) = gammaz*(region.x/region.x);.
end
c = @ccoeffunction;

function f = fcoefficient(region, state)
f = (nu*region.x - ...)
  (3*kappa/32)*(3*region.x-.2 + (2*cos(2*region.z)).*region.y-.2).*region.x + ...
  (gammax/region.x).*state.ux + ...
  + (nu*region.y - ...)
  (3*kappa/32)*(3*region.y-.2 + (2*cos(2*region.z)).*region.x-.2).*region.y + ...
  (gammaz/region.y).*state.uy + ...
  + (3*kappa/32)*(region.x-.2 + region.y-.2).*sin(2*region.z)*state.uz ;
end
f = @fcoefficient;
specifyCoefficients(model, 'm', m, ... 'd', d, ... 'c', c, ... 'a', a, ... 'f', f);

% ICS:

% INITIAL CONDITION: p(a, 0) = (a - A)^n or p(b, 0) = (b - B)^n
% AB = 'B'; % Either 'A' for D_A(n) or 'B' for D_B(n)
n = 1;
xy0 = 1; % Value of A or B in (a - A)^n or (b - B)^n

function u0 = initfun(locations)
u0 = (locations.y - xy0).^n;
end
u0 = @initfun;

setInitialConditions(model, u0);

% MESH:
generateMesh(model, 'Hmax', sMax)
B.3 Rotating Modes: System Identification

% (F, G)

close all, clear all

startScript = tic;

set(groot, 'defaultTextInterpreter', 'latex')
set(groot, 'defaultLegendInterpreter', 'latex')
set(groot, 'defaultAxesTickLabelInterpreter', 'latex')

% PARAMETERs

% SIMULATION

timeSimulation = 500; % (s) Simulation duration
freqSampling = 3000; % (Hz) Sampling frequency, multiple of fN
periodSampling = 1/freqSampling; % (s) Sampling period

freqModal = 150; % (Hz) Modal frequency
angfreqModal = 2*pi*freqModal; % (rad/s) Angular modal frequency
periodModal = 1/freqModal; % (s) Modal period

% F-PARAMETERS

nuF = 2; % (rad/s) Growth rate = (beta - alpha)/2
kappaF = 3; % (rad/Pa3s) Non-linear source term
gammaFx = 1; gammaF = 2*gammaFx; % Normalised noise intensity (/4w2)
gammaF = gammaFx^4*angfreqModal^2; % Noise intensity
seedF = floor(1e3*rand(1)); % Random noise seed
f0 = 1; % Initial value

% G-PARAMETERS

nuG = 5;
kappaG = 3; % (rad/Pa3s) Non-linear source term
gammaG = 1; gammmaxG = 2*gammaG; % Normalised noise intensity (/4w2)
gammaG = gammmaxG+4*angfreqModal^2; % Noise intensity
seedG = floor(1e3*rand(1));
g0 = 1;

% PROCESSING

timePlotRange = 2; % (s) Signal plot range
timeTransientCutoff = 0.5; % (s) Transient cut-off
indDownsample = floor(periodModal/periodSampling); % Downsampling index: one sample per tN
dF = 0.05; dG = 0.05; % (Pa) Amplitude resolutions
%% SIMULATION
startSimulation = tic;
disp('Simulation ... ')
sim simFGvFINAL

timeRaw = simout.time;
fraw = simout.signals.values(:, 1);
graw = simout.signals.values(:, 2);

% Remove extremes to ease binning of signal
fraw(fraw > 10*mean(fraw)) = 0;
graw(graw > 10*mean(graw)) = 0;

ebinedSimulation = toc(startSimulation);
disp(['...' num2str(ebinedSimulation) ' s'])

%% PROCESSING
startProcessing = tic;
disp('Processing ... ')
timeCut = timeRaw(timeRaw > timeTransientCutoff); % Cut transient signal
fCut = fraw(timeCut > timeTransientCutoff);
gCut = graw(timeCut > timeTransientCutoff);
timeDownsampled = cell(indDownsample, 1);
fDownsampled = cell(indDownsample, 1);
gDownsampled = cell(indDownsample, 1);
nSamples = cell(indDownsample, 1);

% Downsample - amplitude varies slowly wrt to oscillation
for i = 1:indDownsample
    timeDownsampled{i} = timeCut(i:indDownsample:end);
fDownsampled{i} = fCut(i:indDownsample:end);
gDownsampled{i} = gCut(i:indDownsample:end);
nSamples{i} = length(timeDownsampled{i});
end

ebinedProcessing = toc(startProcessing);
disp(['...' num2str(ebinedProcessing) ' s'])

%% PDFs
startPdfs = tic;
disp('Calculating PDFs ... ')

% Store PDF in cell for each possible starting point of downsampling in oscillation period
pdfFstore = cell(indDownsample, 1);

if F = 0; % Lower limit of IF - F must be +ve
    uuF = ceil(max(fCut)/dF)*dF; % Upper limit of IF (round up)
    edgesF = [1F (edgesF(end-1)+dF)/2] uuF;
    nF = length(edgesF);
end

parfor i = 1:indDownsample
    if = fDownsampled{i};
    countF = zeros(nF, 1);
    countF(2:end-1) = histcounts(if, edgesF);
    pdfFstore{i} = countF/trapz(binsF, countF);
end

dfFavg = zeros(size(pdfFstore{1}));
for i = 1:indDownsample
    pdfFavg = pdfFavg + pdfFstore{i};
end

72
pdfFavg = pdfFavg/indDownsample;

% . . . PDF(G)
11G = 0;
uwG = ceil(max(gCut)/dG)*dG;
edgesG = [11G (edgesG(1:end-1)+dF/2) uwG];
G = length(edgesG);
parfor i = 1:indDownsample
    iG = gDownsampled{i};
    countG = zeros(G, 1);
    countG(2:end-1) = histcounts(iG, edgesG);
    pdfGstore{i} = countG/ trapz(edgesG, countG);
end
pdfGavg = zeros(size(pdfGstore{1}));
for i = 1:indDownsample
    pdfGavg = pdfGavg + pdfGstore{i};
end
pdfGavg = pdfGavg/indDownsample;

% . . . PDF(F, G)
jpdfFGstore = cell(indDownsample, 1);
parfor i = 1:indDownsample
    iF = fDownsampled{i}; iG = gDownsampled{i};
    countFG = zeros(F, G);
    countFG(2:end-1, 2:end-1) = histcounts2(iF, iG, edgesF, edgesG);
    jpdfFGstore{i} = countFG/ trapz(edgesF, trapz(edgesG, countFG, 2), 1);
end
jpdfFGavg = zeros(size(jpdfFGstore{1}));
for i = 1:indDownsample
    jpdfFGavg = jpdfFGavg + jpdfFGstore{i};
end
jpdfFGavg = jpdfFGavg/indDownsample;

elapsedPdfs = tic (startPdfs);
disp([" . . . " num2str(elapsedPdfs) " s "])

% % SYSTEM ID
startSystemId = tic;
disp("System ID . . .")

nShift = 1;  % # of points to shift
nTau = 10;   % # of tau values to use
indShift = linspace(nShift, nShift+nTau, nTau);  % Shift indices
tau = indShift*periodSampling;  % Values of tau (s)

% . . . PDFs
jpdfFGFGstore = cell(indDownsample, 1);  % [\{f, g\}(t + tau), \{F, G\}(t)]:
jpdfFGFGstore = cell(indDownsample, 1);

jpdfFGFGavg = zeros(F, G, nF, nG);  % Temp. joint counts
for j = 1:nTau
    jF = iF(1:(inS - indShift{j}));
    jG = iG(1:(inS - indShift{j}));
    for k = 1:(nF - 2)  % - 2 because size(fEdg) + 1 = nF
        for l = 1:(nG - 2)
            % Calculate joint counts of {f, g} cycling over {F, G} = \{kF, kG\}
ind = find((edgesF(k) <= jF) & (jF < edgesF(k+1))) & ... 
((edgesG(1) <= jG) & (jG < edgesG(1+1)));

FFG = iF(ind + indShift(j));
GFG = iG(ind + indShift(j));
countFGFG(j, 2:end-1, 2:end-1, k+1, l+1) = ... 
histcounts2(FFG, GFG, edgesF, edgesG);
end
end
end

jpdfFGFG = zeros(nTau, nF, nG, nF, nG); % Temp. joint PDF
for j = 1:nTau
  jpdfFGFG(j, :, :, :, :) = countFGFG(j, :, :, :, :) ... 
  \trapz(binsF, ... 
  \trapz(binsG, ... 
  \trapz(binsF, ... 
  \trapz(binsG, countFGFG(j, :, :, :, :), 5), 4), 3), 2);
end

%fgFGjpdf(fgFGjpdf < 1e-3) = 0; % Remove values close to zero
jpdfFGFG(jpdfFGFG(isinf(jpdfFGFG)) = 0;
jpdfFGFGstore{i} = jpdfFGFG;
end

jpdfFGFGavg = zeros(size(jpdfFGFGstore{1}));
for i = 1:indDownsample
  jpdfFGFGavg = jpdfFGFGavg + jpdfFGFGstore{i};
end
jpdfFGFGavg = jpdfFGFGavg/indDownsample;
cpdfFGFG = zeros(nTau, nF, nG, nF, nG); % CPDF( f, g | F, G)
for i = 1:nTau
  for j = 1:nF
    for k = 1:nF
      integrandDF(i, j, : , k, :) = (binsF(j) - binsF(k))* ... 
      cpdfFGFG(i, j, : , k, :) / tau(i);
      integrandDFF(i, j, : , k, :) = ((binsF(j) - binsF(k))^2)* ... 
      cpdfFGFG(i, j, : , k, :) / (2*tau(i));
    end
  end
end
for j = 1:nG
  for k = 1:nG
    integrandDG(i, :, j, k) = (binsG(j) - binsG(k))* ... 
    cpdfFGFG(i, :, j, k) / tau(i);
    integrandDGG(i, :, j, k) = ((binsG(j) - binsG(k))^2)* ... 
    cpdfFGFG(i, :, j, k) / (2*tau(i));
  end
end
DF = zeros(nTau, nF, nG); % Drift coefficients
DG = zeros(nTau, nF, nG);
for i = 1:nTau
    for j = 1:nF
        for k = 1:nG
            DF(i, j, k) = trapz(binsF, ...
            trapz(binsG, integrandDF(i, :, :, j, k), 3), 2);
            DFF(i, j, k) = trapz(binsF, ...
            trapz(binsG, integrandDFF(i, :, :, j, k), 3), 2);
            DG(i, j, k) = trapz(binsF, ...
            trapz(binsG, integrandDG(i, :, :, j, k), 3), 2);
            DGG(i, j, k) = trapz(binsF, ...
            trapz(binsG, integrandDGG(i, :, :, j, k), 3), 2);
        end
    end
end

elapsedSystemId = toc(startSystemId);
disp(['... ' num2str(elapsedSystemId) ' s'])

% % ANALYTICAL

fVec = linspace(0, max(binsF), 1000); fVec = fVec(fVec > 0);
gVec = linspace(0, max(binsG), 1000); gVec = gVec(gVec > 0);
[gGrid, fGrid] = meshgrid(gVec, fVec);

v = @(f, g) -nuF*f.^2 - nuG*g.^2 + (3*kappa/8)*(f.^4 + g.^4) ...
    + (3*kappa/2)*(f.^2*g.^2) - (gammax)*log(f.*g);
V = v(fGrid, gGrid);
P = exp(-V/gammax);
P = P/trapz(fVec, trapz(gVec, P, 2), 1);

[M, iM] = max(P(:));
[iMrow, iMcol] = ind2sub(size(P), iM);
FMax = fVec(iMrow);
GMax = gVec(iMcol);

DFfun = @(f, g) 2*nuF*f - 3*kappa*(0.5*f.^2 + g.^2).*f + gammax/f;
DGfun = @(f, g) 2*nuG*g - 3*kappa*(0.5*g.^2 + f.^2).*g + gammax/g;

DFanalytical = DFfun(fGrid, gGrid);
DGanalytical = DGfun(fGrid, gGrid);

% % PLOTTING
% % . . . TIME

figSignal = figure;
subplot(2, 1, 1), hold on, grid on
plot(timeRaw(timeRaw<timePlotRange), fRaw(timeRaw<timePlotRange), 'k-')
plot(timeCut(timeCut<timePlotRange), fCut(timeCut<timePlotRange), 'k-')
plot([1 1]*timeTransientCutoff, get(gca, 'ylim'), 'k--')
xlabel('$t$ (s)')
ylabel('$F$')
ylabel('$F$')

figSignalFull = figure;
subplot(2, 1, 1), hold on, grid on
plot(timeRaw(timeRaw<timePlotRange), fRaw(timeRaw<timePlotRange), 'k-')
plot(timeCut(timeCut<timePlotRange), gCut(timeCut<timePlotRange), 'k-')
plot([1 1]*timeTransientCutoff, get(gca, 'ylim'), 'k--')
xlabel('$t$ (s)')
ylabel('$G$')

figSignalFull = figure;
subplot(2, 1, 1), hold on, grid on
plot(timeRaw, fRaw, 'k-')
xlabel('$t$ (s)')
ylabel('$F$')

figSignalFull = figure;
subplot(2, 1, 1), hold on, grid on
plot(timeRaw, gRaw, 'k-')
xlabel('$t$ (s)')
ylabel('$G$')
% % . . . PDFs
figPdfF = figure; hold on, grid on
plot (binsF, pdfFavg, 'ok-')
plot ([1 1]*fMax, get(gca, 'ylim'), 'k--')
ylabel('P(F)')
xlim([0 max(binsF)])

figPdfG = figure; hold on, grid on
plot (binsG, pdfGavg, 'ok-')
plot ([1 1]*gMax, get(gca, 'ylim'), 'k--')
ylabel('P(G)')
xlim([0 max(binsG)])

figJpdfFG = figure; hold on, grid on
contourf (binsF, binsG, jpdfFGavg.')
colormap (flipud(hot))
plot ([1 1]*fMax, get(gca, 'ylim'), 'k--')
plot (get(gca, 'xlim'), [1 1]*gMax, 'k--')
plot (fMax, gMax, 'ow')
xlabel('F')
ylabel('G')
title('$P(F, G) |$ Calculated')
xlim([0 max(binsF)])
ylim([0 max(binsG)])

figJpdfFGanalytical = figure; hold on, grid on
contourf(fVec, gVec, P.')
colormap (flipud(hot))
plot ([1 1]*fMax, get(gca, 'ylim'), 'k--')
plot (get(gca, 'xlim'), [1 1]*gMax, 'k--')
plot (fMax, gMax, 'ow')
ylabel('P(F, G) | Analytical')
xlim([0 max(binsF)])
ylim([0 max(binsG)])

tauPlot = tau( iTauPlot ) ;
tPlot = round( fMax/dF ) + 1 ;
gPlot = round( gMax/dG ) + 1 ;
figFGFGpdfAvg = figure; hold on, grid on
contourf(fVec, gVec, squeeze(jpdfFGFGavg(iTauPlot, :, :, ifPlot, igPlot))
colormap (flipud(hot))
plot ([1 1]*fMax, get(gca, 'ylim'), 'k--')
plot (get(gca, 'xlim'), [1 1]*gMax, 'k--')
plot (fMax, gMax, 'ow')
ylabel('P(F, G) | Analytical')
title('$P( f, g, F = ' num2str(fPlot) ', G = ' num2str(gPlot) ' ) \tau =$')

% % ... KRAMERS-Moyal
iTauPlot = 1;
tPlot = tau(iTauPlot);
ifPlot = round(fMax/dF) + 1;
gPlot = round(gMax/dG) + 1;
figFGFGpdfAvg = figure; hold on, grid on
contourf(fVec, gVec, squeeze(jpdfFGFGavg(iTauPlot, :, :, ifPlot, igPlot))
colormap (flipud(hot))
plot ([1 1]*fMax, get(gca, 'ylim'), 'k--')
plot (get(gca, 'xlim'), [1 1]*gMax, 'k--')
plot (fMax, gMax, 'ow')
ylabel('P(f, g, F = $' num2str(fPlot) ', G = ' num2str(gPlot) ') \tau =$

76
425 xlim([0 max(binsF)])
426 ylim([0 max(binsG)])
427
428 figFGFGcpdf = figure; hold on, grid on
429 contourf(binsF, binsG, squeeze(cpdfFGFG(iTauPlot, :, :, ifPlot, igPlot)).')
430 colormap(flipud(hot))
431 plot(get(gca, 'xlim'), [1 1]*gMax, 'k--')
432 plot(get(gca, 'ylim'), [1 1]*fMax, 'k--')
433 plot(fMax, gMax, 'ow')
434 xlabel('$f$')
435 ylabel('$g$')
436 title(['$P(f, g | F =$ ' num2str(fPlot) ', G = ' num2str(gPlot) ')$ @ ($\tau =$ '$\tau$')] )
437 xlim([0 max(binsF)])
438 ylim([0 max(binsG)])
439
440 figDFcalculated = figure; hold on, grid on
441 contourf(binsF, binsG, squeeze(DF(iTauPlot, :, :, :))).')
442 colormap(flipud(hot))
443 plot([1 1]*fMax, get(gca, 'ylim'), 'k--')
444 plot(get(gca, 'xlim'), [1 1]*gMax, 'k--')
445 plot(fMax, gMax, 'ow')
446 xlabel('$F$')
447 ylabel('$G$')
448 title(['$D{(1)}{\{F, G, \tau =$ ' num2str(tauPlot) '}$] )
449 xlim([0 max(binsF)])
450 ylim([0 max(binsG)])
451
452 figDGcalculated = figure; hold on, grid on
453 contourf(binsF, binsG, squeeze(DG(iTauPlot, :, :, :))).')
454 colormap(flipud(hot))
455 plot([1 1]*gMax, get(gca, 'ylim'), 'k--')
456 plot(get(gca, 'xlim'), [1 1]*fMax, 'k--')
457 plot(fMax, gMax, 'ow')
458 xlabel('$F$')
459 ylabel('$G$')
460 title(['$D{(1)}{\{G, F, \tau =$ ' num2str(tauPlot) '}$] )
461 xlim([0 max(binsF)])
462 ylim([0 max(binsG)])
463
464 figDFanalytical = figure; hold on, grid on
465 contourf(fVec, gVec, DFanalytical.)
466 colormap(flipud(hot))
467 plot([1 1]*fMax, get(gca, 'ylim'), 'k--')
468 plot(get(gca, 'xlim'), [1 1]*gMax, 'k--')
469 plot(fMax, gMax, 'ow')
470 xlabel('$F$')
471 ylabel('$G$')
472 title('DF{(1)}{(F, G)$ | Analytical$}
473 xlim([0 max(binsF)])
474 ylim([0 max(binsG)])
475
476 figDGanalytical = figure; hold on, grid on
477 contourf(fVec, gVec, DGanalytical.)
478 colormap(flipud(hot))
479 plot([1 1]*gMax, get(gca, 'ylim'), 'k--')
480 plot(get(gca, 'xlim'), [1 1]*fMax, 'k--')
481 plot(fMax, gMax, 'ow')
482 xlabel('$F$')
483 ylabel('$G$')
484 title('DG{(1)}{(G, F)$ | Analytical$}
485 xlim([0 max(binsF)])
486 ylim([0 max(binsG)])
487
488 % % AFPE Comparison
489 dLims = [0 5 0 5];
490 sMesh = 0.07;
491 tList = 0:0.001:max(tau);
492 params = [nuF nuG kappa gammax];
493 bins = [binsF binsG];
494 isPlot = 0;
497 \text{indF} = \text{round}(f_{\text{Max}}/dF) + 1;  
498  
499 \text{startAfpe} = \text{tic};  
500  
501 D_{\text{AFPE}} = \text{afpeSolveFG}(dLims, sMesh, tList, params, 'F', [binsF(indF) 1], bins,  
502 isPlot);  
503  
504 \text{elapsedAfpe} = \text{toc}(\text{startAfpe});  
505  
506 DFosc = \text{squeeze}(D_{\text{F}}(:, indF, :));  
507  
508 figDFcompare = \text{figure}; \text{hold on}, \text{grid on}, \text{box on}  
509  
510 \text{plot}(\tau, DFosc(:, indG1), 'or')  
511 \text{plot}(\tau, DFosc(:, indG2), 'ob')  
512 \text{plot}(\tau, DFosc(:, indG3), 'og')  
513 \text{plot}(\tau, DFosc(:, indG4), 'ok')  
514  
515 \text{plot}(tList, D_{\text{AFPE}}(:, indG1), 'r-')  
516 \text{plot}(tList, D_{\text{AFPE}}(:, indG2), 'b-')  
517 \text{plot}(tList, D_{\text{AFPE}}(:, indG3), 'g-')  
518 \text{plot}(tList, D_{\text{AFPE}}(:, indG4), 'k-')  
519  
520 \text{plot}(0, \text{DFfun}(\text{binsF}(\text{indF}), G1), '*r')  
521 \text{plot}(0, \text{DFfun}(\text{binsF}(\text{indF}), G2), '*b')  
522 \text{plot}(0, \text{DFfun}(\text{binsF}(\text{indF}), G3), '*g')  
523 \text{plot}(0, \text{DFfun}(\text{binsF}(\text{indF}), G4), '*k')  
524  
525 \text{xlabel}('\tau')  
526 \text{ylabel}(['D_1 F^{(1)}(\tau, F = ' num2str(binsF(indF)) ' , G = ... , '])  
527 \text{legend}(['G = ' num2str(G1), ...  
528 \text{save}('oscFGvFINALdata.mat', allvars {tosave}.name)  
530 \text{elapsedScript} = \text{toc}(\text{startScript});  
531  
532 B.4 Rotating Modes: AFPE Solver  
533  
534 function D = afpeSolveFG(dLims, sMesh, tList, params, ForG, init, bins,  
535 isPlot)  
536  
537 \% Inputs:  
538 \% dLims : Domain limits : [fMin fMax gMin gMax]  
539 \% sMesh : Maximum mesh element size  
540 \% tList : Times to find solution : [t1 t2 ... tnT]  
541 \% params : System parameters : [nuF nuG kappa gamma/4w2]  
542 \% ForG : Either 'F' for D_{F}^{(n)} or 'G' for D_{G}^{(n)}  
543 \% init : For either (f - X)^n or (g - X)^n, dependent on ForG : [X, n]  
544 \% bins : Interpolate solutions to bins {binF, binG}  
545 \% isPlot : Plot the results : 1 (true) or 0 (false)  
546 \% Outputs:  
547 \% uIntrp : Solution interpolated to mesh of bins : nTxnFxmg  
548 \% D : Kramers–Moyal coefficient at ForG = X : nTxnF or nTxnG  
549 \% No boundary conditions used in the solver.  
550  
551 \% PARAMETERS  
552 \% nuF = params(1); nuG = params(2); kappa = params(3); gammmax = params(4);  
553 \% X = init(1); n = init(2);  
555 \% binsF = bins{1}; binG = bins{2};  
556 \% nF = length(binF); nG = length(binG);  
557 \% nT = length(tList);
% INITIALISE
model = createpde();

% GEOMETRY
xMin = dLims(1); xMax = dLims(2); % Domain limits: \( x = f, y = g \)
yMin = dLims(3); yMax = dLims(4);

gd = [3; ... ]; % Geometry description: 3 = rectangle
4; ... % N = 4 edges
  xMin; xMax; xMax; xMin; ... % x-coordinates
  yMin; yMin; yMax; yMax]; % y-coordinates
dl = decsg(gd);
generateMesh(model, 'Hmax', sMesh);

% MESH

% COEFFICIENTS
m = 0;
d = 1;
c = [gammax; gammax];
a = 0;

function f = coefFunF(region, state)
f = (2*nuF*region.x^2 + region.y.^2).*region.x + ... + (2*nuF*region.y^2 + region.x.^2).*region.y + ...
  (gammax./region.x).*state.ux + ...
  (gammax./region.y).*state.uy;
end

function f = @coefFunF;
specifyCoefficients(model, 'm', m, ...
                   'd', d, ...
                   'c', c, ...
                   'a', a, ...
                   'f', f);

% INITIAL CONDITIONS

function u0F = initFunF(locations)
u0F = (locations.x - X).^n;
end

function u0G = initFunG(locations)
u0G = (locations.y - X).^n;
end

switch ForG
  case 'F'
    u0 = @initFunF;
  case 'G'
    u0 = @initFunG;
end

setInitialConditions(model, u0);

% SOLVE
result = solvepde(model, tList);

% Interpolate solution to bin mesh
[gIntrp, fIntrp] = meshgrid(binG, binF);
queryPoints = [fIntrp(:), gIntrp(:)];
ultrp = interpolateSolution(result, queryPoints, 1:nT);
ultrp = reshape(ultrp, [nF, nG, nT]); % nFxnGxnT
uIntrp = permute(uIntrp, [3 1 2]); % nTxnFxnG

switch ForG
  case 'F'
    uX = squeeze(uIntrp(:, binF == X, :)); % ntxnG
    D = uX./(tList'*ones(1, nG));
  case 'G'
    uX = squeeze(uIntrp(:, :, binG == X)); % ntxnF
    D = uX.*((tList'*ones(1, nF));
end

if isPlot
  set(groot, 'defaultTextInterpreter', 'latex')
  set(groot, 'defaultLegendInterpreter', 'latex')
  set(groot, 'defaultAxesTickLabelInterpreter', 'latex')
  figNodal = figure; hold on, grid on
  pdeplot(model, 'XYData', u(:, end), 'colorbar', 'off', 'mesh', 'off')
  xlabel('$f$')
  ylabel('$g$')
  title(['$P^{\dagger}(f, g, t = $ ' num2str(tList(end)) ')$ - Nodal'])
  colormap(flipud(hot))
  view([90 -90])
  figIntrp = figure; hold on, grid on
  contourf(gIntrp, fIntrp, squeeze(uIntrp(end,:,:)))
  xlabel('$f$')
  ylabel('$g$')
  title(['$P^{\dagger}(f, g, t = $ ' num2str(tList(end)) ')$ - Interpolated'])
  colormap(flipud(hot))
  switch ForG
    case 'F'
    [pp, tt] = meshgrid(binG, tList);
    case 'G'
    [pp, tt] = meshgrid(binF, tList);
  end
  figUx = figure; hold on, grid on
  mesh(pp, tt, uX)
  xlabel('$G$')
  ylabel('$\tau$')
  zlabel('$P^{\dagger}$')
  title(['$n = $ ' num2str(n) ', ' ForG ' = ' num2str(X)'])
  switch ForG
    case 'F'
    xaxis('F' $G$
    case 'G'
    xaxis('G' $F$
  end
  figD = figure; hold on, grid on
  mesh(pp, tt, D)
  xlabel('$\tau$')
  ylabel('$D$')
  zlabel('$P^{\dagger}$')
  title(['$n = $ ' num2str(n) ', ' ForG ' = ' num2str(X)'])
  switch ForG
    case 'F'
    xaxis('F' $G$
    case 'G'
    xaxis('G' $F$
  end
end
References


