Optimal Sensor Location for the Estimation of a Linear Dispersive Wave Equation

by

Tawsif Khan

A thesis
presented to the University of Waterloo
in fulfillment of the
thesis requirement for the degree of
Master of Mathematics
in
Applied Mathematics

Waterloo, Ontario, Canada, 2015

I hereby declare that I am the sole author of this thesis. This is a true copy of the thesis, including any required final revisions, as accepted by my examiners.

I understand that my thesis may be made electronically available to the public.

Abstract

The problem of optimal sensor location for the estimation of a linear dispersive wave equation is considered in this work. A steady-state Kalman filter was used as an optimal state observer with localized velocity information as the measurement. Since the main model is a partial differential equation, the states evolve in infinite-dimensional spaces, and hence an approximation (finite representation) was required to design the observer. Three different approximation methods were compared - eigenfunctions and finite element methods using a linear and a high-order polynomial basis. The latter two methods are the more common choice of approximation schemes for systems with a complex geometry. It was found that the eigenfunctions perform much better as expected. The finite element methods require larger matrices to approximate the system with reasonable accuracy and hence calls for numerical methods to solve the Algebraic Riccati Equation efficiently. The optimal sensor location was considered for three different noise models - a localized noise, a more distributed nature of noise and finally an one-dimensional turbulence model. It was seen that the optimal location tend to be closer to the point where the physical shape of the noise reaches its maximum. Placing the sensor in the optimal location showed significant improvement in the estimation process.

Acknowledgements

I would like to thank my supervisors Professor Kirsten Morris and Professor Marek Stastna for their continuous guidance throughout my program. I was able to learn a lot of different facets of mathematics and research. Their patience, trust and friendliness were very important for my successful completion of this work.

I would also like to thank all my collegaues in the department - specially Sepideh, Ambroise, Brian, Lindsey, Tom, Arman and Amir. These are some good people and I wish them all the very best in their own personal pursuits. My sincere gratefulness also goes to our departmental secretary Laura Frazee whom I always found to be concerned and helpful towards all the graduate students.

Finally, I would like to thank my family and friends whose trust and love makes me keep going forward.

Table of Contents

List of Tables

List of Figures

Chapter 1

Introduction

In the field of environmental and geophysical fluid dynamics real time measurements from a region of interest play a vital role in forecasting. Given that these regions, in most cases, would involve large spatial scales, measurements spanning the entire region would be difficult to obtain. For example consider an ocean and the objective of measuring wave heights that can be used to predict the evolution of wave heights in the entire region. Since deploying an infinite number of measurement equiment is never feasible and given the high costs of some sensing equipment (e.g. Acoustic Doppler Current Profilers - ADCPs), as well as the deployment of such equipment in the field, it is highly desirable to optimize the amount of information gained from a particular deployment of sensors. In the field of control theory, one research area that is dedicated towards such an objective is that of optimal sensor location, where the goal is to find the optimal location of a sensor with respect to a cost.

The first step in any such work would be to create a mathematical model as accurate as possible to describe the physics that is of interest which in this thesis involves a one-dimensional shallow water body. A shallow water body will have different parameters associated with it, for instance the wave heights or a three-dimensional velocity at different locations, temperature or even water quality. The mathematical model should represent the evolution of one or more of these parameters. The second step would be to again create a model for the sensor, which represents how and where one or more of these parameters are measured. Then a cost function with respect to sensor location is defined and the location of the sensor where this cost would be minimum is searched for. There are well established optimal estimation techniques with corresponding cost functions like the H_2 and H_∞ criteria. The optimal estimation and the corresponding optimal sensor location problem is mathematically dual to the optimal control and optimal actuator loca-

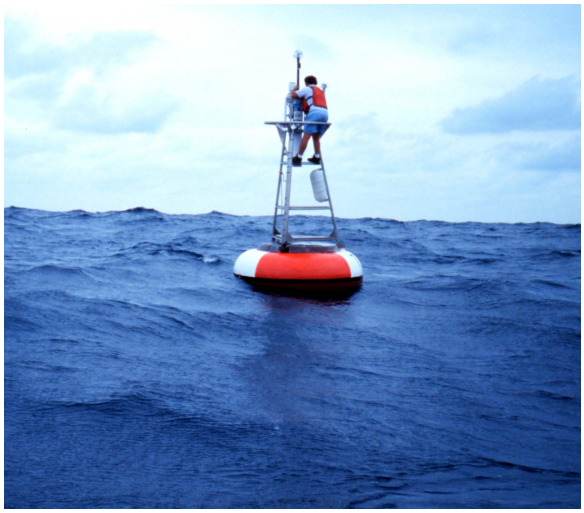

Figure 1.1: A National Oceanic and Atmospheric Administration (NOAA) picture of a buoy deployment which measures various types of quantities

tion problem. These problems have been considered by many researchers for a variety of different contexts and physical systems; see for example [16, 32, 44, 47, 48] and the review [18]. A notable technique used for forecasting by the oceanographic and meteorological communities is data assimilation using Kalman filter, for example [23].

Apart from the steps that were mentioned briefly above, another step that is crucial in these works is that of approximation of the original system. The original systems are almost always represented by partial differential equations, for example the wave and the heat equation. Since in most cases, a closed form solution to such equations are not available, these infinite-dimensional systems require approximations by a finite-dimensional system. In [47], for instance, authors used a Galerkin approximation of a system describing the deflection of a beam, and in [16] and [6] the authors used a Legendre-Galerkin spectral method. Designing a controller or an observer using approximations hence raises questions involving accuracy and convergence. In [26, 34, 35] the issue of obtaining the correct optimal actuator location when approximations are used is addressed. The paper [9] presents an algorithm suitable for multiple actuator (or sensor) placement in large order systems. In [30], the optimal actuator/sensor problem was considered for the nonlinear control of the Kuramoto-Sivashinsky equation using a Fourier basis as the discretization. In [4, 5] combined actuator/sensor location in fluid flow with a H_2 criterion was considered using a polynomial basis for the calculations. Their results indicate that placing the sensor slightly downstream of the disturbance is often optimal. The study of different LQG

metrics for thermostat placement in a room with an advection-diffusion model [2] suggests less sensitivity to sensor location in that context.

In this thesis project, the optimal sensor location problem for estimating the state of a linear, dispersion modified shallow water equations in one spatial dimension is considered. It is the dual problem of the optimal actuator location, so results and algorithms developed for an optimal actuator location can be used. In doing so three different approximation methods are used to approximate the same system – one based on eigenfunctions and the other two based on finite element methods using a linear and a high order polynomial basis. For the estimation technique, a continuous time Kalman filter was implemented. There are two main objectives in this thesis: compute the optimal sensor location for the model and explain how higher order finite elements would work for such a problem and provide a detailed analysis of these different approximation methods that can be used. A crucial factor that has not been yet mentioned is the model of the noise affecting the physical systems. An optimal sensor location problem is assumed to depend on the noise characteristics and thus may call for accurate noise models. In this report, two different noise models will be discussed – one where the effects are locallized and one where it is more distributed over the entire region. In Chapter 2, the physical system of interest and its mathematical model is introduced with some important analysis. It is followed by the introduction of continuous time Kalman filtering in Chapter 3. Chapter 3 will formulate the main optimal sensor location problem with descriptions of the sensor and noise, followed by numerical results.

Chapter 2

Description of the Model

The focus of this work is on the calculating the optimal sensor location for the estimation of an one dimensional linear dispersive wave equation, which will be derived in this chapter. From the most general three-dimensional Navier-Stokes equations for incompressible flow, the model will be derived through common simplifications and reductions, and finally modified by the inclusion of dispersive and damping effects in surface waves.

2.1 Navier-Stokes Equation

The set of Navier-Stokes equations is the most general model describing fluid flow using the concept of a continuum and Newton's second law for the conservation of momentum. The derivation can be found in [27] and the required equation can be represented as,

$$\rho_0 \frac{D\mathbf{v}}{Dt} = \nabla \cdot \tau + \nabla \Pi \tag{2.1}$$

where $\mathbf{v} = (v(x, y, z, t), u(x, y, z, t), w(x, y, z, t))$ is the three-dimensional velocity and ρ_0 is the fluid density . The two terms on the right hand side correspond to the pressure forces and body forces, where τ is the stress tensor and Π is the force potential. The operator,

$$\frac{D}{Dt} = \frac{\partial}{\partial t} + \mathbf{v} \cdot \nabla$$

is called the material derivative which is the time derivative of a property in a velocity field (following a fluid particle).

In Cartesian co-ordinates the stress tensor is a second order tensor with nine components. For an incompressible fluid, this tensor can be simplified to give the stress in terms of a static state and a dynamic state contribution[27, pp.111-113],

$$\tau_{ij} = -p\delta_{ij} + \sigma_{ij}, \tag{2.2}$$

where p is the pressure, δ_{ij} is the Kronecker delta and σ_{ij} is the stress due to viscous forces. The two indices, i and j, on the stress tensor correspond to the normal of the surface (on which the force is acting) and the component of the force on that particular surface. For the static state the stress is only due to isotropic pressure forces which are normal to the surface and hence contribute to the diagonals of the total stress through the delta term. The dynamic contribution is given by σ_{ij} which contribute to both diagonal and off-diagonal components of the tensor. These stress terms arise due to viscosity affecting the fluid flow.

Due to the scales considered and the choice of water as the fluid, the viscous terms were dropped. For the body forces, the buoyancy force due to gravitational acceleration acting along the z-axis is taken into account. Considering these two forces, (2.1) becomes

$$\rho_0 \frac{D\mathbf{v}}{Dt} = -\nabla p - \rho_0 g\mathbf{k} \ , \tag{2.3}$$

where $g\mathbf{k}$ is the gravitational acceleration. The pressure term in (2.3) can be modified to absorb the buoyancy the term. Consider the form,

$$p = p' + p(z),$$

where $p(z)$ is given by

$$p(z) = -\rho_0 g z$$
$$\nabla p(z) = -\rho_0 g\mathbf{k}.$$

Therefore,

$$-\nabla p = -\nabla p' + \rho_0 g\mathbf{k} \ .$$

Substituting ∇p in (2.3) and dropping the prime notation gives,

$$\rho_0 \frac{D\mathbf{v}}{Dt} = -\nabla p \ . \tag{2.4}$$

The incompressibility constraint implies that the mass must be conserved which gives the continuity equation,

$$\nabla \cdot \mathbf{v} = 0 \ . \tag{2.5}$$

Equations (2.4) and (2.5) will be required to derive an 1-layer shallow water model in the following section.

2.2 Layered Models

The important physics behind the adoption of layered models is stratification in water bodies. Stratification means the vertical structure of the fluid is governed by the density variations in vertical direction, where the least dense fluid is at the top or surface. In this work, the focus is on surface gravity waves in shallow water in a single space dimension. Therefore, a 1-layer (density) model for water bodies where the depth H and L satisfy the shallow water assumption, $\frac{H}{L} \ll 1$, will be derived in this section.

2.2.1 One Layer Shallow Water Model

Consider a shallow water region with a constant depth (flat bottom) of H and a free surface height $\eta(x, y, t)$. The velocity is denoted by $\mathbf{v}(x, y, z, t)$. See Figure 2.2.1 for a schematic of the model setup. Note the notational difference between $\mathbf{v}(x, y, z, t)$, which is a vector and $v(x, y, t)$ which is the scalar velocity along the x-axis at (x, y, t). It is instructive for readers to carry out the derivation using the more general 3D model to a standard 2D model for shallow water and then finally reducing the model to a single dimension for this work. The 3D model is reduced to a 2D model by averaging the velocity over depth which is justified by the horizontal scales that are considered. Due to the large horizontal scales, the vertical accelerations can be ignored. This derivation can be found be in [27]. To reduce the model

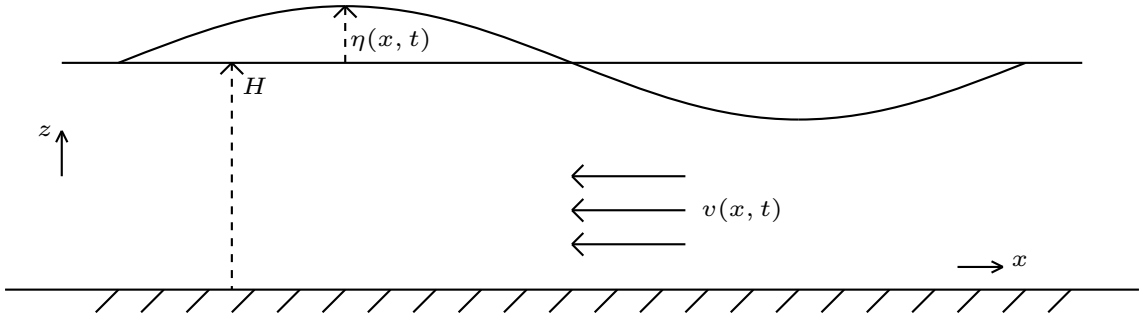

Figure 2.1: Schematic a one layer shallow water model

to two dimensions, integrate the continuity equation to give,

$$\int_0^{H+\eta} w_z dz + \int_0^{H+\eta} (v_x + u_y) dz = 0 \tag{2.6}$$

Let $\nabla \cdot \mathbf{v} = v_x + u_y$ and write (2.6) as

$$w(\cdot, \cdot, H + \eta) - w(\cdot, \cdot, 0) + (H + \eta)\nabla \cdot \mathbf{v} = 0 \ . \tag{2.7}$$

Note that the terms on the left hand side are the vertical velocities at the boundaries - surface and the bottom. It is clear that the vertical velocity is,

$$w = 0, \text{ at } z = 0 \ .$$

For $z = H + \eta$ a kinematic boundary condition is imposed which implies the position of a fluid particle remain unchanged in a frame of reference that is moving with the flow. That is,

$$w = \frac{D(H + \eta)}{Dt} = \frac{\partial(H + \eta)}{\partial t} + \mathbf{v} \cdot \nabla(H + \eta), \text{ at } z = H + \eta \tag{2.8}$$

Recall (2.7) and use the boundary conditions to get,

$$w(\cdot, \cdot, H + \eta) - w(\cdot, \cdot, 0) + (H + \eta)\nabla \cdot \mathbf{v} = 0 \tag{2.9}$$

$$\frac{\partial \eta}{\partial t} + \mathbf{v} \cdot \nabla(H + \eta) + (H + \eta)\nabla \cdot \mathbf{v} = 0 \tag{2.10}$$

$$\frac{\partial \eta}{\partial t} + \nabla \cdot ((H + \eta)\mathbf{v}) = 0 \ . \tag{2.11}$$

Due to the assumption that horizontal length scales are much larger than the vertical excursions, the vertical acceleration can be ignored, which gives the hydrostatic pressure condition. That is,

$$-\frac{1}{\rho_0}\frac{\partial p}{\partial z} - \frac{g}{\rho_0} = 0 \ , \tag{2.12}$$

$$p = \rho_0 g h \ . \tag{2.13}$$

To average the momentum equation, consider the hydrostatic pressure condition. The pressure at height z depends on the mass of water above and the atmospheric pressure at the surface, p_0. Therefore,

$$p(z) = p_0 + \int_z^\eta \rho_0 g dz'$$

Therefore horizontal pressure gradients are

$$\frac{\partial p}{\partial x} = \rho_0 g \frac{\partial \eta}{\partial x} \ , \quad \frac{\partial p}{\partial y} = \rho_0 g \frac{\partial \eta}{\partial y} \ . \tag{2.14}$$

Substituting (2.14) into the horizontal components of (2.1) gives the shallow water equations

$$\frac{\partial v}{\partial t} + v\frac{\partial v}{\partial x} + u\frac{\partial v}{\partial y} = -g\frac{\partial \eta}{\partial x}, \tag{2.15}$$

$$\frac{\partial u}{\partial t} + v\frac{\partial u}{\partial x} + u\frac{\partial u}{\partial y} = -g\frac{\partial \eta}{\partial y}, \tag{2.16}$$

$$\frac{\partial \eta}{\partial t} + \nabla \cdot ((H+\eta)\mathbf{v}) = 0 \ . \tag{2.17}$$

2.2.2 Linearization of Shallow Water Equations

The model is now reduced to a single dimension by dropping one of the dimensions. This simplifies (2.17) to give,

$$\frac{\partial \eta}{\partial t} + H\frac{\partial v}{\partial x} + \eta\frac{\partial v}{\partial x} + v\frac{\partial \eta}{\partial x} = 0 \ ,$$

where the last two terms correspond to advection. For small amplitude waves, the quadratic nonlinear terms can be neglected in comparison to the linear term [27]. Thus the equations simplify to,

$$\frac{\partial v}{\partial t} = -g\frac{\partial \eta}{\partial x} \tag{2.18}$$

$$\frac{\partial \eta}{\partial t} = -H\frac{\partial v}{\partial x} \ . \tag{2.19}$$

in a single space dimension.

2.2.3 Dispersive Shallow Water Model

The linearized shallow water equations (2.18-2.19) admit wave-like solutions that are non-dispersive unlike the full Navier-Stokes model. In order to improve the model, dispersion can be included. A flow is called dispersive when the speed of the waves depends on the wavelength [46]. In [12], the assumption of a purely hydrostatic pressure was weakened and through a perturbation method, a correction to the hydrostatic pressure was obtained, that models surface wave dispersion reasonably well. In particular they capture the observations that longer waves propagate faster. The linearized governing equations of the de

La Fuente's model [12] are,

$$\frac{\partial v}{\partial t} = -g\frac{\partial \eta}{\partial x} + \frac{H^2}{6}\frac{\partial^2}{\partial x^2}\left(\frac{\partial v}{\partial t}\right) \tag{2.20}$$

$$\frac{\partial \eta}{\partial t} = -H\frac{\partial v}{\partial x}\ . \tag{2.21}$$

2.2.4 Boundary Layer Friction

The linear dispersive wave equations (2.20 -2.21) form a non-dissipative system in contrast to what observations indicate. An important mechanism that leads to the energy loss of the system is the friction at the bottom layer and has been studied in [22, 29, 39]. In an attempt to paramatrize this loss of energy without increasing complexity, a damping term is added to the model ,

$$\frac{\partial v}{\partial t} = -g\frac{\partial \eta}{\partial x} + \frac{H^2}{6}\frac{\partial^2}{\partial x^2}\left(\frac{\partial v}{\partial t}\right) - c_d v \tag{2.22}$$

$$\frac{\partial \eta}{\partial t} = -H\frac{\partial v}{\partial x}\ . \tag{2.23}$$

Here the damping constant, c_d, controls the amount of damping. In section 2.3.2 it will be shown that this system is indeed dissipative.

2.3 State Space Formulation

2.3.1 State-Space

By cross-differentiating the governing equations (2.22) and (2.23) the model can be reduced to the following single equation,

$$v_{tt} = c^2 v_{xx} - c_d v_t + \beta v_{xxtt}, \quad x \in [0, L] \tag{2.24}$$

where subscripts denote partial derivatives. The constants, $c^2 = gH$ and $\beta = \frac{H^2}{6}$, are positive. The boundary conditions are,

$$v(0, t) = v(L, t) = 0. \tag{2.25}$$

Writing (2.24) in a first-order descriptor form gives,

$$\begin{bmatrix} I & 0 \\ 0 & I - \beta D^2 \end{bmatrix} \begin{bmatrix} \dot{v} \\ v_t \end{bmatrix} = \begin{bmatrix} 0 & I \\ c^2 D^2 & -c_d I \end{bmatrix} \begin{bmatrix} v \\ v_t \end{bmatrix}, \tag{2.26}$$

where I denotes the identity operator and $D(\cdot) = \frac{\partial}{\partial x}(\cdot)$.

Define the space

$$\mathcal{H}_0^1(0, L) = \{v \in \mathcal{H}^1(0, L),\ v(0) = v(L) = 0\}\ .$$

where $\mathcal{H}^1$ is the Sobolev space of functions with weak first derivatives [41]. Define the state $z(t) = [v(\cdot, t)\ \dot{v}(\cdot, t)]$ and the state-space $\mathcal{Z} = \mathcal{H}_0^1(0, L) \times \mathcal{L}_2(0, L)$.

The Sobolev space $\mathcal{H}^1$ is a Hilbert space with the inner-product [41],

$$\langle z_1, z_2 \rangle_{\mathcal{H}^1} = \langle z_1, z_2 \rangle + \langle z_1', z_2' \rangle,\ \ z_1, z_2 \in \mathcal{H}^1$$

To define an inner-product on $\mathcal{H}_0^1$ the above inner-product can be used directly. However, it can be shown that, for $z_1, z_2 \in \mathcal{H}_0^1$, the inner product

$$\langle z_1', z_2' \rangle,$$

yields an equivalent norm on $\mathcal{H}_0^1$.

Proof. It is easy to see that,

$$||z||_{\mathcal{H}^1} = (||z||^2 + ||z'||^2)^{\frac{1}{2}} \geq c_1 ||z'||,$$

where $c_1 = 1$. For the reverse inequality, note that

$$\begin{aligned} z(x) &= z(0) + \int_0^x z'(\hat{x}) d\hat{x} \\ &= 0 + \int_0^x z'(\hat{x}) d\hat{x} \\ &\leq ||z'|| \int_0^x 1 d\hat{x} \\ &= x||z'|| \end{aligned}$$

Therefore,

$$||z||_{\mathcal{H}^1} = (||z||^2 + ||z'||^2)^{\frac{1}{2}} \leq (1 + x^2)^{\frac{1}{2}} ||z'|| \leq c_2 ||z'||\ ,$$

where $c_2 = 1 + L^2$. □

Therefore, herein the space $\mathcal{H}_0^1$ is equipped with the following inner-product

$$\langle z_1, z_2 \rangle_{\mathcal{H}_0^1} = \langle z_1', z_2' \rangle \ .$$

Let $z = \begin{pmatrix} z_1 \\ z_2 \end{pmatrix} \in \mathcal{Z}$ and $w = \begin{pmatrix} w_1 \\ w_2 \end{pmatrix} \in \mathcal{Z}$, and define the following inner-product on $\mathcal{Z}$,

$$\begin{aligned} \langle z, w \rangle_{\mathcal{Z}} &= c^2 \langle z_1, w_1 \rangle_{\mathcal{H}_0^1} + \langle z_2, w_2 \rangle \\ &= c^2 \langle z_1', w_1' \rangle + \langle z_2, w_2 \rangle \ , \end{aligned}$$

which yields the norm

$$|| \begin{pmatrix} z_1 \\ z_2 \end{pmatrix} ||_{\mathcal{Z}}^2 = \frac{1}{c^2} ||z_1'||^2 + ||z_2||^2 \ .$$

Define the inverse operator of $(I - \beta D^2)$, T and operator D^2 such that,

- $T : \mathcal{L}_2(0, L) \to \mathcal{L}_2(0, L), \mathcal{D}(T) = \mathcal{L}_2(0, L)$,
- $D^2 : \mathcal{D}(D^2) \subset \mathcal{H}_0^1(0, L) \to \mathcal{L}_2(0, L), \mathcal{D}(D^2) = \mathcal{H}^2(0, L)$.

The inverse of $(I - \beta D^2)$ can be calculated considering its Sturm-Liouville form. The calculation, given in Section A.1, leads to the following bounded operator,

$$Tv(x) = \int_0^L g(x, y) v(y) dy,$$

where

$$g(x, y) = \frac{\alpha}{2(1 - e^{2\alpha})} \begin{cases} (e^{\alpha x} - e^{-\alpha x})(e^{\alpha y} - e^{2L\alpha} e^{-\alpha y}), & 0 \leq x \leq y \leq L \\ (e^{\alpha x} - e^{2L\alpha} e^{-\alpha x})(e^{\alpha y} - e^{-\alpha y}), & 0 \leq y \leq x \leq L \end{cases}$$

with $\alpha = \frac{1}{\sqrt{\beta}}$.

Using the operator T, the system in (2.26) can now be reformulated as

$$\begin{bmatrix} \dot{v} \\ v_t \end{bmatrix} = \begin{bmatrix} 0 & I \\ c^2 T D^2 & -c_d T \end{bmatrix} \begin{bmatrix} v \\ v_t \end{bmatrix} \ . \tag{2.27}$$

Define

$$A = \begin{bmatrix} 0 & I \\ c^2 T D^2 & -c_d T \end{bmatrix}$$

with the domain

$$\mathcal{D}(A) = \{ \begin{pmatrix} z_1 \\ z_2 \end{pmatrix} \in \mathcal{Z}, z_1 \in \mathcal{H}^2, z_2 \in \mathcal{H}_0^1 \} \ .$$

Then a state space formulation of the partial differential equation in (2.24) is

$$\frac{d}{dt} z(t) = Az(t). \tag{2.28}$$

2.3.2 Well-posedness

Definition 2.1. (Well-posed problem, [15, Thm 6.7, Def 6.8]). Let $A : \mathcal{D}(A) \subset X \to X$ be a closed operator. Then the associated abstract Cauchy problem

$$\begin{aligned} &\frac{d}{dt} z(t) = Az(t), \ t \geq 0, \\ &z(0) = z_0 \end{aligned}$$

is well-posed if for every $z_0 \in \mathcal{D}(A)$, there exists a unique solution $z(\cdot, z_0)$.

Definition 2.2. (Strongly continuous semigroup [8, Def 2.1.2]). A strongly continuous $(\mathcal{C}_0)$ semigroup is an operator-value function $T(t)$ from $\mathbb{R}^+ \to \mathcal{Z}$ that satifies:

1. $T(t+s) = T(t)T(s) \quad$ for $t, s \geq 0$,
2. $T(0) = I$,
3. $||T(t)z_0 - z_0|| \to 0 \quad$ as $t \to 0^+ \ \forall z_0 \in \mathcal{Z}$.

Definition 2.3. (Dissipative operator, [31]) An operator, A with domain $\mathcal{D}(A)$ on a Hilbert space X is called dissipative if

$$Re\langle Ax, x \rangle \leq 0$$

.

Theorem 2.3.1 (Lumer-Philips Theorem,[15]). *Let A be a linear operator defined on a linear subspace of $\mathcal{D}(A)$ of the Hilbert space X. Then A generates a C_0-semigroup if,*

1. *$\mathcal{D}(A)$ is dense in X ,*

2. A is closed ,

3. A is dissipative .

The operator $A : D(A) \subset \mathcal{Z} \to \mathcal{Z}$ is densely defined on $\mathcal{Z}$ since $\mathcal{H}_0^1$ is dense in $\mathcal{L}_2$.

Proof.

Lemma 2.3.2. *$\mathcal{C}_0^\infty(\Omega)$ is dense in $\mathcal{L}_2(\Omega)$*

Since $\mathcal{C}(0,L)_0^\infty \subset \mathcal{H}_0^1(0,L) \subset \mathcal{L}_0^1(0,L)$, $\mathcal{H}_0^1(0,L)$ is dense in $\mathcal{L}_0^1(0,L)$ □

Note that the choice of $\mathcal{D}(A)$ implies A is closed. The operator A is also dissipative since

$$\begin{aligned}
\langle Az, z\rangle_{\mathcal{Z}} &= \langle \begin{pmatrix} z_2 \\ c^2 z_1'' - c_d z_2 \end{pmatrix}, \begin{pmatrix} z_1 \\ z_2 \end{pmatrix} \rangle_{\mathcal{Z}} \\
&= c^2 \langle z_2', z_1' \rangle + \langle c^2 z_1'', z_2 \rangle + \langle -c_d z_2, z_2 \rangle \\
&= c^2 \langle z_1', z_2' \rangle - c^2 \langle z_1', z_2' \rangle - c_d \langle z_2, z_2 \rangle \\
&\leq 0 \ .
\end{aligned}$$

Corollary 2.3.3. *([15, Cor 6.9]) For a closed operator $A : \mathcal{D}(A) \subset X \to X$, the associated abstract Cauchy problem is well-posed if and only if A generates a strongly continuous semigroup.*

Thus by Theorem 2.3 and Corollary 2.3.3, the state-space formulation generates a C_0 semigroup and is well-posed.

2.4 Eigenfunctions and Stability

Definition 2.4. (Eigenvalues and Eigenvectors) If there exists a scalar λ and a vector $\begin{bmatrix}\phi(x) & \Phi(x)\end{bmatrix}$ such that

$$A \begin{bmatrix} \phi(x) \\ \Phi(x) \end{bmatrix} = \lambda \begin{bmatrix} \phi(x) \\ \Phi(x) \end{bmatrix},$$

then λ is called the eigenvalue for the eigenvector $\begin{bmatrix}\phi(x) & \Phi(x)\end{bmatrix}^T$.

For simplicity consider the system form given in (2.26) which gives the following eigenvalue problem

$$\begin{bmatrix} 0 & I \\ c^2D^2 & -c_dI \end{bmatrix} \begin{bmatrix} \phi(x) \\ \Phi(x) \end{bmatrix} = \lambda \begin{bmatrix} I & 0 \\ 0 & I - \beta D^2 \end{bmatrix} \begin{bmatrix} \phi(x) \\ \Phi(x) \end{bmatrix}$$
$$\begin{bmatrix} \Phi(x) \\ c^2\phi''(x) - c_d\Phi(x) \end{bmatrix} = \begin{bmatrix} \lambda\phi(x) \\ \lambda(\Phi(x) - \beta\Phi''(x)) \end{bmatrix} \tag{2.29}$$

Taking $\Phi(x) = \lambda\phi(x)$ gives,

$$c^2\phi''(x) - c_d\lambda\phi(x) = \lambda^2\phi(x) - \beta\lambda^2\phi''(x)$$
$$\left[(c^2 + \beta\lambda^2)\frac{d^2}{dx^2} - (c_d\lambda + \lambda^2)\right]\phi(x) = 0 \tag{2.30}$$

Lemma 2.4.1. *The eigenfunctions $\phi(x)$ of*

$$\Gamma = \left[(c^2 + \beta\lambda^2)\frac{d^2}{dx^2} - (c_d\lambda + \lambda^2)\right], \tag{2.31}$$

with the boundary conditions $\phi(0) = \phi(L) = 0$ are

$$\phi_n(x) = \sin(mx), \ m = \frac{k\pi x}{L},$$

where k is any natural number.

Proof. Substituting $\phi(x)$ in (2.31) gives,

$$\Gamma(\phi(x)) = -(c^2 + \beta\lambda^2)m^2\phi(x) - (c_d\lambda + \lambda^2)\phi(x) \tag{2.32}$$
$$= -\left[(c^2 + \beta\lambda^2)m^2 + (c_d\lambda + \lambda^2)\right]\phi(x) \tag{2.33}$$
$$= \bar{\lambda}\phi(x) \tag{2.34}$$

where $\bar{\lambda}$ are the eigenvalues of Γ. □

Lemma 2.4.2. *The eigenfunctions and the corresponding eigenvalues of A are,*

$$\begin{bmatrix} \phi(x) \\ \Phi(x) \end{bmatrix} = \begin{bmatrix} \sin(mx) \\ \lambda\sin(mx) \end{bmatrix}, \quad \lambda_k = -\frac{c_d(m)}{2} \pm \frac{1}{2}i\sqrt{4m^2c(m)^2 - c_d(m)^2} \tag{2.35}$$

Proof. To calculate λ, use (2.30) to get,

$$(c^2 + \beta\lambda)m^2 + (c_d\lambda + \lambda^2) = 0,$$
$$\lambda^2 + c_d(m)\lambda + m^2c^2(m) = 0,$$

where,

$$c_d(m) = \frac{c_d}{1+\beta m^2} \ , \quad c(m)^2 = \frac{c^2}{1+\beta m^2} \ . \tag{2.36}$$

Therefore,

$$\lambda = -\frac{c_d(m)}{2} \pm \frac{1}{2} i \sqrt{4m^2c(m)^2 - c_d(m)^2}. \tag{2.37}$$

□

It can be conjectured that a finite-dimensional approximation of this system is asymptotically stable if $c_d > 0$. However, as $k \to \infty$, $\mathbb{R}(\lambda) \to 0$, which means the infinite-dimensional system will not be detectable/stabilizable [24]. The results in [33] show that this can lead to problems in designing controllers using a finite-dimensional approximation, and the same will be true for estimator design. Apart from this, there can be other affects of the dispersive and damping terms. A synopsis of three different cases is provided below.

Case 1 $\beta = 0$ and $c_d = 0$

This is the undamped non-dispersive wave equation which means all the eigenvalues are on the stability boundary (imaginary axis). This can be a problem numerically. In fact, the simulations verified this effect because with a high approximation order and no damping, the numerical scheme to solve the Algebraic Riccati Equation in MATLAB fails to be successful.

Case 2 $\beta = 0$ and $c_d > 0$

To represent the energy dissipation at the bottom boundary to viscosity, a damping term was added to the system. The addition of damping shifts the eigenvalues away from the imaginary axis in the stability region, which helps the numerical schemes to perform better.

Case 3 $\beta > 0$ and $c_d > 0$

Recall the expressions in (2.36) and (2.37),

$$c_d(m) = \frac{c_d}{1+\beta m^2} \ , \quad c(m)^2 = \frac{c^2}{1+\beta m^2} \ .$$

The presence of the dispersive model causes the speed of a wave to depend on the wave number (m). In this case the speed decreases with wave number which means the higher mode waves travel slower. The higher modes are also damped less according to the expression for $c_d(k)$. Moreover, when spatial discretization is carried out to form the finite-dimensional system, numerical dissipation is often also added to the system. The numerical dissipation will cause the higher modes to be eliminated before propagating far enough. Despite all these, however, the dispersive flow should not bring any additional challenges from a design point of view in finite dimensions.

Chapter 3

Approximation Methods

Consider the governing PDE which was derived in Chapter 2,

$$v_{tt} = c^2 v_{xx} - c_d v_t + \beta v_{xxtt}, \quad x \in \{\Omega = [0, L]\}, \tag{3.1}$$

$$v(0, x) = v_0 \quad v_t(0, x) = 0 \ . \tag{3.2}$$

Since a closed form solution is not always available (and never available if the nonlinear terms were retained), this system needs to be approximated. For the purposes of the present work this is best done by a reduction to a finite-dimensional system. Using the so–called Method of Lines, this is carried out by firstly assuming a separable solution exists and secondly by a spatial discretization method where the solution $v(x, t)$ is defined using a set of functions of the spatial variable, x, which are called the basis functions. This will lead to a representation of (3.1) by a finite number of ordinary differential equations which can subsequently be solved by an appropriate time discretization scheme, for instance the forward and backward Euler methods [14]. Once the time varying coefficients are obtained, the solution can be reconstructed and appropriate diagnostics (e.g. kinetic energy) can be obtained. In this chapter, three approximation methods will be discussed and used to derive a finite-dimensional representation of this equation.

3.1 Galerkin Methods

The standard numerical methods fall into three categories - finite difference methods based on Taylor series expansion, finite volume methods based on conservation laws and finite element methods, generally based on the Galerkin approximation [1, 40, 17]. Although,

finite difference methods are easy to construct, these methods are not ideal for complex geometries when compared to the more popular finite-volume and finite element methods. Finite-element methods have the further attraction of a well developed theoretical structure that allows for rigorous studies of convergence (though we note that both finite difference and finite volume methods have a rigorous theory as well), and hence a well defined finite-dimensional system representation that can subsequently employ the techniques of systems theory with confidence in its validity. In this work, the focus is on approximating the system by taking the Galerkin form and using three sets of basis functions- two coming from the standard finite element methods and the other being the set of eigenfunctions- to discretize the equation.

The Galerkin method finds a solution of the weak form (Galerkin form) of an abstract problem where the existence of the solution is guaranteed by the Lax-Milgram theory [1, see p. 118-119]. This problem is conventionally represented by,

$$a(u, v) = f(v), \tag{3.3}$$

where a gives a sesquilinear form between u and v, $u \subset V$ is a test function for the weak formulation and $v \subset V$ is the solution. The objective then is to find a solution v_n in a subspace $V_n \subset V$. The subscript n denotes the dimension of the subspace where the computed solution exists. This dimension reduction is carried out by projecting the functions on a finite-dimensional basis. The difference between v and v_n is orthogonal to the subspace V_n and hence v_n is called the optimal solution.

This method is explained in the following sections using a simple example ODE and then finally extended to construct the approximation of (3.1).

3.2 Approximating an example ODE

Example 3.1. Consider the following ordinary differential equation,

$$v''(x) + v(x) = f(x), \text{ on } \Omega = (0, L), \quad x \in \Omega \tag{3.4}$$
$$v(0) = v(L) = 0\ . \tag{3.5}$$

The weak formulation of the problem is,

$$\int_0^L v'u'dx + \int_0^L vudx = \int_0^L fudx, \tag{3.6}$$

where $u \in \mathcal{H}_0^1$ is a sample test function. Now the objective of the approximation is to find a finite-dimensional $u_n \in U_n$, where U_n is a subspace of $\mathcal{H}_0^1$.

Let $\{\phi_n = \phi_i \in \mathcal{C}^0(\Omega),\ i = 1, 2, \cdots, n\}$ be a set of functions such that the velocity v can expanded as,

$$v_n = \sum_{i=1}^{n} \langle v, \phi_i \rangle \phi_i. \tag{3.7}$$

Then $\Phi_n = \{\phi_1, \phi_2, ...\phi_N\}$ forms an n - dimensional basis and $span\{\Phi_n\}$ forms an n - dimensional subspace of $\mathcal{H}_0^1$. An approximate solution $v_n \in span\{\Phi_n\}$ for (3.5) can be given using the generalized Fourier coefficients $\langle v, \phi_i \rangle$.

Thus, the projection of (3.5) onto the n-dimensional subspace is,

$$\begin{aligned} \int_0^L \sum_{i=1}^{n} \langle v, \phi_i \rangle \phi_i' \sum_{i=1}^{n} \langle u, \phi_i \rangle \phi_i' dx + \int_0^L \sum_{i=1}^{n} \langle v, \phi_i \rangle \phi_i \sum_{i=1}^{n} \langle u, \phi_i \rangle \phi_i dx \\ = \int_0^L \sum_{i=1}^{n} \langle f, \phi_i \rangle \phi_i \sum_{i=1}^{n} \langle u, \phi_i \rangle \phi_i dx. \end{aligned} \tag{3.8}$$

These integrals can be written in a vector form. For instance,

$$\begin{aligned} &\int_0^L \sum_{i=1}^{n} \langle v, \phi_i \rangle \phi_i \sum_{i=1}^{n} \langle u, \phi_i \rangle \phi_i dx \\ &= \int_0^L \begin{bmatrix} a_1 & a_2 & \cdots & a_n \end{bmatrix} \begin{bmatrix} \phi_1 \\ \phi_2 \\ \vdots \\ \phi_n \end{bmatrix} \begin{bmatrix} c_1 & c_2 & \cdots & c_n \end{bmatrix} \begin{bmatrix} \phi_1 \\ \phi_2 \\ \vdots \\ \phi_n \end{bmatrix} dx \\ &= \begin{bmatrix} c_1 & c_2 & \cdots & c_n \end{bmatrix} \int_0^L \begin{bmatrix} \phi_1 \\ \phi_2 \\ \vdots \\ \phi_n \end{bmatrix} \begin{bmatrix} \phi_1 & \phi_2 & \cdots & \phi_n \end{bmatrix} dx \begin{bmatrix} a_1 \\ a_2 \\ \vdots \\ a_n \end{bmatrix}, \end{aligned}$$

where $a_i = \langle v, \phi_i \rangle$ and $c_i = \langle u, \phi_i \rangle$. Define the mass matrix, $M \in \mathbb{R}^{n \times n}$, and the stiffness matrix, $K \in \mathbb{R}^{n \times n}$, to be,

$$M = \int_0^L \begin{bmatrix} \phi_1 \\ \phi_2 \\ \vdots \\ \phi_n \end{bmatrix} \begin{bmatrix} \phi_1 & \phi_2 & \cdots & \phi_n \end{bmatrix} dx,$$

$$K = \int_0^L \begin{bmatrix} \phi_1' \\ \phi_2' \\ \vdots \\ \phi_n' \end{bmatrix} \begin{bmatrix} \phi_1' & \phi_2' & \cdots & \phi_n' \end{bmatrix} dx.$$

Define the vectors,

$$\begin{aligned} \mathbf{a} &= \{a_i \in \mathbb{R}, a_i = \langle v, \phi_i \rangle, i = 1...n\}, \\ \mathbf{b} &= \{b_i \in \mathbb{R}, b_i = \langle f, \phi_i \rangle, i = 1...n\}, \\ \mathbf{c} &= \{c_i \in \mathbb{R}, c_i = \langle u, \phi_i \rangle, i = 1...n\}. \end{aligned}$$

Noting that the vector $\mathbf{c}$ is a common factor, (3.8) can be simplified and represented using matrices as,

$$(K + M)a = Mb, \tag{3.9}$$

which is a linear system of equations and can be solved for a.

Now continue to work with Example 3.1 in order to compute the mass and stiffness matrices using finite element methods. Two different bases will be considered; a set of linear functions and the shape functions based on Legendre polynomials.

The finite element method involves discretizing the spatial domain into a finite number of elements and constructing a basis on each of these elements in order to approximate the solution.

To begin with, divide the spatial domain equally into J elements and construct the Galerkin form for (3.5) over an element

$$\int_{x_{j-1}}^{x_j} v'u'dx + \int_{x_{j-1}}^{x_j} vudx = \int_{x_{j-1}}^{x_j} fudx. \tag{3.10}$$

Now calculate ϕ of (3.7) from a set of basis functions. These functions are constructed on a canonical element,

$$I_e = \{\xi : \xi \in [-1, 1]\},$$

which can be mapped to $[x_j, x_{j-1}]$ by an invertible transformation. Thus the integral in (3.10) is first mapped to the canonical element using the transformation,

$$x(\xi) = \frac{1 - \xi}{2} x_{j-1} + \frac{1 + \xi}{2} x_j,$$

to give,

$$\frac{2J}{L}\int_{-1}^{1}\frac{dv}{d\xi}\frac{du}{d\xi}d\xi+\frac{L}{2J}\int_{-1}^{1}vud\xi=\frac{L}{2J}\int_{-1}^{1}fud\xi. \tag{3.11}$$

Note that because of this transformation, the construction of the finite element method now requires two easy steps:

1. Constructing the local mass and stiffness matrices (m and k) using the particular choice of basis on the canonical element
2. Using these local matrices to form the global mass and stiffness matrices (M and K)

3.2.1 Mass and Stiffness Matrix from Linear Basis

The linear basis functions are,

$$\phi_1=\frac{1}{2}(1-\xi)\ \ ,\ \phi_2=\frac{1}{2}(1+\xi)\ . \tag{3.12}$$

and therefore the approximate solution in each element can be written as,

$$\hat{v}(\xi)=a_1\phi_1(\xi)+a_2\phi_2(\xi). \tag{3.13}$$

Substituting (3.13) and the projection of the rest of the functions in (3.11) gives the finite-dimensional representation,

$$\begin{aligned}\frac{2J}{L}\int_0^1\begin{bmatrix}a_1 & a_2\end{bmatrix}\begin{bmatrix}\phi_1'\\ \phi_2'\end{bmatrix}\begin{bmatrix}c_1 & c_2\end{bmatrix}\begin{bmatrix}\phi_1'\\ \phi_2'\end{bmatrix}d\xi+\frac{L}{2J}\int_0^1\begin{bmatrix}a_1 & a_2\end{bmatrix}\begin{bmatrix}\phi_1\\ \phi_2\end{bmatrix}\begin{bmatrix}c_1 & c_2\end{bmatrix}\begin{bmatrix}\phi_1\\ \phi_2\end{bmatrix}d\xi\\ =\frac{L}{2J}\int_0^1\begin{bmatrix}b_1 & b_2\end{bmatrix}\begin{bmatrix}\phi_1\\ \phi_2\end{bmatrix}\begin{bmatrix}c_1 & c_2\end{bmatrix}\begin{bmatrix}\phi_1\\ \phi_2\end{bmatrix}d\xi,\end{aligned}$$

which can be restructured to give

$$\begin{aligned}\frac{2J}{L}\begin{bmatrix}c_1 & c_2\end{bmatrix}\int_0^1\begin{bmatrix}\phi_1'\\ \phi_2'\end{bmatrix}\begin{bmatrix}\phi_1' & \phi_2'\end{bmatrix}d\xi\begin{bmatrix}a_1\\ a_2\end{bmatrix}+\frac{L}{2J}\begin{bmatrix}c_1 & c_2\end{bmatrix}\int_0^1\begin{bmatrix}\phi_1\\ \phi_2\end{bmatrix}\begin{bmatrix}\phi_1 & \phi_2\end{bmatrix}d\xi\begin{bmatrix}a_1\\ a_2\end{bmatrix}\\ =\frac{L}{2J}\begin{bmatrix}c_1 & c_2\end{bmatrix}\int_0^1\begin{bmatrix}\phi_1\\ \phi_2\end{bmatrix}\begin{bmatrix}\phi_1 & \phi_2\end{bmatrix}d\xi\begin{bmatrix}a_1\\ a_2\end{bmatrix}.\end{aligned}$$

The local mass and stiffness matrices are then computed as follows,

$$\begin{aligned} m &= \int_0^1 \begin{bmatrix} \phi_1 \\ \phi_2 \end{bmatrix} \begin{bmatrix} \phi_1 & \phi_2 \end{bmatrix} d\xi \\ &= \int_0^1 \begin{bmatrix} \phi_1\phi_1 & \phi_1\phi_2 \\ \phi_2\phi_1 & \phi_2\phi_2 \end{bmatrix} d\xi \\ &= \begin{bmatrix} 2/3 & 5/6 \\ 5/6 & 2/3 \end{bmatrix}, \end{aligned}$$

$$\begin{aligned} k &= \int_0^1 \begin{bmatrix} \phi_1' \\ \phi_2' \end{bmatrix} \begin{bmatrix} \phi_1' & \phi_2' \end{bmatrix} d\xi \\ &= \int_0^1 \begin{bmatrix} \phi_1'\phi_1' & \phi_1'\phi_2' \\ \phi_2'\phi_1' & \phi_2'\phi_2' \end{bmatrix} d\xi \\ &= \begin{bmatrix} 1/2 & -1/2 \\ -1/2 & 1/2 \end{bmatrix}. \end{aligned}$$

The next step is to use these local matrices to build the global matrices. This will be carried out in two simpler steps to make things clear. Firstly, place the local mass matrix in an empty global mass matrix starting from the first element to the last. For simplicity assume there are two elements spanning $[0, L]$. The coefficients can be denoted such that the contributions coming from different elements are distinguished, but ignored here for neatness. Then,

$$Ma = \begin{bmatrix} 2/3 & 5/6 & & \\ 5/6 & 2/3 & & \\ & & 2/3 & 5/6 \\ & & 5/6 & 2/3 \end{bmatrix} \begin{bmatrix} a_1 \\ a_2^* \\ a_1^* \\ a_2 \end{bmatrix}.$$

See Figure 3.1 and note the continuity between the elements. Here a_1^* and a_2^* are essentially the same coefficients. Considering this the global mass matrix with two elements is given by,

$$\begin{bmatrix} 2/3 & 5/6 & \\ 5/6 & 2/3 + 2/3 & 5/6 \\ & 5/6 & 2/3 \end{bmatrix}$$

The stiffness matrix can be constructed following the same steps. Note that the required boundary conditions are not reflected in these matrices yet. For the zero Dirichlet boundary

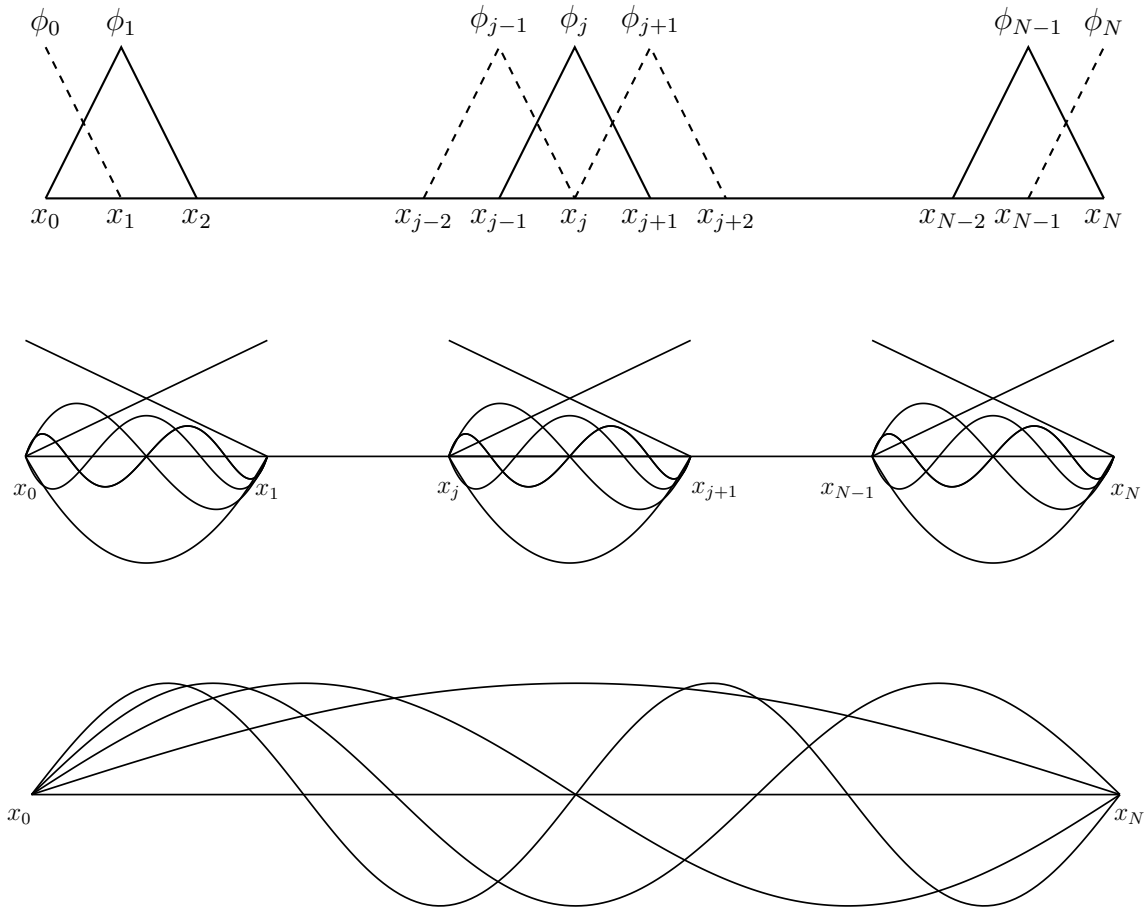

Figure 3.1: A visual representation of the different bases. The finite element methods increase the degree of freedom by increasing number of elements while the eigenfunction increase by adding more modes.

conditions, the 'picture frame' row and columns of the matrix would be truncated such that it becomes,

$$\begin{bmatrix} 2/3 & & \\ & 2/3+2/3 & \\ & & 2/3 \end{bmatrix}.$$

The terms $\frac{2J}{L}$ and $\frac{L}{2J}$ can be absorbed into these matrices to give,

$$(K + M)a = Mb.$$

3.2.2 Mass and Stiffness Matrix from Polynomial Basis

The basis is in this section will be constructed using the first six Legendre polynomials,

$$\begin{aligned}
P_0(x) &= 1 \\
P_1(x) &= x \\
P_2(x) &= \frac{3x^2 - 1}{2} \\
P_3(x) &= \frac{5x^3 - 3x}{2} \\
P_4(x) &= \frac{35x^4 - 30x^2 + 3}{2} \\
P_5(x) &= \frac{63x^5 - 70x^3 + 15x}{8},
\end{aligned}$$

As before, these functions are restricted to an element by the so-called shape functions,

$$N_i(\xi) = \frac{P_i(\xi) - P_{i-2}(\xi)}{\sqrt{2(2i-1)}} \ , \ i \geq 2 \ . \tag{3.14}$$

The linear basis functions are also included as the shape functions for the approximation

$$N_{-1} = \frac{1}{2}(1 - \xi) \quad N_1 = \frac{1}{2}(1 + \xi) \ .$$

A canonical element with these shape functions is shown in Figure 3.1. The approximate solution $\hat{u}$ of (3.10) over the canonical element can be written as

$$\hat{v}(\xi) = a_{-1}N_{-1}(\xi) + a_1 N_1(\xi) + \sum_{i=2}^{5} a_i N_i(\xi) \tag{3.15}$$

and the approximation $\hat{v}$ for the test function v as

$$\hat{u}(\xi) = c_{-1}N_{-1}(\xi) + c_1 N_1(\xi) + \sum_{i=2}^{5} c_i N_i(\xi) \ . \tag{3.16}$$

Similarly to the case of the linear basis, progress from this stage requires two further steps to complete the construction; compute the local mass and stiffness matrices and extend these to the global matrices.

However, for the polynomial basis a specific ordering of the basis functions needs to be set such that the inner products are placed in the matrices using the correct indices. Here the ordering used is $\{N_{-1}\ N_2\ N_3\ N_4\ N_5\ N_1\}$, which implies that the approximate solution, $\hat{v}$, is given by,

$$\hat{v} = \begin{bmatrix} a_{-1} & a_2 & a_3 & a_4 & a_5 & a_1 \end{bmatrix} \begin{bmatrix} N_{-1} \\ N_2 \\ N_3 \\ N_4 \\ N_5 \\ N_1 \end{bmatrix}, \tag{3.17}$$

over the canonical element. Also, the local mass and stiffness matrices are given by,

$$m = \int_{-1}^{1} \begin{bmatrix} N_{-1} \\ N_2 \\ N_3 \\ N_4 \\ N_5 \\ N_1 \end{bmatrix} \begin{bmatrix} N_{-1} & N_2 & N_3 & N_4 & N_5 & N_1 \end{bmatrix} d\xi,$$

$$k = \int_{-1}^{1} \begin{bmatrix} N'_{-1} \\ N'_2 \\ N'_3 \\ N'_4 \\ N'_5 \\ N'_1 \end{bmatrix} \begin{bmatrix} N'_{-1} & N'_2 & N'_3 & N'_4 & N'_5 & N'_1 \end{bmatrix} d\xi.$$

Comparing with the representation introduced in Section 3.1, it can be noted that $\phi_1 = N_{-1},\ \phi_2 = N_2,\ \phi_3 = N_3,\ \phi_4 = N_4,\ \phi_1 = N_5,\ \phi_6 = N_1$, for each element in the finite element method. Therefore let $m_{ij} = \langle \phi_i, \phi_j \rangle$ and $k_{ij} = \langle \phi'_i, \phi'_j \rangle$.

For the finite element method using the polynomial basis, the continuity again is provided by the linear functions. Hence the last step is similar to that of Section 3.2.1.

Following these steps for a two-element case would give the following mass matrix,

$$\begin{bmatrix} m_{1,1} & m_{1,2} & \cdots & m_{1,5} & m_{1,6} & & & & & \\ m_{2,1} & m_{2,2} & \cdots & m_{2,5} & m_{2,6} & & & & & \\ \vdots & \vdots & \ddots & \vdots & m_{1,6} & & & & & \\ m_{6,1} & m_{6,2} & \cdots & m_{6,5} & m_{6,6}+m_{1,1} & & & & & \\ & & & & m_{2,1} & m_{2,2} & \cdots & m_{2,5} & m_{2,6} & \\ & & & & \vdots & \vdots & \ddots & \vdots & m_{1,6} & \\ & & & & m_{6,1} & m_{6,2} & \cdots & m_{6,5} & m_{6,6}+m_{1,1} \end{bmatrix}$$

The stiffness matrix, K can be constructed by following a similar set of steps.

3.3 Approximating the PDE

The sample ODE (3.5) was approximated to provide a description of the Galerkin approximation using two sets of finite element basis functions. In this section the approximation of the PDE (3.1) will be derived with reference to these steps.

First construct the weak form by multiplying (3.1) by a test function, $u \in H_0^1(\Omega)$, and integrate by parts to give,

$$\begin{aligned} \int_\Omega v_{tt} u dx &= c^2 \int_\Omega v_{xx} u dx + \beta \int_\Omega v_{xxtt} u dx - c_d \int_\Omega v_t u dx \\ &= -c^2 \int_\Omega v_x u_x dx - \beta \int_\Omega v_{xtt} u_x dx - c_d \int_\Omega v_t u dx. \end{aligned} \tag{3.18}$$

Let $\{\phi_i \in C^0(\Omega),\ i = 1, 2, \cdots, n\}$ be a set of functions such that the set $\Phi_n = \{\phi_1, \phi_2, ...\phi_n\}$ forms an n - dimensional basis and the $span\{\Phi_n\}$ forms an n - dimensional subspace of $\mathcal{H}_0^1$. A solution $v_n \in span\{\Phi_n\}$ for (3.18) can thus be given using the Fourier coefficients as

$$v_n = \sum_{i=1}^{n} \langle v, \phi_i \rangle \phi_i. \tag{3.19}$$

Recall the definitions of $M = \{M \in \mathbb{R}^{n\times n} : M_{ij} = \langle \phi_i, \phi_j \rangle\}$, $K = \{K \in \mathbb{R}^{n\times n} : K_{ij} = \langle \phi_i', \phi_j' \rangle\}$ and $a = \{a \in \mathbb{R}^{n\times 1} : a_i = \langle v, \phi_i \rangle\}$.Then a finite-dimensional representation of

(3.18) is,

$$\begin{aligned}\ddot{a}_j \int_\Omega \phi_i \phi_j dx =& -c^2 a_j \int_\Omega \phi_i' \phi_j' - \beta \ddot{a}_j \int_\Omega \phi_i' \phi_j' dx - c_d \dot{a}_j \int_\Omega \phi_i \phi_j dx \\ (M + \beta K)\ddot{a} =& -c^2 K a - c_d M \dot{a}. \end{aligned} \tag{3.20}$$

It now remains to compute these mass and stiffness matrices using

1. Finite element method with either the linear basis or the 6th order polynomial basis
2. Eigenfunctions of (2.24)

3.3.1 Approximation using Finite Element Method

Note that the construction of mass and stiffness matrices can be computed following the steps in the ODE example exactly:

1. Divide the domain into J equal elements
2. Transform (3.18) onto the canonical element
3. Compute the local mass and stiffness matrices
4. Use the local mass and stiffness matrices to build the global matrices

3.3.2 Approximation using Eigenfunctions

The approximation using eigenfunctions does not fall in the category of finite-element methods (though can be thought of as a Galerkin method). To construct the finite-dimensional approximation using the eigenfunctions, substitute

$$\phi_j = \sin(\frac{\pi j x}{L})$$

in (3.20). Since the eigenfunctions are orthogonal, i.e.,

$$\langle \phi_i, \phi_j \rangle = 0, \text{ if } i \neq j \ ,$$

the mass and stiffness matrices will be diagonal matrices. Each entry in the diagonal will simply be $K_{ii} = \langle \phi_i, \phi_i \rangle$ and $M_{ii} = \langle \phi_i', \phi_i' \rangle$. The use of eigenfunctions is basically a spectral method approach, which under certain circumstances can lead to high level of accuracy with even small degree of freedom [7, 43] .

3.3.3 Results

In this section the finite element methods will be compared with each other to show why two different order of basis was used for contructing the basis. To validate these comparison, the partial differential equation was solved. In order to compare these different discretization methods in computations, the number of degrees of freedom was fixed. For the continuous Galerkin finite element method, the degree of freedom is $n = p \times J + 1$, where $p = 1$ for the linear basis and $p = 5$ for the polynomial basis, and J is the number of elements. For the sine basis, it is the number of sine modes. The initial condition was,

$$\begin{aligned} v(x,0) &= \operatorname{sech}\left(\frac{J}{2}\left(x - \frac{J}{2}\right)\right) \\ v_t(x,0) &= 0. \end{aligned}$$

The number of degrees of freedom was set to $n = 151$ for the finite element methods. The solution was first computed using the sine basis with a degree of freedom set to $10n$ and then the solutions obtained using the finite element methods were compared to this reference solution. As Figure 3.2 shows, the finite element methods perform almost with similar accuracy for this initial condition.

However, to demonstrate the advantage using a higher order basis for the finite element method, consider the following initial conditions,

$$\begin{aligned} v(x,0) &= \operatorname{sech}\left(\frac{J}{2}\left(x - \frac{J}{2}\right)\right)\sin\left(\frac{2\pi x}{.5}\right) \\ v_t(x,0) &= 0. \end{aligned}$$

From Figure 3.3, it is clear that the finite element method with 6^{th}-order polynomial basis is significantly more accurate than the one with linear basis.

3.4 Summary

In this chapter, three finite-dimensional approximations of the original infinite-dimensional system was constructed which leads to the following expression

$$(M + \beta K)\ddot{a} = -c^2 Ka - c_d M\dot{a} + Mb. \tag{3.21}$$

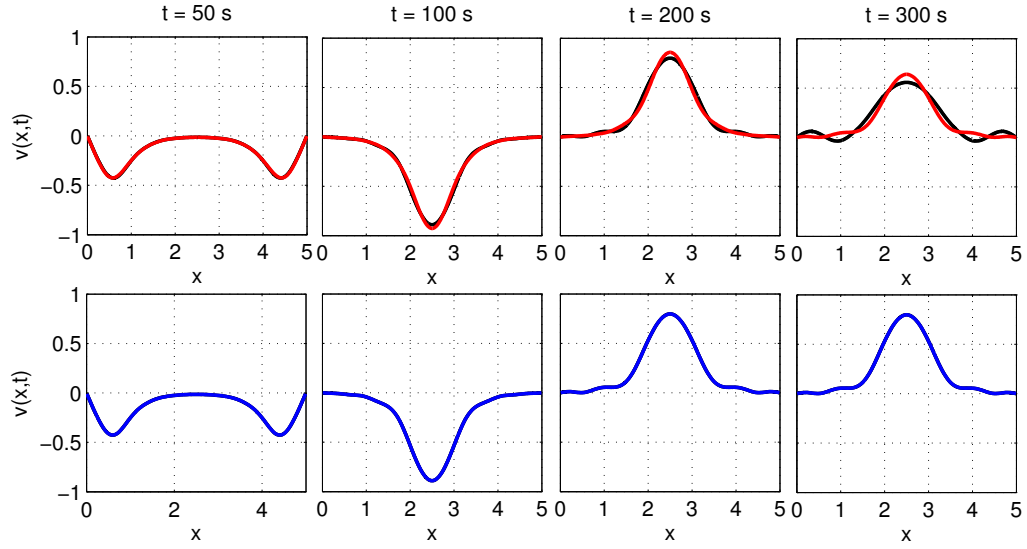

Figure 3.2: The black line represents the solution obtained using the sine basis and a very high degree of freedom (n=400). The red and blue lines correspond to solutions obtained using the finite element methods with linear and 6^{th}-order basis respectively, and $n = 151$.

Three sets of $\{M, K\}$ were derived using three different choice of basis Φ_n,

$$\Phi_n^{sine} = \begin{bmatrix} \sin(1\pi\hat{x}) \\ \sin(2\pi\hat{x}) \\ \sin(3\pi\hat{x}) \\ \vdots \\ \sin(n\pi\hat{x} \end{bmatrix} \quad \Phi_n^{linear} = \begin{bmatrix} 0 \\ \frac{1}{2}(1+\xi) \\ \frac{1}{2}(1+\xi) \\ \vdots \\ 0 \end{bmatrix} \quad \Phi_n^{6^{th}order} = \begin{bmatrix} 0 \\ N_2 \\ N_3 \\ N_4 \\ N_5 \\ \hline N_1 \\ N_2 \\ N_3 \\ N_4 \\ N_5 \\ \hline \vdots \\ \vdots \\ \hline N_1 \\ N_2 \\ N_3 \\ N_4 \\ N_5 \\ 0 \end{bmatrix} \begin{matrix} \left.\vphantom{\begin{matrix}0\\N_2\\N_3\\N_4\\N_5\end{matrix}}\right\} 1^{st} \text{ element} \\ \left.\vphantom{\begin{matrix}N_1\\N_2\\N_3\\N_4\\N_5\end{matrix}}\right\} 2^{nd} \text{ element} \\ \left.\vphantom{\begin{matrix}N_1\\N_2\\N_3\\N_4\\N_5\\0\end{matrix}}\right\} J^{th} \text{ element} \end{matrix} .$$

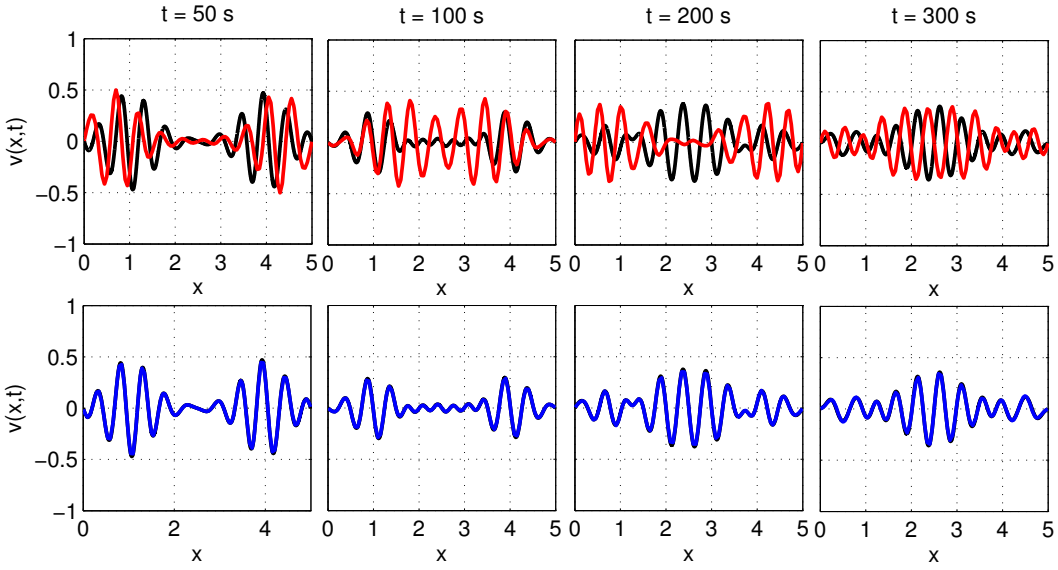

Figure 3.3: Comparison of solutions obtained using the three methods with a new initial condition. The black line represents the solution obtained using the sine basis and a very high degree of freedom. The red and blue lines correspond to solutions obtained using the finite element method with linear and 6^{th}-order basis respectively, and $n = 151$.

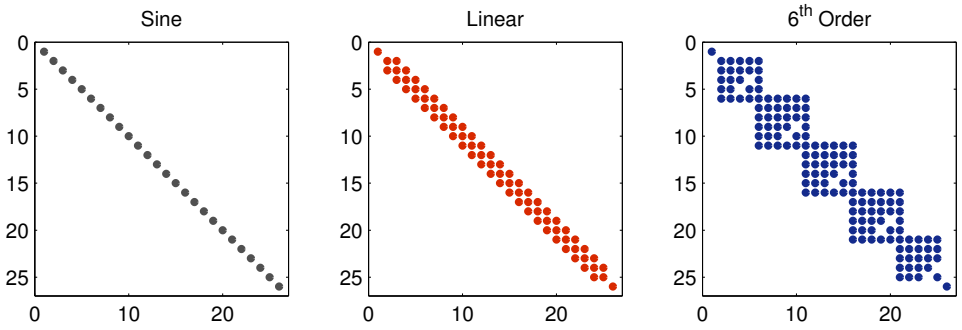

Figure 3.4: Mass matrix structure from the three approximation methods.

Write (3.21) in a descriptor form to give,

$$\begin{bmatrix} I & 0 \\ 0 & M + \beta K \end{bmatrix} \begin{bmatrix} \dot{a} \\ a_t \end{bmatrix} = \begin{bmatrix} 0 & I \\ -c^2 K & -c_d M \end{bmatrix} \begin{bmatrix} a \\ a_t \end{bmatrix}. \tag{3.22}$$

This is a finite-dimensional approximation of the PDE (Chapter 2, Equation 2.26),

$$\begin{bmatrix} I & 0 \\ 0 & I - \beta D^2 \end{bmatrix} \begin{bmatrix} \dot{v} \\ \dot{v_t} \end{bmatrix} = \begin{bmatrix} 0 & I \\ c^2 D^2 & -c_d I \end{bmatrix} \begin{bmatrix} v \\ v_t \end{bmatrix}. \tag{3.23}$$

Unlike (3.23), (3.22) can used to design controllers and estimator for the system. Whether the design in finite dimensions would converge to the infinite-dimensional system is the subject of many research topics [33, 36]. The convergence of the estimator is investigated in [20]. In the current work, the focus will be on the comparisons of an estimator designed using these three approximation methods.

Chapter 4

Optimal State Estimation

4.1 Linear Time Invariant Systems

Consider the following abstract form of a system in finite dimensions,

$$\begin{aligned} \dot{z}(t) &= Az(t) + Bu(t), \qquad z(0) = z_0, \\ y(t) &= Cz(t), \end{aligned} \tag{4.1}$$

on the state space Z with the following description,

- state vector: $z \in \mathbb{R}^{n\times 1}$
- state matrix: $A \in \mathbb{R}^{n\times n}$,
- input matrix: $B \in \mathbb{R}^{n\times 1}$
- input: $u \in \mathbb{R}^{1\times 1}$
- observation matrix: $C \in \mathbb{R}^{1\times n}$
- measurement: $y \in \mathbb{R}^{1\times 1}$.

The solution to the system in (4.1) is [8],

$$z(t) = z_0 e^{At} + \int_0^t e^{A(t-\tau)} Bu(\tau) d\tau. \tag{4.2}$$

Definition 4.1. The matrix A is Hurwitz when all the eigenvalues of A have negative real part.

Theorem 4.2. *The system in (4.1) is internally stable when A is Hurwitz.*

Definition 4.3. The pair is (A, C) detectable if there exists an $F \in \mathbb{R}^{n\times 1}$ such that, $(A - FC)$ is Hurwitz.

4.2 Estimation

Definition 4.4. An estimate $\hat{z}(t)$ of the states $z(t)$ in (4.1) is such that, for any choice of initial condition $\hat{z}(0)$

$$\lim_{t\to\infty} ||\hat{z}(t) - z(t)|| = 0 \ . \tag{4.3}$$

If the pair (A, C) is detectable this can be achieved by creating the following auxiliary system,

$$\dot{\hat{z}}(t) = A\hat{z}(t) + F(\hat{y} - y), \quad \hat{z}(0) = \hat{z}_0. \tag{4.4}$$

$$\hat{y} = C\hat{z} \tag{4.5}$$

The estimator error is,

$$e(t) = z(t) - \hat{z}(t), \tag{4.6}$$

and

$$\dot{e}(t) = \dot{z}(t) - \dot{\hat{z}}(t) \tag{4.7}$$

$$= (A - FC)e(t). \tag{4.8}$$

The error will converge to zero if F is choosen so that $(A-FC)$ is Hurwitz. This shows an entire class of state estimators which has a special name called the Luenberger Observer. Some approaches to select F lead to optimization problems, where F minimizes a certain cost function.

4.3 Steady-State Continuous-Time Kalman Filter

A steady-state continuous-time Kalman filter will be derived in this section. First, a general formulation is provided and then a model specific formulation is given where the descriptor form of the original system is used.

4.3.1 General Formulation

Consider the following linear system,

$$\begin{aligned} \dot{z}(t) &= Az(t) + Gw_1(t), \qquad z(0) = z_0, \\ y(t) &= Cz(t) + w_2(t). \end{aligned} \tag{4.9}$$

Here $w_1(t)$ and $w_2(t)$ represent the process noise and the measurement noise respectively. The process noise in this system can be referred to the natural phenomenon which affects the actual state of the system. Thus, a noise term is included in the governing partial differential equation which leads to the above representation. Measurement noise refers to the noise which is present in the sensor measurements and therefore effects the estimation. The noise elements are assumed to be zero-mean white (uncorrelated) Gaussian random variables with Q and R being the covariance matrices of respective noise elements. Here, R is positive definite and Q is positive semi-definite. It can be noted that many naturally occurring noise processes are in fact uncorrelated, for example blowing wind, and such an assumption is based on physical basis.

The original discrete-time filter is formulated in [25]. A continuous time Kalman filter is derived in [42] and [28] which is represented by,

$$\begin{aligned} \dot{x}(t) &= Ax(t) + F(t)(y(t) - Cx(t)), \\ &= (A - F(t)C)x(t) + F(t)y(t), \quad x(0) = x_0, \end{aligned} \tag{4.10}$$

where F is the Kalman gain given by,

$$F(t) = P(t)C^T R^{-1}, \tag{4.11}$$

and,

$$\dot{P}(t) = AP(t) + P(t)A^T - P(t)C^T R^{-1} CP(t) + GQG. \tag{4.12}$$

The gain $F(t)$ minimizes the following cost functional,

$$J_e = \int_0^{t_f} \mathbf{E}[(z - x)^T (z - x)]dt. \tag{4.13}$$

Note that the system in (4.9) is an LTI system with stationary noise, i.e the matrices A, C, G, Q and R are constants. This implies that under certain conditions a steady state Kalman filter can be obtained and the Riccati equation in (4.12) becomes,

$$0 = AP + PA^T - PC^T R^{-1} CP + GQG. \tag{4.14}$$

Theorem 4.5. *(([42, Thm 27])) Let $CC^T = Q$. If the pair (A,C) is detectable and the pair (A,Q) is stabilizable there exists a unique positive semidefinite solution P to (4.14). This leads to a stabilizing Kalman filter gain, that is the matrix $(A - FC)$ in (4.10) is Hurwitz.*

The Kalman filter is also referred to as the minimum variance filter. The cost minimized by a steady state Kalman filter is,

$$J_e(z_0) = \lim_{t\to\infty} \mathbf{E}\{(z-x)(z-x)^T\} \ , \tag{4.15}$$

where $\mathbf{E}$ is the expectation. The minimum cost J over all the estimates $\hat{z}$ is

$$\min_{\hat{z}} J_e = P \ ,$$

where P is the positive semidefinite solution to (4.14). This cost function will be a subject of discussion in the next chapter, where the requirement would be to define a scalar cost function so that an optimal sensor location problem can be formulated.

Example 4.6 (A steady state Kalman filter)**.** The following example is taken from [3, Chapter 5]. Consider,

$$\dot{z} = \begin{bmatrix} a_1 & 0 \\ 0 & a_2 \end{bmatrix} z + w \tag{4.16}$$

$$y = \begin{bmatrix} 1 & 0 \end{bmatrix} + v \tag{4.17}$$

$$Q = \begin{bmatrix} 1 & 0 \\ 0 & 0 \end{bmatrix} \tag{4.18}$$

$$R = 1 \tag{4.19}$$

Note that the eigenvalues of this systems are a_1 and a_2. If $a_2 < 0$, then (A, C) is detectable and (A, Q') is stabillizable. For instance, let $a_1 = 2$ and $a_2 = -3$. Then the positive semi-definite solution to (4.14) is,

$$P = \begin{bmatrix} 2+\sqrt{5} & 0 \\ 0 & 0 \end{bmatrix}. \tag{4.20}$$

The steady state Kalman gain is,

$$\begin{aligned} F &= \begin{bmatrix} 2+\sqrt{5} & 0 \\ 0 & 0 \end{bmatrix} \begin{bmatrix} 1 \\ 0 \end{bmatrix} 1 \\ &= \begin{bmatrix} 2+\sqrt{5} \\ 0 \end{bmatrix} \\ A - FC &= \begin{bmatrix} 2 & 0 \\ 0 & -3 \end{bmatrix} - \begin{bmatrix} 2+\sqrt{5} & 0 \\ 0 & 0 \end{bmatrix} \\ \lambda(A - FC) &= \{-\sqrt{5}, -3\} \end{aligned}$$

Recall (4.9) and (4.10), and let the error in estimation be $\dot{\xi}(t) = \dot{z}(t) - \dot{x}(t)$. Then,

$$\dot{\xi}(t) = \begin{bmatrix} -\sqrt{5} & 0 \\ 0 & -3 \end{bmatrix} \xi(t) + f(t),$$

where $f(t) = w(t) - Fv(t)$. The solution to this ODE can be given as,

$$\begin{aligned} \xi(t) &= \xi(0)\mathrm{e}^{(A-FC)t} + \dot{f}(t), \\ \text{where } \xi(t) &\to \dot{f}(t), \text{ as } t \to \infty \, . \end{aligned}$$

The function $\dot{f}(t)$ is the residual error and depends on the noise characterisitcs w and v, see [28, pp. 163–165].

4.3.2 Model Specific Formulation

In the previous section, a general formulation of the Kalman filter was provided. Since the original system for this work is in a descriptor form, and the need to invert matrices during computations should be avoided, the Kalman filter is now given for a descriptor form of a dynamical system. Recall from Section 3.4, the approximation of the original system in a descriptor form,

$$\begin{bmatrix} I & 0 \\ 0 & M + \beta K \end{bmatrix} \begin{bmatrix} \dot{a} \\ a_t \end{bmatrix} = \begin{bmatrix} 0 & I \\ -c^2 K & -c_d M \end{bmatrix} \begin{bmatrix} a \\ a_t \end{bmatrix} . \tag{4.21}$$

Define,

$$A_1 = \begin{bmatrix} I & 0 \\ 0 & M + \beta K \end{bmatrix}, A_2 = \begin{bmatrix} 0 & I \\ -c^2 K & -c_d M \end{bmatrix}, z = \begin{bmatrix} a \\ a_t \end{bmatrix} .$$

Note that the real part of the eigenvalues of the finite-dimensional system are all negative due to the presence of damping. Thus, there does not exist any unstable modes and the approximation satisfies the detectability and stabilizability conditions required for the existence of a steady state filter.

Thus the Algebraic Riccati equation is,

$$A_2 P A_1^T + A_1 P A_2^T - A_1 P C' R^{-1} C P A_1^T + G Q G^T = 0 \tag{4.22}$$

and the solution to (4.22) is written as,

$$P = \begin{bmatrix} P_1 & P_3 \\ P_3 & P_4 \end{bmatrix}.$$

Defining $F = A_1 P C^T R^{-1}$, the optimal estimator is,

$$A_1 \frac{d}{dt} \hat{z}(t) = A_2 \hat{z}(t) - F(C\hat{z}(t) - y(t)) \ . \tag{4.23}$$

In this case, F has two components, corresponding to the two components of the state z

$$\begin{aligned} F = \begin{bmatrix} F_1 \\ F_2 \end{bmatrix} &= A_1 P C^T R^{-1} \\ &= \begin{bmatrix} I & 0 \\ 0 & M + \beta K \end{bmatrix} \begin{bmatrix} P_1 & P_3 \\ P_3 & P_4 \end{bmatrix} \begin{bmatrix} C_1 \\ C_2 \end{bmatrix} R^{-1} \\ &= \begin{bmatrix} P_1 C_1 \\ [M + \beta K] P_3 C_1 \end{bmatrix} R^{-1} . \end{aligned}$$

Chapter 5

Optimal Sensor Location

In this chapter the optimal sensor location problem will be formulated. A description of the sensor and the noise assumed are provided. Then some simulations are carried out to show the convergence of the estimator, cost functions with respect to the sensor locations and other important parameters.

The objective of the optimal sensor location problem is to find the optimal locations for the sensor with respect to the cost function,

$$J_e(z_0) = \lim_{t\to\infty} \mathbf{E}\{(z-x)(z-x)^T\} \ . \tag{5.1}$$

The solution to the Riccati equation (4.22) depends on the sensor C, which in turn depends on the sensor location ℓ. For each location there is an associated minimal cost $P(\ell)$. Note that P is a symmetric positive definite matrix which represents the error covariance in the case of Kalman filter. In order to formulate an optimal sensor location problem, a scalar cost function that is representative of P would be more suitable. If the initial condition is assumed to be random with zero mean then the nuclear norm, or the trace, of P is an appropriate cost. See [34] where the dual problem is considered. It is easy to see that the trace indeed would be an appropriate cost function. Recall the definition of P in (5.1) and rewrite as

$$P_{ij} = \lim_{t\to\infty} \mathbf{E}[(z_i - \hat{z}_i)(z_j - \hat{z}_j)].$$

If the expected error is bounded then the trace is,

$$\begin{aligned} tr(P) &= \sum_{i=1}^{n} P_{ii} \\ &= \sum_{i=1}^{n} \mathbf{E}[(z_i - \hat{z}_i)(z_i - \hat{z}_i)] \\ &= \lim_{t\to\infty} \mathbf{E}\left[\sum_{i=1}^{n}(z_i - \hat{z}_i)(z_i - \hat{z}_i)\right] \\ &= \lim_{t\to\infty} \mathbf{E}[(z - \hat{z})^T(z - \hat{z})] \\ &= J_c. \end{aligned}$$

Definition 5.1 (Optimal Sensor Location)**.** An optimal sensor location is an element of the set $\ell \in [0, L]$ such that $J_c(\ell)$ is minimal.

5.1 Description of the Sensor

A wide variety of measurement techniques are applied in the field, often with their own set of technique specific assumptions. For example, while wave height data are of very high importance for various engineering purposes (e.g. wave stresses on harbour walls), they are difficult to measure directly [19, 38]. In practice, significant wave height is measured, which is the average of the highest one-third of waves [38]. Velocity measurements can also be obtained by direct methods, though modern observational towers often employ indirect methods such as acoustic Doppler (i.e. ADCPs). In this project, for simplicity, the following form for the velocity measurement is assumed,

$$y(t) = \int_0^L f_l(x)v(x,t)dx, \quad l \in [0, L] \tag{5.2}$$

where $f_l(x) = \text{sech}(15(x - l))$ models the sensor. Let $\Pi \in \mathbb{R}^n$ be the projection of $f_l(x)$ onto the n dimensional basis Φ_n. Then, the finite-dimensional sensor is,

$$\begin{aligned} \int_0^L f_l(x)v(x,t)dx &\approx \int_0^L (\Pi\Phi_n)(a^T\Phi_n)dx \\ &= \Pi\left(\int_0^L \Phi_n\Phi_n^T dx\right) a \ . \end{aligned} \tag{5.3}$$

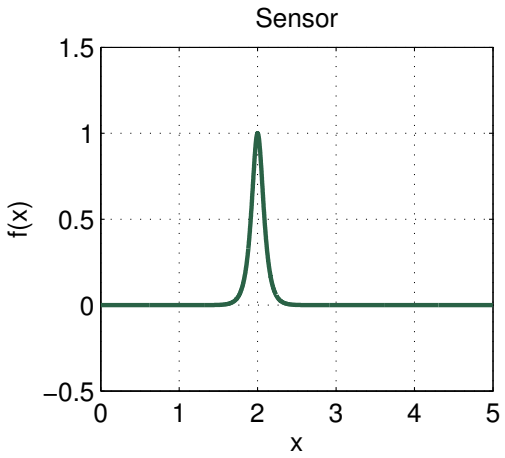

Figure 5.1: Model sensor ($\ell = 2$)

Recall the definition of the mass matrix,

$$M_{ij} = \langle \phi_i, \phi_j \rangle.$$

Defining $C_1 = \Pi M$ the sensor matrix is

$$C = \begin{bmatrix} C_1 & 0^{1\times n} \end{bmatrix}.$$

5.1.1 Description of the Noise

The shape and behaviour of the noise assumed in the model is crucial for an optimal sensor location problem. As such, three different noise models were considered in this project. The time characteristics of the noise were mentioned in Section 4.3 where the Kalman filter was formulated. For the spatial characteristics, two different shapes are considered in this study.

5.1.2 Noise Type I

The first form of the noise considered here is a localized noise and is modelled by the same function that was used for the sensor,

$$g(x,t) = d(x)w_1(t), \tag{5.4}$$

where $d(x) = \text{sech}(15(x - x_0))$. This noise is localized at $x_0 \in [0, 5]$. Unless mentioned otherwise, x_0 is arbitrarily selected to be 1.5. It is intuitive that the optimal sensor location for such a noise might be exactly where this disturbance is added. Thus although this is clearly not a realistic choice, it allows the convenience for a comparison of the computational methods for locating the optimal sensor location. With this noise the finite-dimensional representation of the system is,

$$\begin{bmatrix} I & 0 \\ 0 & M + \beta K \end{bmatrix} \dot{z}(t) = \begin{bmatrix} 0 & I \\ -c^2 K & -c_d M \end{bmatrix} z(t) + \begin{bmatrix} 0 \\ GM \end{bmatrix} w_1(t), \tag{5.5}$$

$$y(t) = Cz(t) + w_2(t). \tag{5.6}$$

5.1.3 Noise Type II

As the second noise model a the following function was used,

$$g(x, t) = \sum_{j=1}^{m} b_j(t) \sin\left(\frac{j\pi x}{L}\right), \tag{5.7}$$

where $b = \{b_i(t), i = 0, 1, 2..m\}$ are the time dependent co–efficients. Each of these co–efficients is a random variable and are selected such that $g(x, t)$ is a white Gaussian noise. The variance can be choosen to increase or decrease with mode number. The former would imply noise in electronic equipments where the noise is greater at higher frequencies while the latter would be more representative of a turbulence type noise.

Let $w_1(t) = \begin{bmatrix} 0 & b \end{bmatrix}^T$. Then, the representation of the system in this case is,

$$\begin{bmatrix} I & 0 \\ 0 & M + \beta K \end{bmatrix} \dot{z}(t) = \begin{bmatrix} 0 & I \\ -c^2 K & -c_d M \end{bmatrix} z(t) + \begin{bmatrix} 0 & 0 \\ 0 & M \end{bmatrix} w_1(t), \tag{5.8}$$

$$y(t) = Cz(t) + w_2(t). \tag{5.9}$$

This noise type represents the physical nature of a noise where the disturbance is greater at higher frequencies.

5.2 Numerical Results

In this section the numerical results are presented for the optimal sensor location problem in using a Kalman filter for the linear dispersive equation (2.24). Results are shown to

indicate the convergence of the estimator calculated from the three discretization methods and also the cost function with respect to sensor locations.

Recall that the cost function which was introduced at the beginning of this chapter was for a steady state filter and hence it is not a function of time. However, in order to be practical, it is required that this steady state filter is simulated over time assuming real-time data. This however requires a function that would be representative of this cost for a given sensor location. Recall the cost function in (5.1). For simulations, the following time dependent cost function will be used as well.

$$J_e(\ell, t) = \mathbf{E}[(z(t) - \hat{z}(t))^T (z(t) - \hat{z}(t))], \tag{5.10}$$

For computing the optimal sensor location a naive approach was used where the spatial domain was discretized with resolution .01 units and a sensor could be placed at each of these locations. The resulting ARE was solved and the trace of the solution was recorded and normalized using the maximum of all locations to give the cost with respect to the sensor location. Then the location corresponding to the minimum value was recorded as the optimal location.

H	$0.05\ m$
L	$5.0\ m$
g	$9.81\ ms^{-2}$
c_d	$0.0025\ m$
c	$\sqrt{gH}\ ms^{-1}$
β	$H^2/6\ m^2$

Table 5.1: Physical Constants

The constants selected were such that a reasonable physical system is represented. As long as, $\frac{H}{L} << 1$ any H or L should be fine. The value of g is $9.81\ ms^{-2}$ which is the gravitational acceleration. The damping constant was selected in an ad–hoc manner such that the dissipation is not very high, that is the system does not reach a steady–state very fast.

5.2.1 MATLAB Riccati Solver

Solving the Algebraic Riccati Equation (4.22) is one of the most important steps in obtaining the results. This is the numerical procedure where the structure of the state matrices

derived from different methods might play an important role. The MATLAB function CARE was used to solve the ARE. A desktop computer system with a 2.5GHz Intel Core i5 processor was used for the computations. All the methods yielded residues less than 10^{-11}. The time taken to solve the ARE was almost the same for all the methods, as shown in Figure 5.2. This is a consequence of the similar sparsity structure shown in Figure 3.4.

5.2.2 Convergence of the Kalman filter

Recall the expression of the gain of the filter in Section 4.3.2, which is rewritten below,

$$F = \begin{bmatrix} F_1 \\ F_2 \end{bmatrix} = \begin{bmatrix} P_1 C_1 \\ [M + \beta K] P_3 C_1 \end{bmatrix} R^{-1} .$$

In order to show the convergence of the estimator for the different methods with respect to the degrees of freedom, the Kalman gain can be regarded as a function of the spatial variable, x. Using the vector of the basis functions Φ_n this function can be reconstructed as

$$H(x) = F_1 \Phi_n(x) + [M + \beta K]^{-1} F_2 \Phi_n(x) .$$

The relative difference calculated as,

$$\frac{||H_{\sin}(x) - H_{fem}(x)||}{||H_{\sin}(x)||}$$

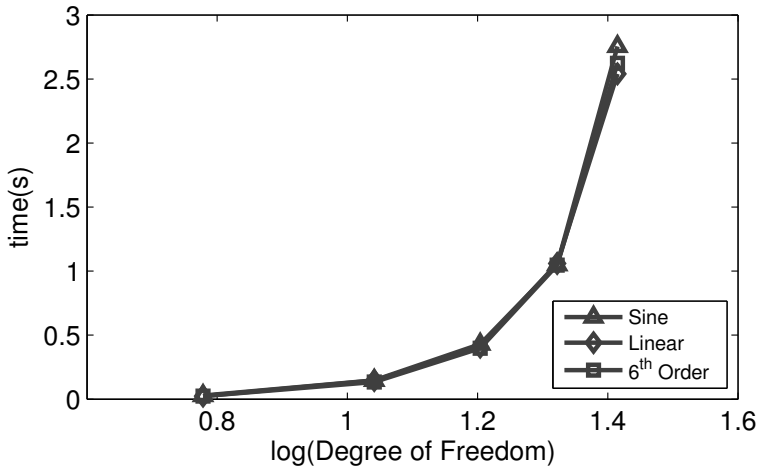

Figure 5.2: Time taken to solve ARE in MATLAB

in the L_2-norm between $H(x)$ for the linear basis and the sine basis with $n = 101$ was 6.77×10^{-3}. The same for the sine basis and the 6^{th} order polynomial basis with $n = 101$ was 8.49×10^{-3}. This indicates that all three methods yield solutions that converge to the same Kalman gain. However, Figure 5.3 shows that the estimator with the sine basis converges much faster than the other two methods. The two finite element methods show similar rate of convergence with degrees of freedom.

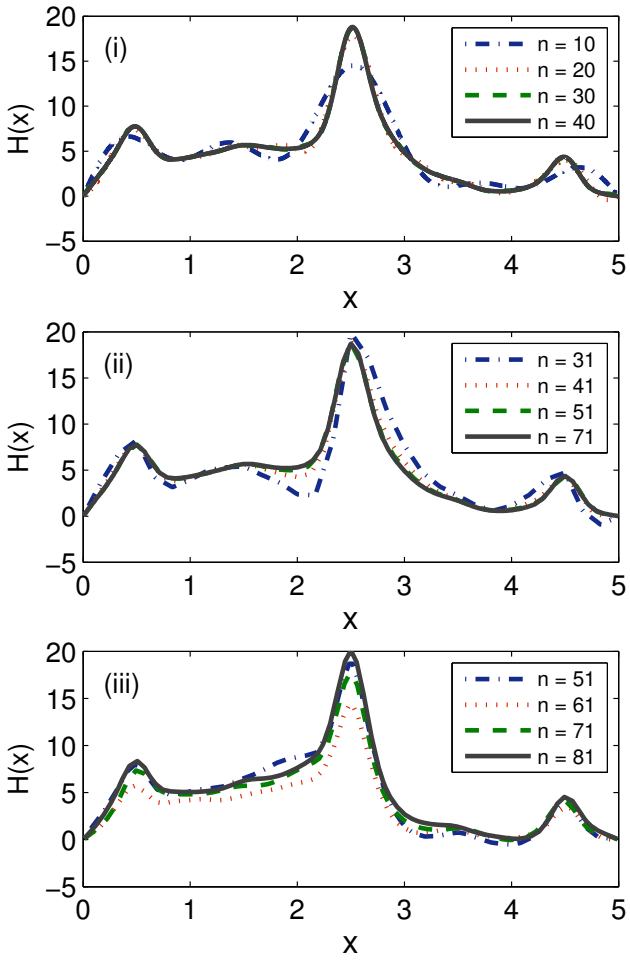

Figure 5.3: $H(x)$ on - (i) Sine basis (ii) FEM with linear basis (iii) FEM with 6^{th} Order basis . (The sensor location was arbitrarily selected to be $f_{2.5}$ and $G = f_{1.5}$. The noise parameters are $Q = 10$ and $R = 0.01$.)

From Sections 5.2.2 and 5.2.1 it is clear that the performance of the sine basis is

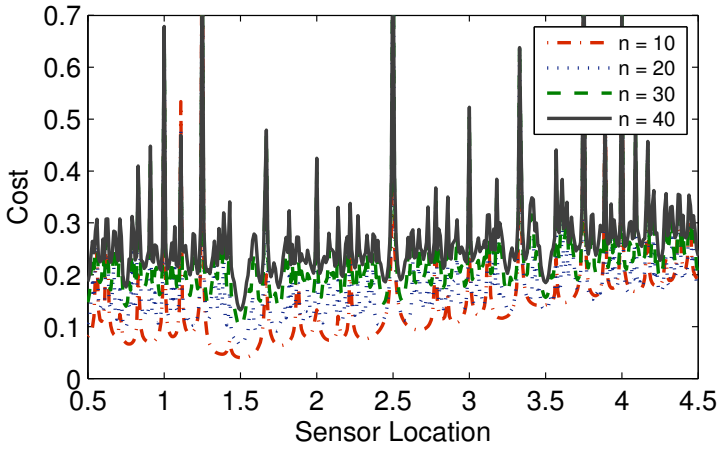

Figure 5.4: Normalized cost vs sensor location with increasing number of modes in sine basis and Noise Type 1. The optimal location is where the cost is minimum ($x = 1.5$).

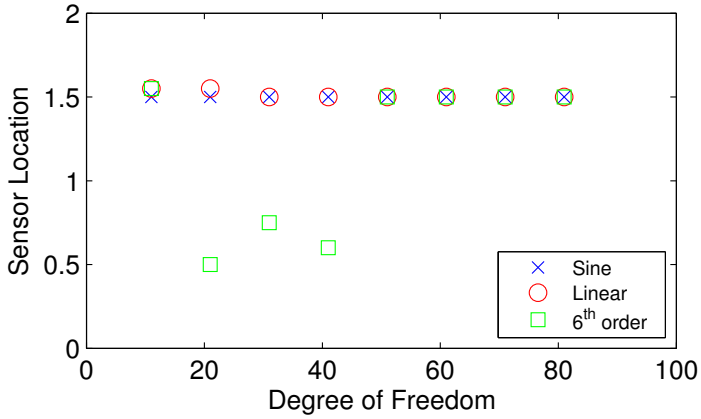

Figure 5.5: Optimal sensor locations with an increasing order of approximation for Noise Type 1.

marginally better than the finite element methods mainly because appropriate accuracy levels can be reached with a smaller degree of freedom. However, all the methods do converge to the same filter.

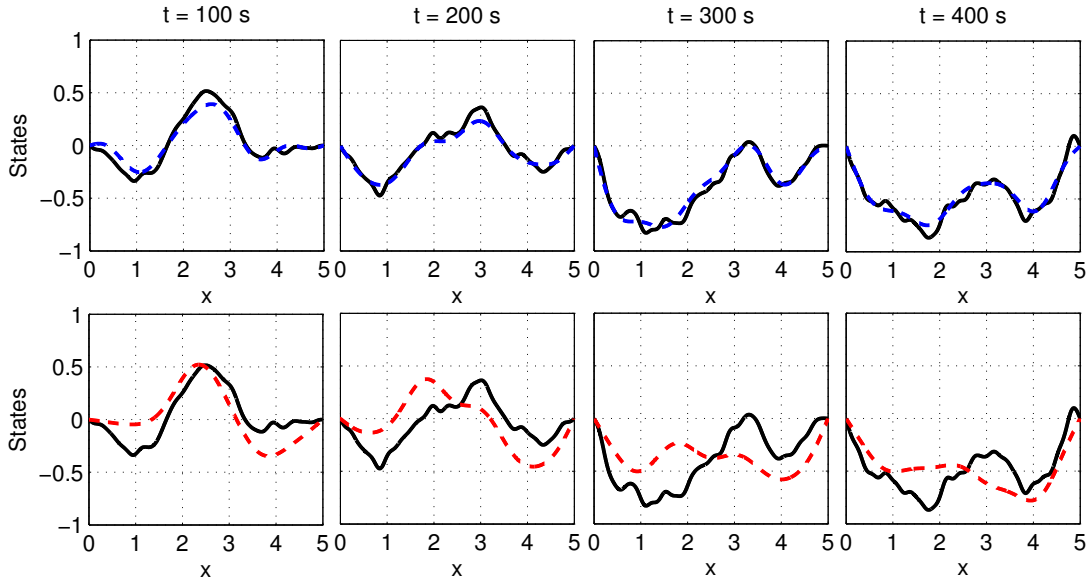

Figure 5.6: State Estimation (Noise Type 1)- Solid lines represent true states and dashed lines represent estimated states. Row 1: Optimal sensor location ($\ell = 1.5$), Row 2: Non-optimal sensor location. ($\ell = 2.5$) The noise parameters are Q = 1 and R = .001.

5.2.3 Optimal Sensor Location (Noise Type I)

The numerical results in Figure 5.4 indicate that the cost function is converging to the true cost function where the minimum point corresponds to the optimal sensor location. Similar results were obtained for different approximation methods, but convergence was slower than for the sine basis. All the methods were found to converge to the cost function where the optimal sensor location is given by $\ell = 1.5$, which co-incides with the location where noise is added.

The Kalman gain calculated with the sine basis with 20 degrees of freedom was used to investigate the effect of different sensor locations. For comparison, a set of system states was derived using the sine basis and $n = 40$. In Figure 5.6, the estimation results for two sensor locations are shown. The estimator clearly provides a better estimate of the state when placed optimally. This is also evident in Figure 5.7.

The estimator is sensitive to the sensor noise variance R. The calculated optimal sensor locations for different values of R are shown in Figure 5.8. For higher values of the variance R the cost is less sensitive to the location of the sensor. This may be because with the

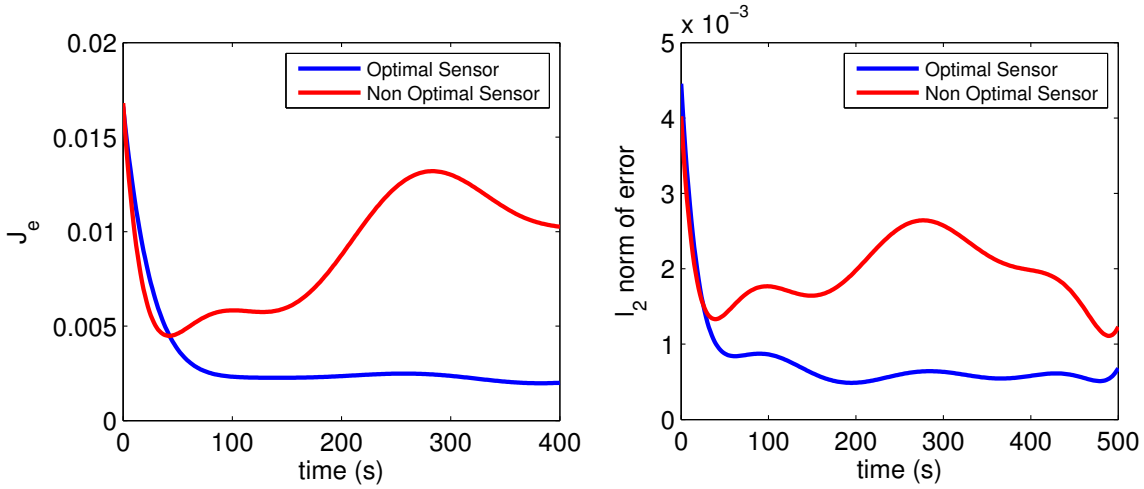

Figure 5.7: J_e and l_2-norm of error for two sensor locations with Noise Type 1. Blue represents the optimal sensor($\ell = 1.5$) and red represents a non-optimal sensor ($\ell = 2.5$).

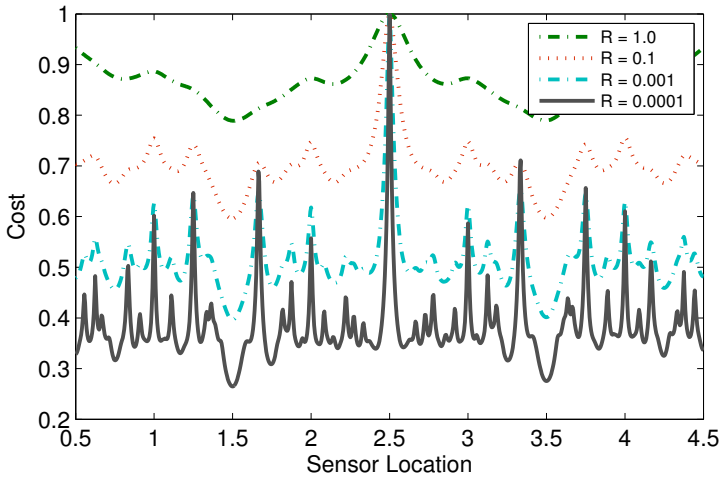

Figure 5.8: Cost function for different values of R and $Q = 1$ (Noise Type 1). Optimal sensor location given by $\ell = 1.5$ for all.

same sensor but more noise, it will take more time for the estimated states to converge to the required states, and local spatial information becomes less important.

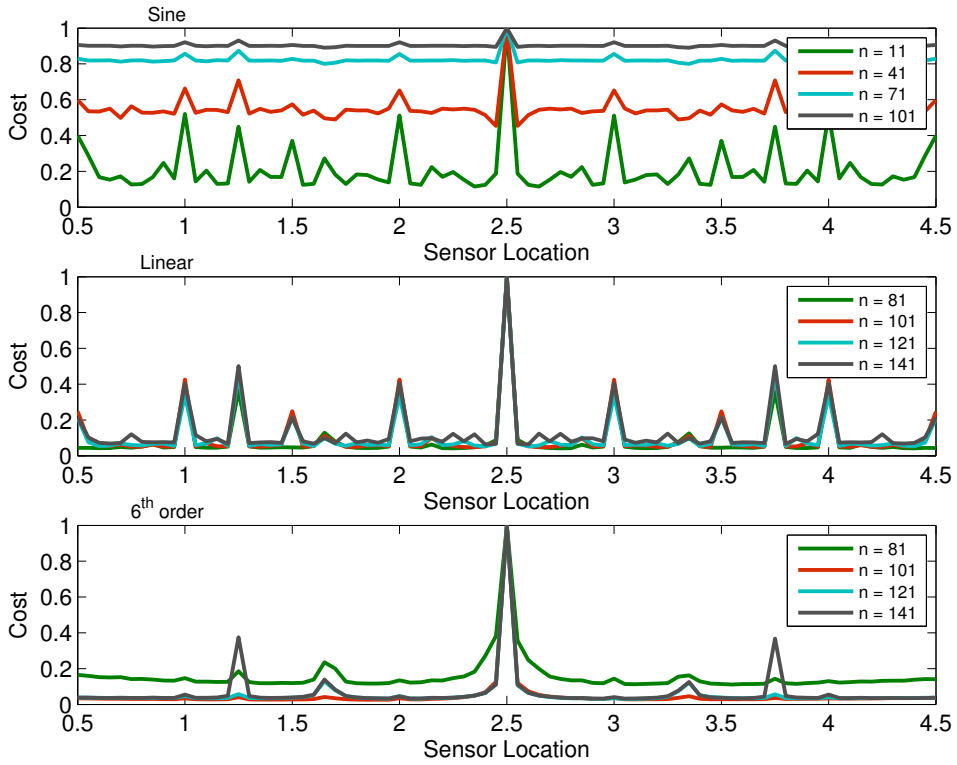

Figure 5.9: Cost functions for Noise Type 2 with increasing degree of freedom

5.2.4 Optimal Sensor Location (Noise Type II)

The optimal sensor location for Noise Type II is a more interesting problem since the noise is not localized and hence there exists no obvious location to place the sensor. In this simulation, noise is added to all the state components as shown in (5.8). Then the optimal sensor location is checked for the estimators designed with increasing degree of freedom using all the approximation schemes. The assumed true system here is again constructed using the sine basis and $n = 200$. The variance of the disturbance in the sensor is $R = .0001$ implying a good quality sensor for the estimation. Although there exists good and bad locations. Due to the physical shape of the noise, there does not exist very distinctive locations where the cost is significantly lower than any other point. See Figure 5.9 where the cost functions using an increasing number of modes for all the approximation methods are shown. However, placing the sensor in a good location still improves the estimation process as shown in Figure 5.10.

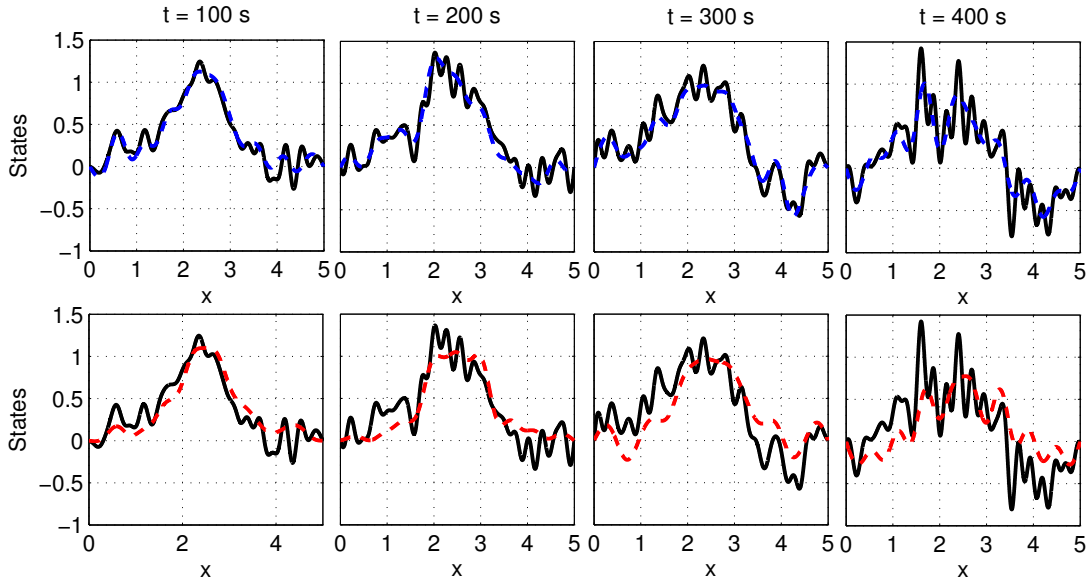

Figure 5.10: State Estimation (Noise Type 2)- Solid lines represent original states and dashed lines represent estimated states. Row 1: Good sensor location ($\ell = 2.45$), Row 2: Poor sensor location. ($\ell = 2.5$).

5.2.5 Optimal Sensor Location (Turbulence as Noise)

With the noise selection in the previous section, the results of optimal sensor location show that there is significant improvement in the estimation when the sensor is placed optimally. However, none of the noise model considered is very physical - one is very localized where the optimal sensor location is intuitive, and the other being very distributed where still an optimal location is still more efficient.

The following noise model is a step towards a more physical case. The noise model is taken from [13, Section F] and all the approximation methods are used in this case. First the optimal sensor location for all the three methods were computed with increasing degree of freedom and it is shown in Figure 5.13. The turbulence noise model is formulated as show below. Let,

$$\phi(x) = \frac{1}{\sqrt{2\pi}} e^{\frac{-x^2}{2}}, \quad \Phi(x) = \frac{1}{2}\left[1 + erf\left(\frac{x}{\sqrt{2}}\right)\right],$$

and

$$f(x) = 2\phi(\frac{x-\zeta}{\kappa})\Phi(\alpha\frac{x-\zeta}{\kappa}) \ .$$

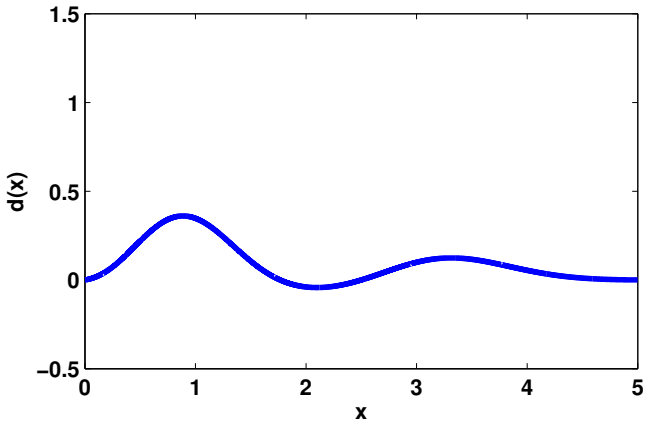

Figure 5.11: Physical shape of the turubulence type noise model

Then,

$$d(x) = \sin(\frac{2\pi x}{L}) f(x) \ .$$

The optimal sensor locations from all the methods converged to the location $x = .95$ for the optimal location. Note that the physical shape of the turbulence noise model also reaches a maximum at $x = .95$.

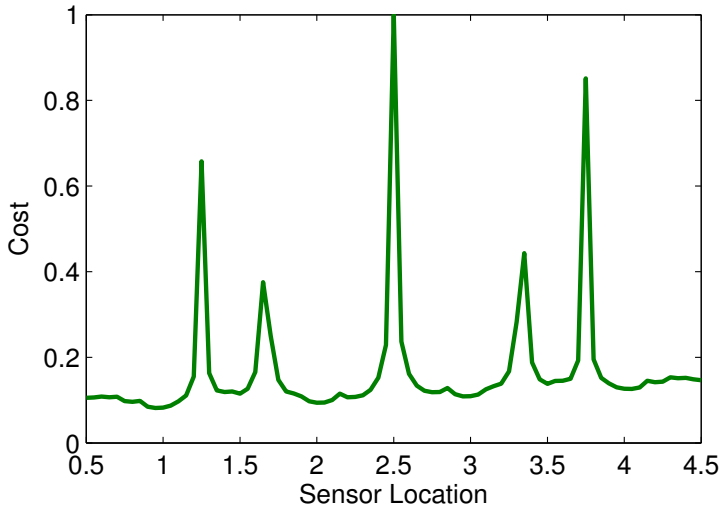

Figure 5.12: Cost vs sensor location for the turbulence noise model using sine basis and 40 modes

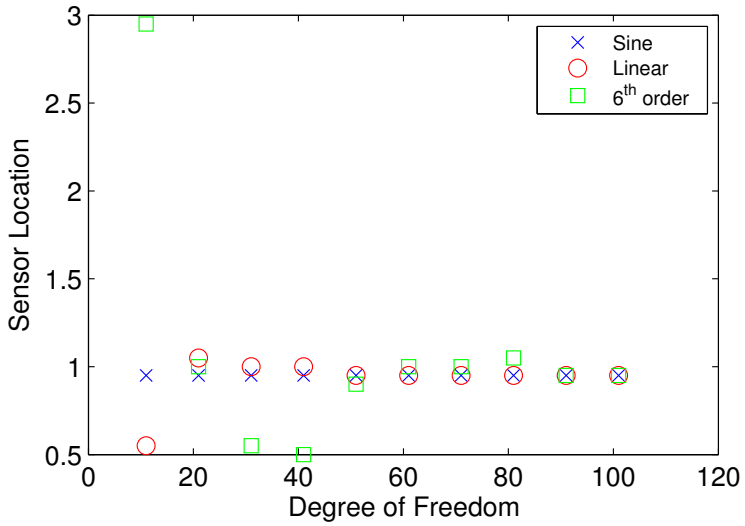

Figure 5.13: Optimal sensor locations with an increasing order of approximation for the turbulence noise model

All the methods converge to a single location, with the eigenfunctions converging the fastest in terms of the degree of freedom. This location was selected as the sensor location for an estimation process with all the methods of approximation. The degree of freedom where the finite element method with the polynomial basis converged to the optimal location was selected as the degree of freedom for all.

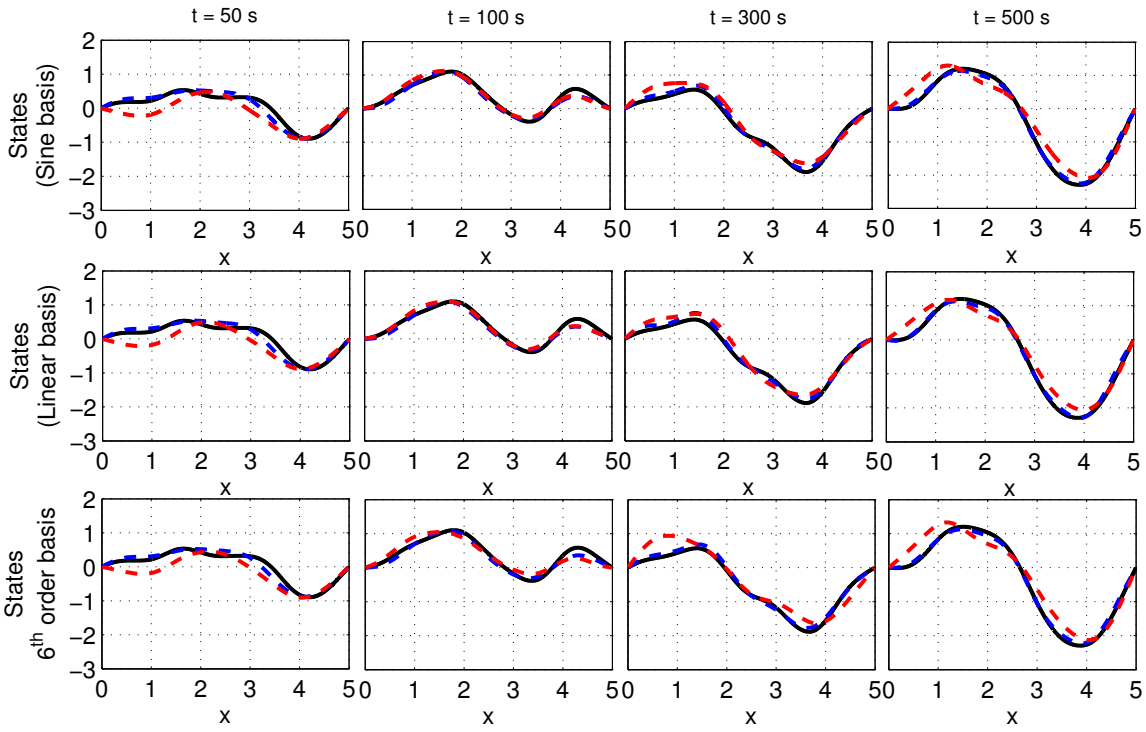

Figure 5.14: State estimation using all the three methods with the optimal sensor location and a non-optimal sensor location for the turbulence noise model

Chapter 6

Conclusions and Future Work

It was found that for certain descriptions of the physical system there exists certain locations where placing the sensor would result in a much better estimation. This location depends on various parameters of the noise model. The noise model consists of the spatial and transient characteristics, where the spatial structure played a more vital role since the noise was assumed stationary. Therefore, the more accurate the model of noise is the more reliable these results are. For a localized noise, the optimal location was always found to be where the noise is added, while for a distributed case, the optimal location was close to the center from either side of it. The variances of the process and sensor noise affect the significance of an optimal sensor location. For example if the sensor quality is bad, then the local spatial information will become less important and although an optimal sensor location might exist, it will not show significant improvements in estimation. On the other hand, if the process noise is very high relative to the sensor noise, then a sensor should be placed optimally to achieve better performance from the estimator.

In this project, the objective was to compare three discretization methods for the calculation of an optimal sensor location of a Kalman filter for estimating a linear dispersive wave equation. A large portion of the project was focused towards comparing the discretization methods. Although, the availability of eigenfunctions should by itself justify the selection of this basis, finite element methods become important when the eigenfunctions are not available. The eigenfunctions are not available for the same model on a complex shaped lake for instance. That is why these numerical methods were considered for a system where the eigenfunctions are available. Certainly, the eigenfunctions give better results than the finite element methods, although the latter were reasonable accurate as well. The major issue with the finite element methods is that they require large matrices for approximating the system. MATLAB warnings were issued for few cases while solving the Algebraic

Riccati Equation using a finite element approximation scheme. This might suggest the use of better algorithms to solve the ARE.

Finally, the project has many areas where it can be improved. To begin with, an optimization rule to guide the search for a discretized spatial domain for the optimal location would be very effective. In [10] the authors introduce such an optimization algorithm for the optimal actuator location problem. Secondly, the main model was constructed through simplifications, the most crucial of which is the linearization. Athough linearizing simplifies the problem, the physical system remains non-linear and the discrepancies have the potential to make the results insignificant in an applicative framework. There exists well-known estimation techniques for non-linear systems, such as the Extended Kalman filter [21], the unscented Kalman filter [45] or the sliding-moder observer [11]. This work is aimed at playing an instructive role towards a more useful construction of this problem. Another topic, where the problem could be improved is by using the full set of equations for the surface height and velocity (avoiding step 2.24). This will allow the adoption of displacement sensors among other type of sensors.

APPENDICES

Appendix A

Technical Background

A.1 The inverse operator in Section 2.3.1

Consider the operator $H = I - \beta D^2$ with I being the identity operator and $D(\cdot) = \frac{\partial}{\partial x}(\cdot)$. Let the domain,

$$D(H) = \{v \in \mathcal{L}_2(0, L) \mid v, \frac{dv}{dx} \text{ are absolutely continuous,} \\ \frac{d^2v}{dx^2} \in \mathcal{L}_2(0, L) \text{ and } v(0) = v(L) = 0\} \ .$$

This operator is in a Sturm-Liouville form. Therefore to obtain L^{-1}, the following steps are required [37].

Step 1 Solve $Hv(x) = v(x) - \beta v''(x) = 0$ for the boundary conditions separately. Let $\alpha = \frac{1}{\sqrt{\beta}}$. The solutions are,

$$v_1(x) = e^{\alpha x} - e^{-\alpha x}, \implies v_1(0) = 0 \\ v_2(x) = e^{\alpha x} - e^{2L\alpha} e^{-\alpha x}, \implies v_2(L) = 0.$$

Step 2 Calculate the Wronskian,

$$\begin{aligned} w(x) &= \begin{vmatrix} v_1 & v_2 \\ v_1' & v_2' \end{vmatrix} \\ &= \begin{vmatrix} e^{\alpha x} - e^{-\alpha x} & e^{\alpha x} - e^{2L\alpha}e^{-\alpha x} \\ \alpha e^{\alpha x} + \alpha e^{-\alpha x} & \alpha e^{\alpha x} + \alpha e^{2L\alpha}e^{-\alpha x} \end{vmatrix} \\ &= \alpha e^{2\alpha x} + \alpha e^{2L\alpha} - \alpha - \alpha e^{2L\alpha}e^{-2\alpha x} - \alpha e^{2\alpha x} - \alpha + \alpha e^{2L\alpha} + \alpha e^{2L\alpha}e^{-2\alpha x} \\ &= 2\alpha e^{2L\alpha} - 2\alpha \ . \end{aligned}$$

Step 3 Construct the associated Green's function,

$$H^{-1}v(x) = Tv(x) = \int_0^L g(x,y)v(y)dy,$$

where

$$g(x,y) = \frac{1}{-\beta w(0)} \begin{cases} v_1(x)v_2(y), & 0 \le x \le y \le L \\ v_2(x)v_1(y), & 0 \le y \le x \le L \end{cases} .$$

Note that $Tv(0) = 0$ since,

$$\begin{aligned} Tv(0) &= \int_0^0 v_2(x)v_1(y)v(y)dy + \int_0^L v_1(0)v_2(y)v(y)dy, \\ &= v_1(0)\int_0^L v_2(y)v(y)dy. \\ &= 0. \end{aligned}$$

Similarly, $Tv(L) = 0$ since,

$$\begin{aligned} Tv(L) &= \int_0^L v_2(x)v_1(y)v(y)dy + \int_L^L v_1(0)v_2(y)v(y)dy, \\ &= v_2(L)\int_0^L v_1(y)v(y)dy. \\ &= 0. \end{aligned}$$

Appendix B

MATLAB Code

This chapter contains some of the MATLAB functions that was written by the author for the project. The entire software can be found at www.github.com/tawsifkhan.

```
function [L, basisE]=myKalman(method1,method2)
% This function can be used to simulate a Kalman filter. The two arguments
% method1 and method2 refer to the basis of the estimator and the original
% systems respectively.
% This file will assume Noise Type 2. For Noise Type 1, minor changes are
% needed.
noise = load('noise.mat');     % Load Noise file for simulation
lArray= 2.5;%.5:.05:4.5;     % Array of sensor locations/Single location

saveOSL = 0;
saveSIM = 1;

specs= getSpecs;               % Problem specifications

tt=0:specs.dt:specs.T;         % Time span

switch lower(method1)          % This will make sure x-values are same for
                               % all methods
    case {'sine'},       matchN = specs.sineN;
    case {'linear'},     matchN = specs.linearN;
    case {'poly'},       matchN = specs.polyN;

end

basisE=basisSelect(method1,matchN);     % Basis for the estimator
```

```
basisO=basisSelect(method2,matchN);    % Basis for the true system

[A1_E,A2_E] = getStateMatrix(method1); % Estimator matrix A = inv(A1)*A2
[A1_O,A2_O] = getStateMatrix(method2); % True matrix A = inv(trueA1)*trueA2

nE = specs.n;                          % Modes in Estimator
nO = specs.nTrue;                          % Modes in Original/True

m = size(tt,2);
x = linspace(0,specs.J,matchN+1);       % x-values for estimate solution
x = getGaussianPoints(x,matchN);        % x-values gaussian points

Zo0= noise.Zo0;                         % Initial conditions are defined
Ze0= noise.Ze0(1:2*nE);                 % in the noisefile so that
Zo0 = [Zo0(1:nO); zeros(nO,1)];         % the program can compare different
                                        % methods with similar conditions

G = [zeros(nO,nO) zeros(nO,nO); zeros(nO,nO) basisO.M]; % G matrix
Ge= [zeros(nE,nE) zeros(nE,nE); zeros(nE,nE) basisE.M];


for sL = 1:size(lArray,2)                       % Search for optimal ...
   location
    l = lArray(sL);
    [C1_E,C2_E] = mySensor2(method1,l);         % Estimator matrix C ...
       =[C1 C2]
    [C1_O,C2_O] = mySensor2(method2,l);         % True matric C = ...
       [trueC1 trueC2]

    % Computing the Q matrix for estimator. The size is smaller than the
    % true system noise and hence it needs to be taken care of.
    for i=1:nE
        Q_E(i,i)=noise.QQ(i,i);
        Q_E(nE+i,nE+i)=noise.QQ(nO+i,nO+i);
    end

    Q = Ge*(Q_E*(Ge'));
    R = noise.RR;
    S = zeros(nE+nE,1);

    sprintf('Solving Riccati Equation')
    [X,¬,L,¬] = care(full(A2_E)',[C1_E C2_E]',(Q+Q')/2,R,S,full(A1_E)');
    cost(sL) = trace(X);
    L = L';                                     % Kalman gain
```

```
    % The following section will solve a finite time kalman filter. It will
    % require the m-file named myKalman_FT

%     options3 = odeset('Mass',A1_E(:));
%     P_init = cov(noise.Ze0*noise.Ze0');
%     P_init = reshape(P_init,size(A1_E));
%     [¬,P] = ode15s(@(t,P) myKalman_FT(t,P,full(A1_E),full(A2_E),[C1_E ...
   C2_E],Q,R,X),tt,P_init);
%
%     save('FTvsINFTNONOPT.mat')

end

if saveOSL==1
    jj = num2str(specs.n);
    filename = strcat('MAY_20_',jj,'.mat');
    save(filename)
    return
end

% Construct the ODE for the original system
original = @(t1,Z) A2_O*Z + G*(interp1(tt,[noise.w1]',t1,'spline'))';
options1 = odeset('Mass',A1_O);

sprintf('Solving Original System')
[¬, Z] = ode15s(original,tt,Zo0,options1);
Z = Z';
originalStates = Z(1:nO,:);               % Recall velocity-[v(t) dot(v(t)]

% Compute measurements
for i=1:size(tt,2), y(:,i) = C1_O*Z(1:nO,i)+noise.w2(1,i); end

% Construct the ODE for the Estimator
estimator = @(t2,Ze) (A2_E-L*[C1_E C2_E])*Ze+ L*(interp1(tt,y,t2,'spline'));
options2 = odeset('Mass',A1_E);

sprintf('Solving Estimator')
[¬,Ze]= ode15s(estimator,tt,Ze0,options2);
Ze= Ze';
estimatedStates = Ze(1:nE,:);

sprintf('Solving Original System Without Noise')
true = @(t1,ZZ) A2_O*ZZ;%+ GT*(interp1(tt,w1,t1,'spline'))';
options3 = odeset('Mass',A1_O);
[¬,ZZ]= ode15s(true,tt,Zo0,options3);
```

```
ZZ=ZZ';
trueStates = ZZ(1:nO,:);
if saveSIM==1
    jj = num2str(specs.n);
    filename = strcat('MAY_21_NonOptimal',jj,'.mat');
    save(filename)
    return
end

% Simulate the estimation
i=0;
for ii=1:size(tt,2)
   i=i+1;
    [noiseF,x] =  reconstructSineTrue(noise.w1(nO+1:end,ii),basisO,matchN);
    [vOriginal,x] = reconstructorSelect(originalStates(:,ii),basisO,...
        method2,matchN);
    [vEstimate,x] = reconstructorSelect(estimatedStates(:,ii),basisE,...
        method1,matchN);
    [vTrue, xT] = reconstructorSelect(trueStates(:,ii),basisO,...
        method2,matchN);

    figure(10)
    clf
    set(gcf, 'color', [1 1 1])
    subplot(3,1,1)
    plot(x,vTrue,'r-',x,vOriginal,'b-','LineWidth',2)
    axis([0 specs.J -.5 .5])
    grid on

    ylabel('Displacement')
    xlabel('x')
    subplot(3,1,2)
    plot(x,noiseF,'LineWidth',2)
    axis([0 specs.J -.5 .5])
    grid on
    ylabel('Displacement')
    xlabel('x')

    subplot(3,1,3)
    plot(x,vOriginal,'b-',x,vEstimate,'r-','LineWidth',2);
    axis([0 specs.J -2 2])
    grid on
    ylabel('Displacement')
    xlabel('x')
```

```
    drawnow
end
end
```

```
function [C1, C2]=mySensor2(method,l)

% Returns coloumn vector of C 1x2n of the sensor
% n is the number of modes
% Recall state vector = [v(t) dot{v(t)}]
specs=getSpecs;

actualSensor = @(x) sech(15*(x-l)); %l is location of the sensor.
plotFlag=0 ;

switch lower(method)
    case {'sinetrue'}
        xg = getGaussianPoints(linspace(0,specs.J,specs.sineTrueN+1)...
            ,specs.sineTrueN);
        basis=getBasisSineTrue(specs.sineTrueN);
        coEffs=getCoEffsSineTrue(actualSensor(xg'),basis);

    case {'sine'}
        sprintf('Projecting Sensor on Sine Basis')
        xg = getGaussianPoints(linspace(0,specs.J,specs.sineN+1)...
            ,specs.sineN);
        basis=getBasisSine;
        coEffs=getCoEffsSine(actualSensor(xg'),basis);

    case {'linear'}
        sprintf('Projecting Sensor on Linear Basis')
        xg = getGaussianPoints(linspace(0,specs.J,specs.linearN+1)...
            ,specs.linearN);
        basis=getBasisLinear;
        coEffs=getCoEffsLinear(actualSensor(xg'),basis);
    case {'poly'}
        sprintf('Projecting Sensor on Poly Basis')
        xg = getGaussianPoints(linspace(0,specs.J,specs.polyN+1)...
            ,specs.polyN);
        basis=getBasisPoly;
        coEffs=getCoEffsPoly(actualSensor(xg'),basis);
end

if plotFlag==1
    figure(1)
    subplot(2,1,2)
    [constructedSensor,xSensor]=reconstructorSelect(coEffs,basis,method);
    set(gcf,'DefaultLineLineWidth',2,'DefaultTextFontSize',12,...
        'DefaultTextFontWeight','normal','DefaultAxesFontSize',12,...
        'DefaultAxesFontWeight','normal','color','w');
```

```
    hold on
    plot(xSensor,constructedSensor,'g')
    xlabel('x')
    ylabel('f(x)')
    axis([0 specs.J -0.5 1.5])
    title('Model Sensor')
    grid on
end

% Assemble C matrix
sizeCoEffs=size(coEffs,1);
C1=coEffs'*basis.M;
C2=zeros(1,sizeCoEffs);
end
```

Below is the program that will generate the basis using the 6^{th}-order polynomial functions. There are two more similar functions generating the sine and linear basis. The following function was written by Professor Marek Stastna and was modified by the author.

```
% Function defines the polynomial basis on a canonical element
% and returns the shape functions, mass and stiffness matrices.

function [basis, kl, ml] = getBasisPoly

specs=getSpecs;
x=linspace(0,specs.J,specs.polyN+1);
xi=[-0.932469514203152 -0.661209386466265 -0.238619186083197 ...
      0.238619186083197 0.661209386466265 0.932469514203152];
wts=[0.17132449237910 0.360761573048139 0.467913934572691];
wts=[wts wts(3:-1:1)];
m = length(xi);
% Legendre Polynomials
p0=ones(size(xi));
p1=xi;
p2=(1/(1+1))*((2*1+1)*xi.*p1-1*p0);
p3=(1/(2+1))*((2*2+1)*xi.*p2-2*p1);
p4=(1/(3+1))*((2*3+1)*xi.*p3-3*p2);
p5=(1/(4+1))*((2*4+1)*xi.*p4-4*p3);

% Derivatives using the recursive method
p0p=0;
p1p=1;
p2p=(2*1+1)*p1+p0p;
p3p=(2*2+1)*p2+p1p;
p4p=(2*3+1)*p3+p2p;
p5p=(2*4+1)*p4+p3p;

p0pp=0;
p1pp=0;
p2pp=3;
p3pp=15*xi;
p4pp=0.5*105*xi.*xi-0.5*15;
p5pp=0.5*315*xi.*xi.*xi-0.5*105*xi;

% Shape functions
nm1=0.5*(1-xi);
np1=0.5*(1+xi);
n02=(p2-p0)/sqrt(2*(2*2-1));
```

```
n03=(p3-p1)/sqrt(2*(2*3-1));
n04=(p4-p2)/sqrt(2*(2*4-1));
n05=(p5-p3)/sqrt(2*(2*5-1));

nm1p=-0.5;
np1p=0.5;
n02p=(p2p-p0p)/sqrt(2*(2*2-1));
n03p=(p3p-p1p)/sqrt(2*(2*3-1));
n04p=(p4p-p2p)/sqrt(2*(2*4-1));
n05p=(p5p-p3p)/sqrt(2*(2*5-1));

nm1pp=0;
np1pp=0;
n02pp=(p2pp-p0pp)/sqrt(2*(2*2-1));
n03pp=(p3pp-p1pp)/sqrt(2*(2*3-1));
n04pp=(p4pp-p2pp)/sqrt(2*(2*4-1));
n05pp=(p5pp-p3pp)/sqrt(2*(2*5-1));

% store the polynomials for a more elegant way of defining the k matrix

myps=zeros(m,6);
mypsp=zeros(m,6);
mypspp=zeros(m,6);

myps(:,1)=nm1;
myps(:,2)=np1;
myps(:,3)=n02;
myps(:,4)=n03;
myps(:,5)=n04;
myps(:,6)=n05;

mypsp(:,1)=nm1p;
mypsp(:,2)=np1p;
mypsp(:,3)=n02p;
mypsp(:,4)=n03p;
mypsp(:,5)=n04p;
mypsp(:,6)=n05p;

mypspp(:,1)=nm1pp;
mypspp(:,2)=np1pp;
mypspp(:,3)=n02pp;
mypspp(:,4)=n03pp;
mypspp(:,5)=n04pp;
mypspp(:,6)=n05pp;
```

```
%   Assemble the local matrices
%   Set Global Order: nm1 n01 n02 n03 n04 np1
%   Map the local matrices to the global matrices

K=(zeros((specs.polyN)*5+1));
M=(zeros((specs.polyN)*5+1));

%   gli stores the indices for mapping local to global matrices
for kk=1:specs.polyN, gli(kk,:)=[1+(kk-1)*5, 1+(kk)*5, ...
    (kk-1)*5+2:(kk-1)*5+5]; end
for kk=1:specs.polyN
    xm=x(kk); xp=x(kk+1); dx=xp-xm; %xvals=xm+(xi+1)*0.5*dx;
    % Define the local matrices: kl,ml
    for ii=1:6,
        for jj=1:6,
        kl(ii,jj)=(2/dx)*sum(wts.*(mypsp(:,ii)').*(mypsp(:,jj))');
        ml(ii,jj)=(dx/2)*sum(wts.*(myps(:,ii)').*(myps(:,jj))');
        i_kk=gli(kk,ii);    j_kk=gli(kk,jj);
        K(i_kk,j_kk)=K(i_kk,j_kk)+kl(ii,jj);
        M(i_kk,j_kk)=M(i_kk,j_kk)+ml(ii,jj);
        end
    end
end
K(1,:)   =[1 zeros(1,(specs.polyN)*5)];   K(:,1)   =[1 ...
    zeros(1,(specs.polyN)*5)]';
K(end,:) =[zeros(1,(specs.polyN)*5) 1];   K(:,end) ...
    =[zeros(1,(specs.polyN)*5) 1]';
M(1,:)   =[1 zeros(1,(specs.polyN)*5)];   M(:,1)   =[1 ...
    zeros(1,(specs.polyN)*5)]';
M(end,:) =[zeros(1,(specs.polyN)*5) 1];   M(:,end) ...
    =[zeros(1,(specs.polyN)*5) 1]';
K = sparse(K);
M = sparse(M);

basis = struct('elements',specs.polyN,...  % Number of Elements
               'ps',myps,...               % Basis functions
               'psp',mypsp,...             % Spatial Derivatives
               'pspp',mypspp,...           % 2nd Spatial Derivatives
               'M',M,...                   % Mass Matrix
               'K',K,...
               'dx',dx);                     % Stiffness Matrix

end
```

References

[1] O. Axelsson and V. A. Barker. *Finite Element Solutions of Boundary Value Problem*. Academic Press Inc., 1984.

[2] J. Borggaard, J.A. Burns, and L. Zietsman. On using LQG performance metrics for sensor placement. In *Proceedings of the American Control Conference*. IEEE, 2011.

[3] R. Bucy and P. Joseph. *Filtering for Stochastic Processes with Applications to Guidance*. John Wiley and Sons, New York, 1968.

[4] K.K. Chen and C.W. Rowley. H_2-optimal actuator and sensor placement in the linearised complex Ginzburg-Landau system. *Journal of Fluid Mechanics*, 681(241–260), 2011.

[5] K.K. Chen and C.W. Rowley. Fluid flow control applications of h-2 optimal actuator and sensor placement. In *Proceeding of the American Control Conference*, 2014.

[6] Y. Chen, N. Xia, and N. Yi. A legendre galerkin spectral method for optimal control problems. *J. Syst. Sci. Comp.*, 24:663–671, Apr 2011.

[7] Bruno Costa. Spectral methods for partial differential equations. *A Mathematical Journal*, 6(4):1–32, 2004.

[8] Ruth F. Curtain and Hans Zwart. *An Introduction to Infinite Dimensional Linear Systems Theory*. Springer - Verlag New York, Inc., 1995.

[9] N. Darivandi, K. Morris, and A. Khajepour. An algorithm for LQ-optimal actuator location. *Smart Materials & Structures*, 22(3):035001, 2013.

[10] Neda Darivandi, Kirsten Morris, and Amir Khajepour. An algorithm for LQ optimal actuator location. *Smart Materials and Structure*, 22(3):035001, January 2013.

[11] Jorge Davila, Leonid Fridman, and Arie Levant. Second-order sliding-mode observer for second-order sliding-mode observer for mechanical systems. *IEEE Tran. on Automatic Control*, 50(11), 2005.

[12] A. de la Fuente, K. Shimizu, J. Imberger, and Y. Niño. The evolution of internal waves in a rotating, stratified, circular basin and the influence of weakly nonlinear and nonhydrostatic accelerations. *Limnol. Oceanogr.*, 53(6):2738–2748, 2008.

[13] P. Clark di Leoni, P. J. Cobelli, and P. D. Mininni. Wave turbulence in shallow water models. *Cornelly University Library*, 2014.

[14] Lars Elden, Linde Wittmeyer-Koch, and Hans Bruun Nielsen. *Introduction to Numerical Computation*. Studentlitteratur, 2004.

[15] K. J. Engel and R. Nagel. *One-Parameter Semigroup for Linear Evolution Equations*. Springer, 2000.

[16] F. Fahroo and M. A. Demetriou. Optimal actuator/sensor location for active noise regulator and tracking control problems. 11:737–745, September 2003.

[17] J. E. Flaherty. Finite element analysis.

[18] M. I. Frecker. Recent advances in optimization of smart structures and actuators. *J. Intell. Mat'l & Sys. & Struct.*, 14:4-5:207–216, May 2013.

[19] Ø. Grønlie. Wave radars a comparison of concepts and techniques. *Hydro International*, 8, Jun 2004.

[20] Guoxian Gu and Pramod P. Khargonekar. Approximation of infinite-dimensional systems. *IEEE Tran. on Automatic Control*, 1989.

[21] Simon Haykin. *Kalman filtering and neural networks*, volume 47. John Wiley & Sons, 2004.

[22] D. O. Hodgins, P. H. LeBlond, and D. A. Huntley. Shallow-water wave calculations. Technical report, Marine Environmental Data Services Branch, Department of Fisheries and Oceans, 1985.

[23] P. L. Houtekamer, Herschel L. Mitchell, Gérard Pellerin, Mark Buehner, Martin Charron, Lubos Spacek, and Bjarne Hansen. Atmospheric data assimilation with an ensemble kalman filter: Results with real observations. *Monthly Weather Review*, 133:604–620, 2005.

[24] Clas A. Jacobson. Linear state-space systems in infinite-dimensional space: The role and characterization of joint stabilizability/detectability. *IEEE Tran. on Automatic Control*, 33(6):541–549, June 1988.

[25] R. E. Kalman. A new approach to linear filtering and prediction problems. 82:35–45, 1961.

[26] D. Kasinathan and K. A. Morris. H_∞-optimal actuator location. *IEEE Tran. on Automatic Control*, 58(10):2522 – 2535, 2013.

[27] P. K. Kundu and I. M. Cohen. *Fluid Mechanics*. Elsevier Academic Press, 4th edition, 2008.

[28] Frank L. Lewis. *Optimal Estimation with an Introduction to Stochastic Control Theory*. Wiley-Interscience, 1986.

[29] Pengzhi Lin. *Numerical Modeling of Water Waves*. Taylor and Francis Group, 2008.

[30] Yiming Lou and Panagiotis D. Christofides. Optimal actuator/sensor placement for nonlinear control of the kuramoto-sivashinsky equation. 11:737–745, September 2003.

[31] G. Lumer and R. S. Philips. Dissipative operators in a banach space. *Pacific Journal of Mathematics*, 1961.

[32] M. Moallem, M.R. Kermani, R. V. Patel, and M. Ostojic. Flexure control of a positioning system using piezoelextric transducers. 12:757–752, September 2004.

[33] K. A. Morris. Control of systems governed by partial differential equations. *The IEEE Control Theory Handbook*, 2010.

[34] K. A. Morris. Linear-quadratic optimal actuator location. 56:113–124, Jan 2011.

[35] K. A. Morris, M. A. Demetriou, and S. D. Yang. Using $\mathbb{H}_2$-control performance metrics for infinite-dimensional systems. *IEEE Tran. on Automatic Control*, 2014. to appear.

[36] Kirsten Morris. Design of finite-dimensional controllers for infinite-dimensional systems by approximation. *Journal of Mathematical Systems, Estimation, and Control*, 1994.

[37] A. W. Naylor and G. R. Sell. *Linear Operator Theory in Engineering and Science*. Springer, 1982.

[38] The NOAA website, 2011.

[39] J. A. Putnam. Loss of wave engergy due to percolation in a permeable sea bottom. *Transactions, American Geophysical Union*, 30(349-356), 1949.

[40] J. N. Reddy. *An Introduction to the Finite Element Method*. McGraw-Hill Education, 2005.

[41] R. E. Showalter. Hilbert space methods for partial differential equations. *Electronic Journal of Differential Equations*, 1994.

[42] Dan Simon. *Optimal State Estimation: Kalman, H_∞ and Nonlinear Approaches*. Wiley-Interscience, 2006.

[43] L. N. Trefethen. *Finite Difference and Spectral Methods for Ordinary and Partial Differential Equations*. Cornell University [Department of Computer Science and Center for Applied Mathematics], 1996.

[44] M. van de Wal and B. de Jager. A review of methods for input/output selection. *Automatica*, 37:487–510, April 2001.

[45] Eric A. Wan and Rudolph van der Menve. The unscented kalman filter for nonlinear estimation. *Adaptive Systems for Signal Processing, Communications, and Control Symposium 2000. AS-SPCC. The IEEE 2000*, pages 153–158, 2000.

[46] G. B. Whitham. *Linear and Nonlinear Waves*. John Wiley & Sons, 1974.

[47] S. Yang and K. Morris. Comparison of linear-quadratic and controllability criteria for actuator placement on a beam. In *American Control Conference*, pages 4069–4074, Portlan, Oregon, USA, June 2014.

[48] S. D. Yang and K. A. Morris. Comparision of actuator placement criteria for control of beam vibrations. *J. of Sound and Vibration*, 2014. submitted.