ABSTRAK

JUDUL ABSTRAK

Oleh
Ahmad Izzuddin
NIM: 1908919

Abstrak merupakan penjelasan singkat dan padat tentang pekerjaan dan hasil penelitian TA, yang dituliskan secara teknis. Abstrak memiliki karakter tegas dan komprehensif, dan hanya dapat dituliskan setelah pekerjaan penelitian telah mencapai tahap tertentu, dan karenanya ada hasil penelitian yang dapat dilaporkan. Abstrak ditulis menjelang akhir penyelesaian penulisan buku TA.

Secara umum, abstrak memuat beberapa komponen penting, yaitu: konteks atau cakupan pekerjaan penelitian, tujuan penelitian, metodologi yang digunakan selama penelitian, hasil-hasil penting yang dapat ditambahkan dengan implikasinya, dan simpulan dari penelitian. Dengan demikian, suatu abstrak tidak dapat dituliskan apabila penelitian belum mencapai hasil tertentu, apalagi kalau penelitiannya pun belum dilakukan.

Panjang abstrak sebaiknya dicukupkan dalam satu halaman, termasuk kata kunci. Tiga kata kunci dipandang cukup, yang masing-masingnya memuat paduan kata utama, yang dapat merepresentasikan isi Abstrak. Halaman Abstrak tidak memuat informasi judul dan penulis, sehingga tidak secara langsung dapat digunakan sebagai lembaran Abstrak Sidang TA yang disediakan untuk hadirin, yang memerlukan tambahan (sekurangnya) dua informasi tersebut.

Kata kunci: Konsep Abstrak, Komponen Abstrak, Kata Kunci.

ABSTRACT

TITLE

by
Ahmad Izzuddin
NIM: 1908919

In general, Abstract is a translation of Abstrak. However, appropriate paraphrase may need some words or sentences whose meanings are close enough to those written in Abstrak.

Key words: Abstract Concepts, Abstract Components, Key Words.

PENGESAHAN

APPROXIMATING SOLUTIONS OF *PARTIAL DIFFERENTIAL EQUATIONS* WITH PSUEDOSPECTRAL METHODS AND *SUPPORT VECTOR MACHINE*

Oleh
Ahmad Izzuddin
NIM 1908919

Program studi Sarjana Fisika
Fakultas Matematika dan Ilmu Pengetahuan Alam
Institut Teknologi Bandung

Menyetujui

Bandung, November 26, 2024
Dosen Pembimbing,

Prof. Dr. Lala Septem Riza, M.T.
NIP. 197809262008121001

Tim Penguji:
1. Nama Penguji 1
2. Nama Penguji 2

PEDOMAN PENGGUNAAN
BUKU TUGAS AKHIR

Buku Tugas Akhir Sarjana ini tidak dipublikasikan, namun terdaftar dan tersedia di Perpustakaan Institut Teknologi Bandung. Buku ini dapat diakses umum, dengan ketentuan bahwa penulis memiliki hak cipta dengan mengikuti aturan HaKI yang berlaku di Institut Teknologi Bandung. Referensi kepustakaan diperkenankan dicatat, tetapi pengutipan atau peringkasan hanya dapat dilakukan seizin penulis, dan harus disertai dengan kebiasaan ilmiah untuk menyebutkan sumbernya.

Memperbanyak atau menerbitkan sebagian atau seluruh buku Tugas Akhir harus atas izin Program Studi Sarjana Fisika, Fakultas Matematika dan Ilmu Pengetahuan Alam, Institut Teknologi Bandung.

Tugas Akhir ini dipersembahkan untuk
Tuhan, Bangsa, dan Almamater

KATA PENGANTAR

Kata pengantar berperan sebagai gerbang masuk bagi pembaca dan mendapat sajian ringkas tentang hal-hal terkait paparan pada buku Tugas Akhir (TA). Sajian ini sejatinya merupakan pengenalan umum bagi pembaca tentang isi tulisan. Hal ini berbeda dengan abstrak yang mendeskripsikan pekerjaan dan hasil penelitian secara lebih teknis.

Kata pengantar merupakan wadah penulis untuk mengenalkan dan mempromosikan pekerjaan dan hasil penelitian dengan bahasa yang sederhana, sehingga pembaca tertarik untuk menelusuri lebih jauh dengan mencermati seluruh paparan pada buku TA. Ini salah satu tujuan kata pengantar. Contoh paragraf yang mengantar pembaca pada isi Buku TA: *Template* LATEX diberikan berikut ini.

Menuliskan pekerjaan dan hasil penelitian TA dalam suatu laporan buku TA memerlukan panduan standar. Panduan ini dibuat dalam beberapa dokumen, yang salah satunya adalah Buku TA: *Template* LATEX. Suatu template adalah cetakan yang siap dituang oleh curahan buah pikiran yang keluar dari pengalaman dalam melakukan pekerjaan penelitian dan hasil-hasilnya. Mencermati cetakan yang memberikan sejumlah contoh dapat memperlancar penulisan laporan tersebut menjadi suatu produk, yaitu buku TA.

Tujuan lain dari Kata Pengantar adalah memberi tempat untuk menyampaikan rasa syukur dan terima kasih kepada banyak pihak, misalnya keluarga, staf akademik, staf tenaga kependidikan, teman, individu atau komunitas pemberi dukungan dan inspirasi, dan institusi pendukung pendanaan seperti pemberi beasiswa atau dana penelitian, atau pendukung akses fasilitas.

Pengorbanan, kegigihan, dedikasi, dan penuh tanggung jawab dari para pahlawan pekerja medis dalam perawatan pasien terpapar Covid-19 telah memberi inspirasi melalui nilai-nilai kejuangan tanpa pamrih. Inspirasi inilah yang membangkitkan spirit pamungkas pada penyelesaian Buku TA: Template LATEX ini. Suatu inspirasi selalu bekerja dan mengena secara tidak langsung. Banyak berterima kasih atas inspirasi yang memantik spirit ini.

Tidak ada sub bab/bagian pada Kata Pengantar, namun daftar rincian diperkenankan. Pada bagian indentitas akhir, seperti berikut ini, dituliskan nama mahasiswa dan NIM, bukan *penulis*, dan tidak perlu ditandatangani. Berikut adalah contoh penulisan rincian yang berisi ucapan terima kasih:

- Prof. Dr. Lala Septem Riza, M.T. dan Dr. Muhammad Nursalman, M.T. selaku dosen pembimbing tugas akhir.

- Jyesta, sebagai teman bimbingan yang selalu bersedia untuk diajak berdiskusi selama penelitian.

- Rekan-rekan Spectranova, ...

Bandung, November 26, 2024

Ahmad Izzuddin

1908919

Contents

DAFTAR NOTASI

Notasi	Arti
$F_{\mu\nu}$	Tensor Elektromagnetik
$R^{\mu}{}_{\alpha\nu\beta}$	Tensor Riemann
$\Gamma^{\rho}_{\mu\nu}$	Simbol Christoffel
$g_{\mu\nu}$	Tensor Metrik
A_{μ}	Medan Gauge
$R_{\mu\nu}$	Tensor Ricci
$\mathscr{L}$	Densitas Lagrangian
$\hbar$	Konstanta Planck Tereduksi
$\mathbb{R}$	Himpunan Bilangan Real

DAFTAR SINGKATAN

Notasi	Arti
FWHM	*Full width half maximum*
rms	*root mean square*
RFS	*Rotary forcespinning*
PVP	Polivinil pirolidon
SI	Satuan Internasional

List of Figures

List of Tables

CHAPTER I

INTRODUCTION

1.1   Background

Partial differential equations (PDE) are a common tool widely used in the modern scientific understanding and many engineering processes. This is because many systems can be described by the way they change and often this can be more intuitive. As an example, think of someone heating up a large frying pan on a gas stove. To describe how the pan heats up when the center is right above the burner, one can say that the center would heat up first followed by its surroundings. Eventually the pan would not heat up any further. Also notice that the edges of the pan would always be cooler than the center. However, once a more complex setup is introduced such as an uneven heat source or more complex materials with different heat rates of transferring heat it becomes much harder to describe how the pan heats up. A more useful way to describe the system is using the heat equation. For a temperature function $u(\mathbf{x}, t)$ of spatial coordinates vector $\mathbf{x}$ and time t, the generalized heat equation is defined in equation (1.1).

$$\frac{\partial u}{\partial t} = \nabla \cdot (\alpha \nabla u) \tag{1.1}$$

The divergence operator $\nabla \cdot$ denotes sum of all first spatial derivatives. In three dimensions this is $\nabla \cdot = \frac{\partial \cdot}{\partial x_1} + \frac{\partial \cdot}{\partial x_2} + \frac{\partial \cdot}{\partial x_3}$. And the gradient operator ∇ is the vector of all first spatial derivatives which in three dimensions is $\nabla f = \begin{bmatrix} \frac{\partial f}{\partial x_1} & \frac{\partial f}{\partial x_2} & \frac{\partial f}{\partial x_3} \end{bmatrix}^\mathsf{T}$. This equation says that the rate at which the temperature changes in time $\frac{\partial u}{\partial t}$ is proportional to how different the differences in temperature of a spot in the pan with its surroundings $\nabla^2 u$ multiplied with thermal diffusivity parameter α. One example of an insight this gives is that given a homogeneous material (α is constant) and the stove outputs heat at a constant rate the temperature will no longer change for a particular point once the temperature difference is constant. In other words once the function $u(\mathbf{x}, t)$ at some time $t > 0$ approaches a linear function in space, the temperature at all spots will no longer change in time. Mathematically this is because the time derivative is zero if the

second spatial derivatives are also zero. Intuitively this is because once the temperature distribution approaches linear, the current spot on the pan is outputting as much heat it is getting. This insight is an example of why PDEs are useful.

Many other fields in physics such as waves, quantum dynamics, fluid dynamics, elastics, and many more also define systems using PDEs. PDEs are also used outside the physical sciences. In Finance, the Black-Scholes-Merton or Black-Scholes equation models the dynamics of the financial market. In ecology, PDEs are used to model population growth which is useful for modelling location dependent carrying capacity. This is used to model species distribution which can help develop better conservation policies. In the social sciences, PDEs such as the cross-diffusion model is used to explore the evolution of the urban environment (Jin et al., 2023). Crowd dynamics is another field where researchers have found it useful to model using PDEs (Hughes, 2000; Mukherjee et al., 2015). These are used to model dynamics such as shock-waves, pathing, and crowd flow as a whole. There are a variety of applications, ranging from bird flocks, pedestrian crowds, to robotic swarms (Gong et al., 2023).

An increasingly critical use of PDEs is in Numerical weather prediction (NWP). The information weather forecasts provide is an integral part of modern life. Many sectors rely on timely and accurate weather forecasts. Individuals, day to day rely on forecasts for a range of different decisions ranging from activities they can do to personal safety from extreme weather. The weather also affects retail as consumer behavior change with whether it is rain or shine (Govind et al., 2020; Moon et al., 2018; J. Tian et al., 2018; X. Tian et al., 2021). Similarly, the types of transportation people use and operational decisions changes with the weather (Lepage & Morency, 2021; Nurmi et al., 2013). Another critical sector is agriculture, where extreme weather events and changes in climate threatens the food supply (Anwar et al., 2013; Cogato et al., 2019; Malhi et al., 2021). As the climate continues to warm and extreme weather events increase in frequency, the mitigation for these events become all the more important (*Climate Change 2022*, 2023). Because of this, forecasting is critical in making accurate decisions for policymakers and warnings of extreme weather events for civilians (Astitha & Nikolopoulos, 2023; Stott et al., 2016). A major challenge is the fact that observations of weather can be sparse in the sense that observations are localized and do not cover entire areas (Galkin et al., 2020; Monmonier, 1999; Wilby & Yu, 2013). This lack of spatially distributed meteorological information

2

hampers decision-making on areas to be prioritized and what to prioritize based on location. For example, weather stations can only observe their immediate surroundings. And they are for the most part stationary. Even more mobile observation platforms like satellites only show the portion of the surface the satellite can see at any one time such as in figure 1.1. While accumulating observations through more satellite passes is possible, this, however, means short-lived features may not be observed. As impoverished regions often are the most data sparse, the effect compounds on the fact that for impoverished regions, scarce resources need to be effectively and efficiently applied. To this end, numerical models are employed to model a more representative view by using known physics such as the compressible Euler equations and statistical models (Kwasniok, 2012; Mengaldo et al., 2019). The next step is in forecasting future weather which in itself is a challenge. The information this could provide is invaluable because it means planning for future weather is possible.
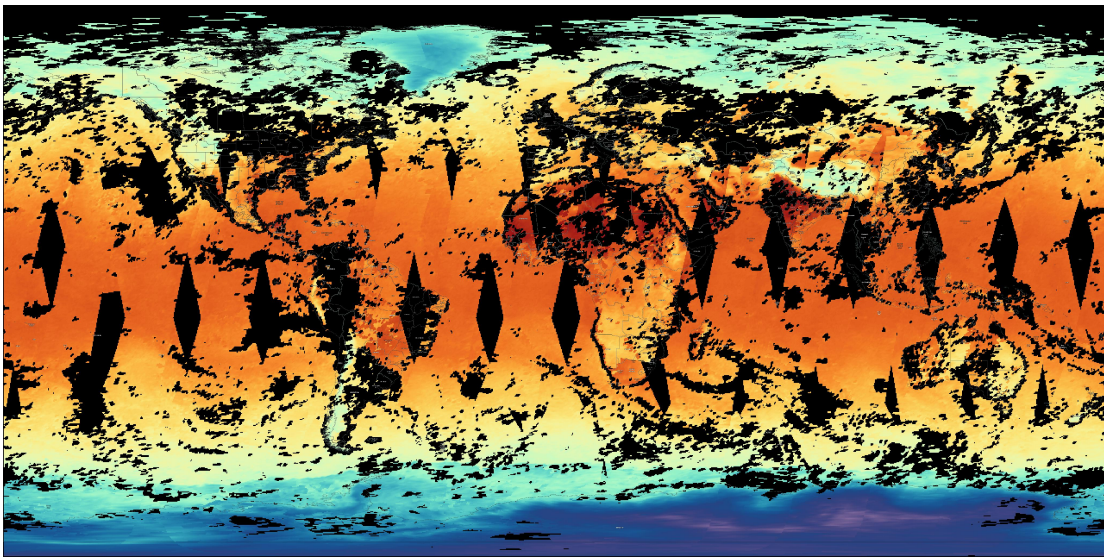
Figure 1.1: Daytime and nighttime surface air temperature from observations by the AIRS instrument onboard the NASA AQUA satellite. This shows observations recorded through a whole day on the 21st of May 2024. Notice that parts of the globe, especially near the equator have not been observed. This image was produced using NASA Worldview (https://go.nasa.gov/46SaYyJ).

Forecasting the weather or evolution of any other system is formulated as an initial value problem where past observations are used to predict an unknown future state of the system. A different formulation can be how heat spreads over time in different types of frying pans from known information of the pan's materials, their

heat conduction properties, and that the stove gives off constant heat. In general the scenario is predicting the effect (future atmospheric temperature distribution, etc.) of some given causes (current and past weather, etc.). This scenario is termed the forward problem. The inverse problem on the other hand solves for an unknown parameter using other known parameters and observations of the partial solution such as temperature field at the object's surface. Traditional methods used to solve these problems utilized knowledge of the exact equations to numerically solve PDEs. As an example, the Finite Difference Method (FDM) approximates the solution by substituting the partial derivatives with finite differences and manipulating the equation such that the solution can be computed. The inverse problem on the other hand is determination of initial conditions or other parameters from observed state of the system. For example determining the heat source distribution given the temperature distribution at some point in time. One traditional approach to this would have meant solving the forward problem from an initial guess of the parameter and then evaluating some cost function such as how close the predicted solution is to the observed one. As expected, this would require many evaluations of the forward solver. This means that the cost of the solver would be greatly multiplied, emphasizing the importance of an efficient solver.

Traditional numerical solvers have enjoyed many decades of development due to their long history. There are many general approaches to solve PDEs and many more very specific approaches. Other than FDM, some widely used approaches include Finite Element Methods (FEM), Finite Volume Methods (FVM), Collocation Methods, and Spectral Methods. FEM and FDM are both mesh based approaches, meaning they rely on discretization of the computational domain. There are several challenges associated with this. First, irregular domains such as bio-inspired materials like bone, spider silk, or aggregate materials like gravel pose a challenge due to the complexity of the domain geometry (Buoni & Petzold, 2007; Gaul et al., 1991; Jia et al., 2024). Also, irregular domains which create large deformations or mesh entanglement cause these methods to become ineffective (J.-S. Chen et al., 2017). While there are strategies to mitigate this, by definition they are an additional layer of difficulty to the process. Second, multiscale applications where the micro and macro scales are both important require fine meshes such that the small scale structures are adequately simulated. This creates meshes with very large number of points that are very resource intensive (Buoni & Petzold, 2007). Third, a single evaluation of traditional mesh-based solvers may not be very costly, however multiple evaluations can add up. This is very apparent

in inverse problems where the solver is queried multiple times to solve the forward problem in order to obtain parameter functions. Therefore, in problems where these issues important or resources are limited, mesh-free methods may be preferred. As an example, the spectral method solves PDEs by formulating the solution as a linear combination of global basis functions like the Fourier series. This can be likened to how music combines different sound waves to produce the overall sound. For some function s, the approximation s^* gets closer to the original with more linear combinations of basis functions. The complex formulation of the Fourier series is presented in equation (1.2). With this approach, the approximated function is characterized completely by its coefficients c_k. This fact is exploited by the spectral method to solve differential equations. The solution function can be approximated by finding the mapping between input function coefficients and solution function coefficients. The mapping between a function s and its derivative s' can be found using the derivative equation from their coefficients leading to $\hat{s}_k = \hat{s}'_k/(2\pi ik)$. Practically this would be done with a finite number of coefficients ($k < \infty$) and the coefficients $\hat{s}'_k$ are obtained using a Discrete Fourier Transform (DFT) algorithm like the Fast Fourier Transform (FFT).

$$s \approx s^* = \sum_k \hat{s}_k e^{2\pi ikx} \tag{1.2}$$

This leads us to the fourth challenge with traditional methods which is that they require prior knowledge of analytic forms of PDEs. This is because traditional solvers use the equations to formulate solutions. This makes traditional numerical approaches unsuited to problems where the governing equations are unknown or partially known. These challenges altogether are some of what weather forecasting faces; NWP has the immense task of modeling multiple scales of the Earth's atmosphere while accounting for features of physical systems that are still not fully understood. Many other fields such as epidemiology (Brauer et al., 2019) and ecology (Holmes et al., 1994; Turchin, 2001) also face these challenges. As with many other fields, at some point in time the governing equations of many systems were not known. This has motivated research into alternative methods that do not completely rely on prior knowledge, are mesh free, and fast enough when solving forward problems.

Machine Learning for PDEs

With the increasing prevalence of machine learning methods and their use in more and more fields, research into their use for scientific computing has taken off in recent years. While statistical modeling has already been widely used in areas such as physical constants, stellar population studies, risk assessments of events such as earthquakes and coronal mass ejections (Anselmo & Pardini, 2005; Berliner, 2003; Bernardi et al., 2022; Reinhardt et al., 2016; Spanos, 2006; Uzan, 2003), the dominant approach for forward modeling or inverse modeling has remained physics based numerical models. Machine learning has provided an alternative approach to model the solutions of PDEs. In their work, Aarts and Van Der Veer (2001) utilized neural networks to approximate each term of PDEs describing damped and undamped free vibrations and substituting them into the PDEs and associated initial conditions. The network parameters were then optimized to reduce the PDE residual and boundary condition loss using evolutionary algorithms. This approach was taken in order to make machine learning models more transparent which at the time was being pursued because of the high cost of optimizing uncertainties in water management numerical simulators. However, since this method approximates the mapping between coordinates and the values of a function and their derivatives, retraining would be necessary for changes to the function itself. This could become very costly as retraining costs accumulate. A more recent approach that also utilizes soft constrains from PDE residuals is termed physics informed neural network (PINN) (Raissi et al., 2019). The authors propose a framework that leverages advancements in computing, namely automatic differentiation (AD) techniques made readily available by modern machine learning libraries. In general, PINNs use AD to compute each term of the PDE from the output of the model and this is then substituted into the PDE to in order to compute the residuals and loss from boundary conditions. The network residual and boundary loss are then weighted and summed with the data loss. For a neural network $\hat{u}(\mathbf{x},t)$ approximating the real solution $u(\mathbf{x},t)$, the residual loss for the heat equation in equation (1.1) is equation (1.3). There are several advantages of incorporating physics knowledge into the model including regularization of the model outputs to be more consistent with physics, faster convergence, and less to no data required depending on whether the network is trained in a manner that is supervised, self-supervised, or a combination of both. One issue with using AD to compute the residual is that the network input needs to be the independent variable (i.e. coordinates, time, etc.). This once again means that if one wants to compute a different

solution, the network needs to be retrained.

$$\mathcal{L}_{PDE} = \left(\frac{\partial \hat{u}(\mathbf{x},t)}{\partial t} - \nabla \cdot (\alpha \nabla \hat{u}(\mathbf{x},t)) \right)^2 \tag{1.3}$$

Learning PDEs with CNNs

Other works utilize convolutional neural networks (CNN) to compute the solution from input functions such as forcing terms or initial conditions. This approach generally means discretizing the functions on a grid and using these as training data. One study by R. Wang et al. (2020) predicts turbulent flow using spatial and temporal decomposition and a specialized U-Net, an architecture based on CNNs, to predict the velocity field from the decomposition of the previous velocity field. Part of the loss function is a regularization term for zero divergence in the velocity field to enforce incompressible fluid flow. This term was calculated using finite differences since auto differentiation is not applicable in this situation. Finite differences was also utilized in another CNN based fluid flow upscaling model by Gao et al. (2021b) to compute the residual terms of the steady incompressible Navier-Stokes equation. This model also inferred unknown physical parameters such as boundary conditions. However, this approach would mean the model would need to scale as a quadratic in 2D, cubic in 3D, and much steeper in higher dimensions. Outside fluid dynamics, the combination of specialized CNNs and finite differences or another numerical differentiation method have been used for many other PDEs including Poisson's equation for temperature fields (Gao et al., 2021a; Zhao et al., 2023), velocity models from seismic data (Muller et al., 2023), and seismic response of structures (Ni et al., 2022; Zhang et al., 2020). While the use of CNNs mean that discretization is implied, solutions of different initial conditions or parameter functions can be computed by inference and no retraining is required. This property is especially useful for many-query problems such as computing gradients for inverse problems.

Operator Learning

The mapping between discretized functions done by CNNs are related to an alternative approach that starts by viewing PDEs as operators, which are generalized mappings between spaces. One group of familiar operators are functions which maps between spaces of scalar values or vector values. PDEs on the other hand are operators

7

that map between function spaces. A simple example is the derivative. The derivative takes in a function and returns the derivative of said function. In other words it is an operator that maps between the space of all functions to the space of derivatives of those functions. Another way to view operators starts by viewing functions as infinite dimensional vectors. Where elements in the vector are the function's value evaluated at every point in space. The operator can be seen as a vector function mapping between these infinite dimensional vector spaces. Operators are important because a field of research has sprung up around this mathematical concept. Operator learning is the use of machine learning to learn operators using data driven approaches. As an analogy, function regression traditionally has been used to approximate the mapping between input values such as coordinates and output values of functions evaluated at said coordinates. In the case of operator learning, the mapping between function spaces are approximated. The aforementioned approaches using CNNs does this directly using the values of functions at discrete points. There are other approaches like DeepONet that does not require the uniform grid like CNNs (Lu et al., 2021). This architecture instead uses both input functions and coordinates as inputs. The output is the output function evaluated at the coordinates provided. This architecture is based on an extension for deep learning of the universal operator approximation theory for neural networks first proposed almost three decades ago at the time of writing by T. Chen and Chen (1995). The proposed architecture is composed of two subnetworks, where one termed the trunk $\hat{\mathbf{T}}(\mathbf{x},t)$ learns the latent mapping for coordinates and the other network termed the branch $\hat{\mathbf{B}}(\mathbf{f})$ learns the latent mapping for the input function f. The two latent mappings are combined through a dot product to obtain the approximated output function value $u(\mathbf{x},t)$. This is formulated in equation (1.4).

$$u(\mathbf{x},t) \approx \hat{G}(\mathbf{f})(\mathbf{x},t) = \hat{\mathbf{B}}(\mathbf{f}) \cdot \hat{\mathbf{T}}(\mathbf{x},t) \tag{1.4}$$

Fourier Neural Operators (FNO) is an alternative avenue for learning operators by utilizing the fact that functions can be decomposed into linear combinations of basis functions, namely trigonometric basis in this particular case (Li et al., 2021). With this method the input function value is mapped to its corresponding output function value. This is done by first lifting the input function value to a higher dimension using a neural network and this is then passed through blocks composed of a Fourier transform, then a linear transform and filtering of higher modes, and finally the inverse Fourier transform. These blocks are stacked to a desired depth and finally another network

projects the outputs to the target dimension. The reason a linear can be used is that differentiation is multiplication in the Fourier domain. One drawback with FNO is the requirement that output functions are not parameterized by coordinates and therefore is implicitly relative to the input function coordinates. To avoid this issue, Fanaskov and Oseledets (2023) reframes the problem by directly utilizing the coefficients of Fourier or Chebyshev basis. The model, termed Spectral Neural Operator (SNO) is trained on features of input function coefficients which are computed using Fourier or Chebyshev transforms and labels of output function coefficients using the same transforms. The authors point out one motivation for this approach which is that training neural networks on discretized data may not be ideal because unexpected outputs such as non-smooth interpolation may happen when the network is trained on one grid size and evaluated other grid sizes. With SNO, the interpolation of the function is smooth due to interpolation being done by Fourier basis functions which are sines and cosines for example. In a similar study, Du et al. (2024) extends the concept of mapping coefficients by proposing residuals in the spectral domain and leveraging Parseval's Identity to compute the spectral analog to the loss term in PINNs. This allows for self supervised learning in the spectral domain. The same benefits incorporating physics into PINNs also apply here without the pain points introduced by discretized model inputs and outputs. As a whole, operator learning creates an alternative approach that addresses the issue of retraining or recomputing the solution model. In addition, due to its data based approach, even systems with partially or fully unknown governing equations may be simulated.

A persistent challenge with all these approaches is the issue of optimization. While neural networks are modular and expressive which is proven by the universal approximation theorem (Cybenko, 1989; Hornik et al., 1989), their loss function present many local minima meaning it is non-convex. This can be mitigated by using advanced optimization techniques that can find a local minima close enough to the global minima such as Adam (Shrestha & Mahmood, 2019; Soydaner, 2020). However, the addition of PDE residuals into the loss function have worsened the highly non-convex loss landscape issue (Basir & Senocak, 2022; Krishnapriyan et al., 2021; Rathore et al., 2024). These problems range from the disparity in size of boundary and residual loss gradients to the fact that incorporation of residuals and boundary conditions themselves create a much more complex loss landscape. As a result, it is desirable to utilize a different machine learning algorithm that possesses a convex

loss landscape. One family of such algorithms are Support Vector Machines (SVM) (Vapnik, 2000). The appeal of SVMs are the fact that the model is formulated as a quadratic programming problem. This means there are strong guarantees for convergence, generalization, and complexity. Another formulation called Least Squares Support Vector Machines (LSSVM) reformulates the problem as a linear system (Suykens, 2005). This leads to an easier problem that can be computed faster by well established algorithms like the many implementations of least squares solvers. Another advantage of the linear formulation is that this can be easily parallelized to exploit hardware like graphics processing units more widely known as GPUs in contrast to the commonly used Sequential Minimal Optimization (SMO) used for SVMs with quadratic objective functions.

The advantageous properties of SVM based methods have attracted research into their use for solving PDEs. An early work using SVMs to solve PDEs by Youxi Wu et al. (2005) introduced a method for solving the forward problem of Electro-Impedance Tomography. This work solved for the mathematical model of EIT which is given by Maxwell's equations by modeling the trial function as using an ε-SVR model. Another approach much more similar to PINNs was presented by Mehrkanoon and Suykens (2015). The residual and initial/boundary conditions are imposed as equality constraints on an LS-SVM objective function. A different study by Leake et al. (2019), the incorporation of physics into the model is done slightly differently by utilizing the theory of functional connections to directly embed constraints into the solution. This means that the proposed method would satisfy the boundary condition exactly. However, the authors point out that for PDEs in higher dimensions deriving and implementing this method can become cumbersome. These approaches, however, do not learn the PDE operator itself. Meaning they are also not practical for many-query problems.

Operator Learning for Weather Forecasting

In terms of weather forecasting, operator learning has been applied in terms of initial value problems. This problem formulation is reminiscent of time series prediction problems widely found in machine learning research. Researchers Kurth et al. (2023) developed FourCastNet which utilized Adaptive Fourier Neural Operator (AFNO), a transformer based model containing the previously mentioned FNO

computational blocks by Li et al. (2021). This model was then able to be trained in a massively parallel manner. In a comparison with a traditional model called the Integrated Forecasting System from the European Center for Medium Range Weather Forecasts (ECMWF), FourCastNet is faster and much more efficient in terms of inference time, resulting in about 80,000 times speed up for a 100-member ensemble forecast. This is while performing much better than a previous deep learning approach. In another study, Bonev et al. (2023) proposed a variation on neural operators called Spherical Fourier Neural Operator (SFNO) which exploited the spherical nature of global forecasting by using Spherical Harmonic Transform (SHT) in place of Fourier Transform. This model when compared to AFNO and FNO, produced no visible artifacts in autoregressive rollouts for long range forecasting. In terms of forecasting, the model shows outcomes that matches the IFS which is a big leap forward in parity for traditional and machine learning based methods.

1.2   Problem Statement

The problems this work sets out to solve based on section 1.1 are:

1. What is the formulation a computational model for operator regression and therefore solving PDEs in the spectral domain using support vector machines?

2. How does the number of basis functions and model parameters impact the model performance?

3. How can one interpret the learned model?

4. Can the model learn to forecast the weather effectively?

1.3   Aims

Based on the stated problems in section 1.2, this study aims to acomplish the following:

1. The design and implementation of a computational model that maps coefficients in the spectral domain using Least Squares Support Vector Regression (LSSVR).

2. Hyperparameter optimization of the model.

3. Interpretation the model results and why some predictions turn out the way they do.

4. Comparison of SpectralSVR with SNO, DeepONet, FNO, and FDM on 4 problems

1.4   Contribution

In achieving the aims of this study the following contributions are made:

1. A novel use LSSVR which has a convex objective to learn solution operators of partial differential equations and operators in general.

2. Comparison of non-uniform Fourier transform for arbitrary spatial sampling with interpolated grid point values.

3. Interpretation of trained model.

1.5   Limitations

This work is limited to the following:

1. Functions the model works with are only continuous functions.

2. The basis functions are limited to Fourier basis.

3. Modeled fields are compact or dense meaning not sparse. Sparse observations or other data are assimilated using other methods.

1.6   Sistematika Penulisan

CHAPTER II

KAJIAN PUSTAKA

Bab ini mengulas secara rinci konsep-konsep dasar yang berkaitan dengn pekerjaan penelitian TA dan deskripsi studi pustaka yang dilakukan. Judul bab tidak harus seperti yang dituliskan, melainkan dapat lebih fleksibel yang mencerminkan isi paparan pada bab ini. Demikian halnya dengan judul sub bab.

2.1 Fourier Transform and Series

Complex numbers are numbers consisting of real and imaginary components. Complex numbers are written as in equation (2.1) with *a* and *b* being the real and imaginary components respectively. The imaginary component is multiplied by the imaginary unit $i = \sqrt{-1}$.

$$c = a + bi \tag{2.1}$$

$$\tag{2.2}$$

2.2 Least Squares Support Vector Machine

An arguably fundamental model widely used whether in pedagogical settings or otherwise is the support vector machine. It dates back to works by Vapnik & Lerner in 1963 (Recognition of Patterns with help of Generalized Portraits) and V. N. Vapnik & A. Ya. Chervonenkis in 1964 (A note on one class of perceptrons/On a perceptron class). As the development on SVM continued, what originally was a model for classification of separable data generalized to regression tasks as well Vapnik (2000 The Nature of Statistical Learning Theory).

2.2.1 Disadvantages

However, the main disadvantage of LSSVMs are the fact that they do not have the sparse property of SVMs which can leave performance on the table. A simple

mitigation can be done by filtering training samples with small absolute values of lagrangian multipliers (Haifeng Wang & Dejin Hu, 2005).

To derive the least squares support vector regression (J.A. Suykens LSSVM Book) model we start with the linear expression:

$$y = W^\mathsf{T}\mathbf{x} + b \tag{2.3}$$

$$\min_{W,e} J(W,e) = \frac{1}{2}W^\mathsf{T}W + C\frac{1}{2}\sum_{k=1}^{n} e_k \tag{2.4}$$

Such that

$$y_k = W^\mathsf{T}\mathbf{x}_k + b + e_k \qquad k = 1,\ldots,n \tag{2.5}$$

$$L(W,b,e;\alpha) = \frac{1}{2}W^\mathsf{T}W + C\frac{1}{2}\sum_{k=1}^{n} e_k + \sum_{k=1}^{n} \alpha_k \left(W^\mathsf{T}\mathbf{x}_k + b + e_k - y_k\right) \tag{2.6}$$

Derive the KKT system

$$
\begin{aligned}
\frac{\partial L}{\partial W} &= 0 \rightarrow W = \sum_{k=1}^{n} \alpha_k x_k \\
\frac{\partial L}{\partial b} &= 0 \rightarrow \sum_{k=1}^{n} \alpha_k = 0 \\
\frac{\partial L}{\partial e_k} &= 0 \rightarrow \alpha_k = Ce_k & k = 1,\ldots,n \\
\frac{\partial L}{\partial \alpha_k} &= 0 \rightarrow W^\mathsf{T}\mathbf{x}_k + b + e_k - y_k = 0 & k = 1,\ldots,n
\end{aligned}
\tag{2.7}
$$

After eliminating W and e, with $\mathbf{1}_n = \langle 1,\ldots,1\rangle$, $\mathbf{y} = [y_1,\ldots,y_n]$, and $\alpha =$

14

$[\alpha_1, \ldots, \alpha_n]$ the solution is as follows in block matrix notation

$$\begin{bmatrix} 0 & \mathbf{1}_n^\mathsf{T} \\ \mathbf{1}_n & \Omega + \frac{I}{C} \end{bmatrix} \begin{bmatrix} b \\ \alpha \end{bmatrix} = \begin{bmatrix} 0 \\ \mathbf{y} \end{bmatrix} \tag{2.8}$$

The solution in eq. (2.8)

Algorithm 1 LSSVR Training

Input: X, y, γ
Output: α, b
1:   $\Omega \leftarrow [[]]$ ▷ Construct matrix of inner products in high dimensional space
2:   **for** $k = 0 \rightarrow n$ **do**
3:      **for** $l = 0 \rightarrow n$ **do**
4:         $\Omega_{k,l} \leftarrow K(\mathbf{X}_k, \mathbf{X}_l)$
5:      **end for**
6:   **end for**
7:   $\mathbf{H} \leftarrow \Omega + \frac{\mathbf{I}}{\gamma}$
8:   $\mathbf{A} \leftarrow [[]]$ ▷ Construct left hand side matrix
9:   $\mathbf{A}_{0,0} \leftarrow 0$
10:   **for** $k = 0 \rightarrow n$ **do**
11:      $\mathbf{A}_{k+1,0} \leftarrow 1$
12:      $\mathbf{A}_{0,k+1} \leftarrow 1$
13:   **end for**
14:   **for** $k = 0 \rightarrow n$ **do**
15:      **for** $l = 0 \rightarrow n$ **do**
16:         $\mathbf{A}_{k+1,l+1} \leftarrow \mathbf{H}_{k,l}$
17:      **end for**
18:   **end for**
19:   $\mathbf{B} \leftarrow []$ ▷ Construct left hand side of the equation
20:   $\mathbf{B}_0 \leftarrow 0$
21:   **for** $k = 0 \rightarrow n$ **do**
22:      $\mathbf{B}_{k+1} \leftarrow \mathbf{y}_k$
23:   **end for**
24:   $\mathbf{A}^\dagger \leftarrow pseudoInverse(\mathbf{A})$ ▷ Compute solution using pseudo inverse
25:   $\mathbf{S} \leftarrow \mathbf{A}^\dagger \mathbf{B}$
26:   $b \leftarrow \mathbf{S}_0$
27:   **for** $k = 0 \rightarrow n$ **do**
28:      $\alpha_k \leftarrow \mathbf{S}_{k+1}$
29:   **end for**

The basic psuedocode from the LSSVM Equation for function regression is defined

in LSSVR Training for a training set of length n, features $\mathbf{X}$, and labels $\mathbf{y}$. Training the LSSVM means computing the values of langrange multipliers α and bias b. K is the kernel function used to compute the inner products in high dimensional space, here we asume the RBF kernel. $\mathbf{A}$ is a matrix of size $n+1$ by $n+1$. $\mathbf{H}$ is a matrix of size n by n. $\mathbf{I}$ is the identity. $\mathbf{B}$ is a vector of size $n+1$. $\mathbf{S}$ is a vector of size $n+1$.

After training the model can be used for prediction of unseen points. The psuedocode for prediction is shown in LSSVR Prediction for prediction features $\mathbf{U}$. The trained model uses the learned multipliers of training points α, the training points themselves $\mathbf{X}$, and the bias b.

Algorithm 2 LSSVR Prediction

Input: $\mathbf{U}, \alpha, \mathbf{X}, b$
Output: $\mathbf{v}$
1: $\Omega \leftarrow [[]]$ $\triangleright$ Construct matrix of inner products in high dimensional space
2: **for** $k = 0 \to m$ **do**
3:      **for** $l = 0 \to n$ **do**
4:          $\Omega_{k,l} \leftarrow K(\mathbf{U}_k, \mathbf{X}_l)$
5:      **end for**
6: **end for**
7: $\mathbf{v} \leftarrow \Omega\alpha + \mathbf{1}_m b$

In this exmple we will be using the function $2x^2 + 4$. The values of this function can be seen in Table 2.1.

No	x	y
1	0.0	4.0
2	0.33...	4.22...
3	0.66...	4.88...
4	1.0	6.0

Table 2.1: Example data of function $2x^2 + 4$

For $\Omega_{i,j} = K(x_i, x_j) = exp\left(-\frac{\|x_i - x_j\|^2}{2\sigma^2}\right)$

For example, with $\sigma = 1$, $x_i = 0.0$, & $x_j = 0.33\ldots$

$$
\begin{aligned}
K(0.0, 0.33) &= exp\left(-\frac{\|0.0 - 0.33\|^2}{2(1)^2}\right) \\
&= exp\left(-\frac{0.33^2}{2}\right) \\
&= 0.9460
\end{aligned}
\tag{2.9}
$$

$$
\Omega \leftarrow
\begin{bmatrix}
1.0000 & 0.9460 & 0.8007 & 0.6065 \\
0.9460 & 1.0000 & 0.9460 & 0.8007 \\
0.8007 & 0.9460 & 1.0000 & 0.9460 \\
0.6065 & 0.8007 & 0.9460 & 1.0000
\end{bmatrix}
\tag{2.10}
$$

$$
\mathbf{I}\frac{1}{\gamma} \rightarrow \mathbf{I}\frac{1}{5} \rightarrow
\begin{bmatrix}
0.2000 & 0.0000 & 0.0000 & 0.0000 \\
0.0000 & 0.2000 & 0.0000 & 0.0000 \\
0.0000 & 0.0000 & 0.2000 & 0.0000 \\
0.0000 & 0.0000 & 0.0000 & 0.2000
\end{bmatrix}
\tag{2.11}
$$

$$
\Omega + \mathbf{I}\frac{1}{5} \rightarrow H \rightarrow
\begin{bmatrix}
1.2000 & 0.9460 & 0.8007 & 0.6065 \\
0.9460 & 1.2000 & 0.9460 & 0.8007 \\
0.8007 & 0.9460 & 1.2000 & 0.9460 \\
0.6065 & 0.8007 & 0.9460 & 1.2000
\end{bmatrix}
\tag{2.12}
$$

$$
A \rightarrow
\begin{bmatrix}
0.0000 & 1.0000 & 1.0000 & 1.0000 & 1.0000 \\
1.0000 & 1.2000 & 0.9460 & 0.8007 & 0.6065 \\
1.0000 & 0.9460 & 1.2000 & 0.9460 & 0.8007 \\
1.0000 & 0.8007 & 0.9460 & 1.2000 & 0.9460 \\
1.0000 & 0.6065 & 0.8007 & 0.9460 & 1.2000
\end{bmatrix}
\tag{2.13}
$$

$$B \rightarrow \begin{bmatrix} 0.0000 \\ 4.0000 \\ 4.2222 \\ 4.8889 \\ 6.0000 \end{bmatrix} \tag{2.14}$$

$$A^{\dagger} \rightarrow \begin{bmatrix} -0.8994 & 0.4348 & 0.0652 & 0.0652 & 0.4348 \\ 0.4348 & 2.0686 & -1.6490 & -0.5292 & 0.1096 \\ 0.0652 & -1.6490 & 3.3774 & -1.1992 & -0.5292 \\ 0.0652 & -0.5292 & -1.1992 & 3.3774 & -1.6490 \\ 0.4348 & 0.1096 & -0.5292 & -1.6490 & 2.0686 \end{bmatrix} \tag{2.15}$$

$$A^{\dagger}B \rightarrow S \rightarrow \begin{bmatrix} 4.9421 \\ -0.6177 \\ -1.3737 \\ -0.5625 \\ 2.5538 \end{bmatrix} \tag{2.16}$$

$$b \rightarrow 4.9421 \tag{2.17}$$

$$\alpha \rightarrow \begin{bmatrix} -0.6177 \\ -1.3737 \\ -0.5625 \\ 2.5538 \end{bmatrix} \tag{2.18}$$

Prediction

$$U \rightarrow \begin{bmatrix} 0.3 \\ 0.2 \\ 0.5 \end{bmatrix} \qquad (2.19)$$

$$\Omega \rightarrow \begin{bmatrix} 0.9560 & 0.9994 & 0.9350 & 0.7827 \\ 0.9802 & 0.9912 & 0.8968 & 0.7261 \\ 0.8825 & 0.9862 & 0.9862 & 0.8825 \end{bmatrix} \qquad (2.20)$$

$$\Omega\alpha \rightarrow \begin{bmatrix} -0.4904 \\ -0.6170 \\ -0.2008 \end{bmatrix} \qquad (2.21)$$

$$\Omega\alpha + b\mathbf{1}_m \rightarrow v \rightarrow \begin{bmatrix} 4.4516 \\ 4.3251 \\ 4.7413 \end{bmatrix} \qquad (2.22)$$

Where $\mathbf{1}_m$ is a vector of 1s with the length of U.

2.3 Sub Bab B

Suatu penelitian tidak dapat lepas dari capaian pengetahuan dan pemahaman yang sudah dipublikasikan. Deskripsi tentang capaian ini menjadi penting karena selain menunjukkan tingkat pemahaman mahasiswa, juga mengetahui tempat pekerjaan penelitian TA dalam konstelasi capaian tersebut. Studi pustaka dan paparan hasilnya dapat memperkaya wawasan tentang topik yang diangkat pada penelitian TA.

2.4 Membuat Persamaan

Secara prinsip, suatu persamaan menyatu dalam kalimat. Letak persamaan dapat berada di awal, tengah, atau akhir kalimat. Dengan demikian, pada akhir persamaan

harus diberikan tanda baca, misalnya koma, titik koma, atau titik, yang menekankan kehadiran persamaan dalam kalimat. Tidak semua persamaan harus diberi nomor. Persamaan yang dirujuk pada naskah TA saja yang harus diberi nomor. Kode awal penomoran ini adalah nomor urut bab, termasuk untuk persamaan pada Lampiran, dengan urutan alfabet kapital.

Setiap notasi harus unik atau tunggal, sehingga arti setiap notasi adalah unik atau tunggal juga. Arti satu notasi harus dituliskan segera ketika notasi tersebut muncul, dan tidak diulang lagi setelahnya.

2.4.1 Contoh Persamaan Sederhana

Persamaan (2.23) mendeskripsikan dinamika fungsi gelombang $\psi(\vec{r}, t)$ di bawah pengaruh potensial $V(\vec{r})$ dan dituliskan sebagai berikut:

$$i\hbar\frac{\partial\psi}{\partial t} = -\frac{\hbar^2}{2m}\vec{\nabla}^2\psi + V(\vec{r})\psi, \tag{2.23}$$

dengan m adalah massa partikel dan $\hbar$ merupakan konstanta Planck tereduksi.

Di akhir persamaan (2.23) diberi koma karena berada ditengah kalimat. Untuk merujuk ke persamaan yang telah ditulis, gunakan perintah \ref{}.

Untuk menuliskan beberapa set persamaan yang masih terhubung, gunakan \subequations{}. Misal kita punya persamaan diferensial terkopel, kita bisa tuliskan

$$\frac{dy}{dt} = -x, \tag{2.24a}$$

$$\frac{dx}{dt} = -y. \tag{2.24b}$$

Kalau perlu matriks, kita bisa tulis seperti berikut.

$$\sigma_x = \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix}, \quad \sigma_y = \begin{pmatrix} 0 & -i \\ i & 0 \end{pmatrix}, \quad \sigma_z = \begin{pmatrix} 1 & 0 \\ 0 & -1 \end{pmatrix} \tag{2.25}$$

2.5 Referensi dan Citation

CHAPTER III

RESEARCH METHODOLOGY

3.1 Research Design

This study is carried out under the framework of a research design. In this section, this design will be laid out and explained. The design includes the process from the beginning to the end of the overall study. The research design is visually depicted in figure 3.1.
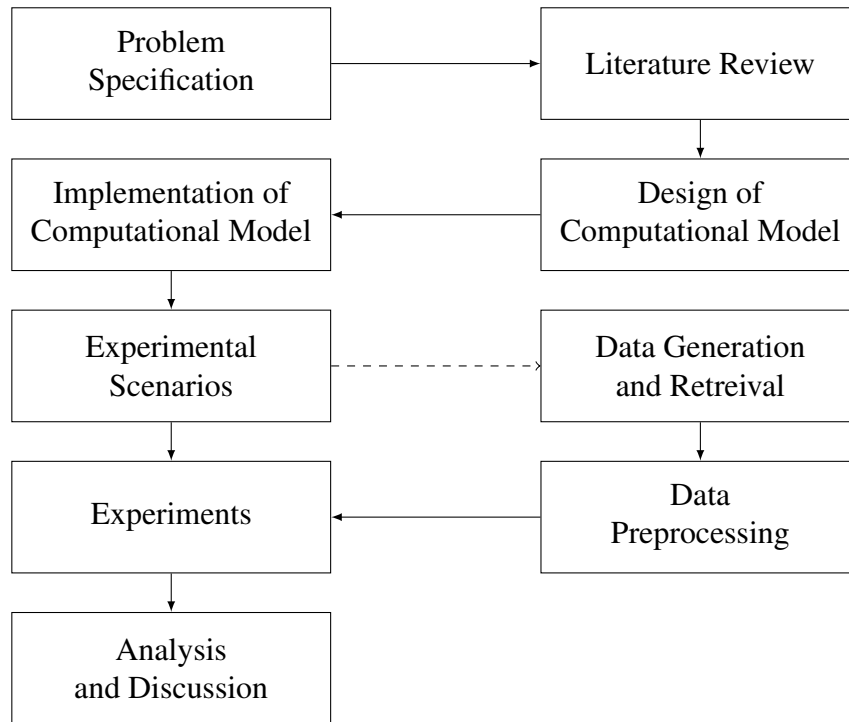

Figure 3.1: Research design diagram

The study is carried out in 9 main phases. Each phase has a specific aim to accomplish such that the subsequent phases are able to proceed. Each phase is explained as follows:

1. Problem Specification

   In this phase, relevant literature of related works and other supporting materials

will be reviewed to lay the groundwork of this study. Specifically, the literature review will encompass partial differential equations and their real world associations, traditional methods employed to solve them, systems with partially known or unknown governing equations, machine learning approaches to the problem, least squares support vector machines, performance metrics, and tools that will be used such as the PyTorch library.

2. Design of Computational Model

This phase of the study is allocated to the design of the computational model based on the literature that has been reviewed. The design of the overall process includes data retrieval, preprocessing, and the core spectral regression model itself.

3. Implementation of Computational Model

After the computational model design is completed, the model is implemented in the Python Language using the Pytorch library as a core component. Implementation of the computational model is done using an iterative software development model. This development model was chosen due to its adaptability to unexpected challenges which is necessary because of the challenges and unknowns with developing a novel method.

4. Planning of Experimental Scenarios

To properly gauge the performance of the proposed method at approximating operators, specifically the solution operator of partial differential equations, experimental scenarios are developed in this phase. Specifically, the scenarios serves three goals which are proof of concept or validation that the method is able to learn operators, a case study of a more complex system with real world data, and a using the model as a surrogate. Together, these experimental scenarios determine the applicability of the model to the problem. In addition, the model will be compared to a baseline model in order to put into context the proposed model's performance. In addition, kernel-input correlation matrices and input-Lagrangian multiplier inner product matrix will also be computed.

5. Data Generation and Retrieval

This phase of the study is tied to the experimental scenarios that have been developed. After determination of the scenario specifics, the data that will be used for training, validation, and testing of the model in each scenario will be either generated or retrieved from an external source. In addition, in each scenario the hyperparameters of the model will be determined using Bayesian optimization as an alternative to the traditional grid search method.

6. Data Preprocessing

   The next phase after data generation or retrieval is preprocessing. This is a crucial step in allowing the core LSSVR to learn the mappings in spectral space. In order to do this, the data will need to be reshaped, and important features selected for. Finally, inputs to the LSSVR will need to be normalized or scaled such that the LSSVR model can much more easily learn the data.

7. Experiments

   This phase of the study executes the experimental scenarios that was determined previously. The preprocessed data will be used in accordance to the preplanned scenarios.

8. Analysis and Discussion

   The final phase of the study is the analysis and discussion of the results. Analysis of each experimental scenario will assess the extent of the model's capabilities. Another component of the analysis is interpretation the trained model and how it comes to the predictions that it makes. The discussion will also touch on the hyperparameter optimization and comparisons with the baseline model. This will show how the model performs differently compared to the baseline.

3.2 Literature Study Method

As a basis for this study, information surrounding the topic is collected from literature sources such as books, journal articles, and other academic works like dissertations. The tools that are used to find these sources include Google Scholar and Google Search. The search terms used start with two terms which are partial differential equations and machine learning. Based on reading the most relevant literature, further search terms

are created from variants of previous search terms combined with terms from literature that has been found.

3.3   Data Generation and Retrieval Method

Data to be used in this study are acquired in two different ways. The first method is data generation. This is motivated by the fact that some partial differential equations do not have much open and accessible real world data available. As such the systems are simulated using the equations themselves. In simple scenarios like computing the antiderivative u of some function f, this can be done by randomly generating u and then taking their derivatives to compute f. The random function generation itself is done by generating random coefficients for basis functions like the Fourier series. Once all functions are generated, random noise is added to ensure that the model is also robust towards inexact measurements. A diagram illustrating the process is shown in figure 3.2.
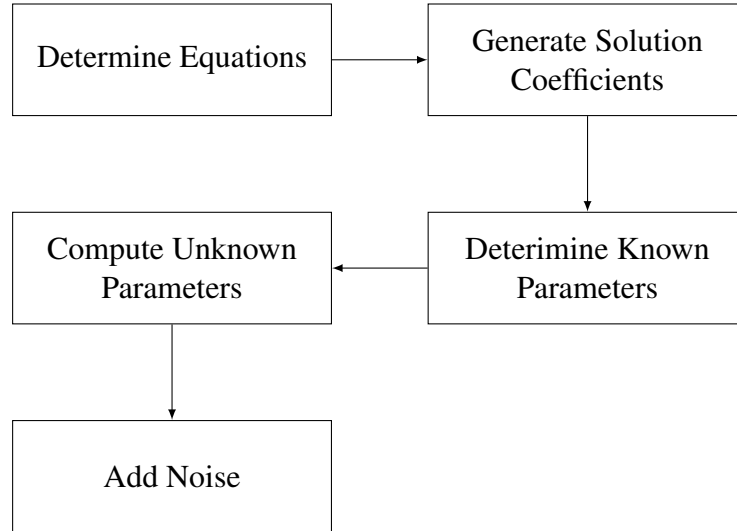

Figure 3.2: Data generation diagram

Data retrieval is the other option which in this case is used to retrieve data for weather prediction. Based on the scenario the data is to be used for, relevant subsets of the dataset is retrieved to a working machine. This study specifically uses a dataset provided by the European Center for Medium Range Weather Forecast (ECMWF) named ERA5 hourly data on single levels from 1940 to present (C3S, 2018). The

first step is selecting a geographical area to study, time period, NetCDF4 data format, and variables such as temperature at 2 meters above the surface. Then the second step is specifications are then used to request the data through the climate data store application programming interface. The downloaded data is then parsed using the Xarray library and preprocessed.

3.4   Computational Model Implementation Method

Implementation of the computational model follows an iterative development model. This model basically consists of repeated cycles or iterations of software development. Each cycle is loosely based on the waterfall development model, namely the processes of requirement gathering, analysis & design, implementation, and testing. This iterative property is crucial for this study because all the requirements cannot be known beforehand. This model minimizes the risk in developing complex software with partially known initial requirements. A diagram of the model can be seen in figure 3.3.
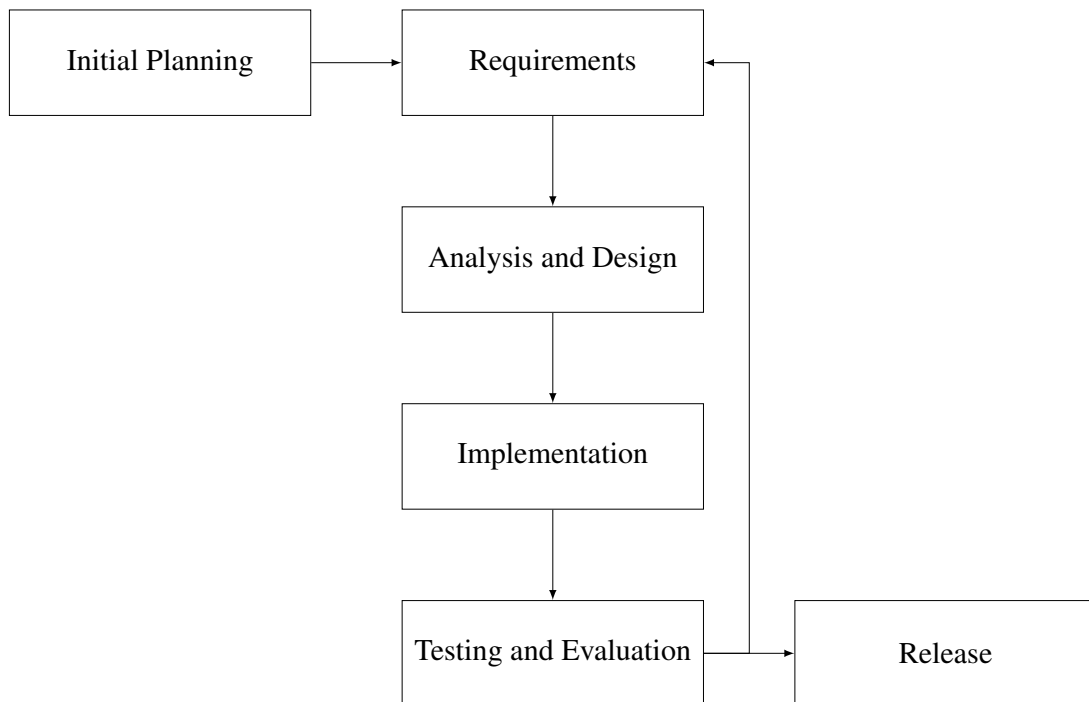

Figure 3.3: Iterative development model

This development model was adapted from Bittner and Spence (2006). Each process in the model is explained as follows:

1. Initial Planning

   The first process of the iterative development model sets up the parameters for the iterative development. These parameters are the broad scope of the project and initial requirements such as core components like LSSVR. The initial requirements are crucial in guiding the rest of the development as more requirements are gathered. The initial planning also determines technical choices like tooling and choice of language. The planning process also includes a rough architecture based in initial requirements.

2. Requirements

   This process separates out the backlog of requirements into those that will be worked on in the current iteration. These requirements ideally share some functionality in order to allow the same context to be retained within the iteration which reduces the burden of context switching.

3. Analysis and Design

   This process an analysis of the requirements of the current iteration is performed. This involves determining the constraints and goals for each requirement. Constraints also need to consider previous iterations such that the requirements can be fulfilled effectively and efficiently without regressing progress that has been made such as breaking existing functionality. From this analysis, the design is updated so that the requirement can be fulfilled. The design itself informs how the code should be structured. The interfaces used in the code are also specified.

4. Implementation

   The implementation process applies the design that has been produced. In addition to implementing the program, this process also implements tests which validate and verify the produced code. In the case of complex pieces of the design, an approach of creating tests and assertions first is taken. This way, the code can be run quickly and fail immediately to allow a quicker convergence on the intended functionality.

5. Testing and Evaluation

   Once the design is implemented, tests that have been created are run. Each

feature is tested in an integrated manner to expedite the process while still providing enough confidence in the software. The results of the testing process are evaluated and any failed tests are resolved whether by reimplementation or reassessment of the correctness of the tests. Any missing functionality or feature discovered during evaluation are added to the backlog of requirements.

6. Release

After an iteration, the current version of the software is released. The iterative nature of this development model necessitates the use of a versioning system. As such each release can be identified by a version number loosely based on semantic versioning (Preston-Werner, n.d.). This version numbering system consists of major number that indicates breaking changes, minor numbers which indicate features and non-breaking changes, and lastly patch numbers are used for fixes. A version number is read as `MAJOR.MINOR.PATCH`. During active development, only major version 0 will be used to prevent too many major versions. The released software is hosted in its own GitHub repository at https://github.com/nidduzzi/SpectralSVR.

3.5 Research Tools

The tools that are used in this study can be categorized into hardware and software. The following are hardware that are used in this study:

- Kaggle CPU Kernel ("Getting Started on Kaggle | Kaggle," n.d.)

    - 4 Cores Intel Xeon

    - 30 Gigabytes RAM

    - 20 Gigabytes Working storage

- Kaggle GPU Kernel ("Getting Started on Kaggle | Kaggle," n.d.)

    - 4 Cores Intel Xeon

    - 29 Gigabytes RAM

    - 1 Nvidia P100 GPU

- 20 Gigabytes Working storage

- Personal Computer

  - Intel i5-8300H

  - 16 Gigabytes RAM

  - Nvidia GeForce GTX 1060 Mobile

  - 1.5 Terabytes Storage

The software tools used in this study are the following:

- Python

- Jupyter Notebook

- Git

- Poetry Python package manager

- Web Browser

- LaTeX

CHAPTER IV

RESULTS AND DISCUSSION

This chapter will elaborate on the results of the study and relevant phases for analysis. This chapter is organized into five sections which are the results of data generation, data retrieval, design of computational model, implementation of computational model, experimental scenarios, and the experiments themselves.

4.1 Data Generation

This study uses two data acquisition approaches. The first approach which is discussed in this section is data generation. This process involves manufacturing data randomly in such a way that they obey the PDEs to be modeled. The generated data consists of features and labels. These features and labels in the context of PDEs are parameters and the solution of the PDEs, respectively. The PDEs enforce a relationship between parameters and the solution. This relationship is implicitly encoded into the features and labels, which is what machine learning models can learn. Because the solution and parameters to PDEs are functions and there are many kinds with different properties, generating all the different kinds of functions is a difficult task. This is why this study focuses on Fourier functions. This means that out of the space of all functions F which include categories such as polynomials $f(x) = a_n x^n + \cdots + a_1 x + a_0$ where $f \in F$, and a_n are constants; we only consider the subset of functions $U \subset F$ which are of the form $u(x) = \sum_k^m c_k e^{2\pi i k \frac{x}{P}}$.

Using the subset of Fourier functions has several benefits which has motivated the choice. First these functions are characterized purely by their coefficients c_k, meaning there is no need to store discretized values which potentially saves space and computation. The second benefit is that other functions such as polynomials can be approximated by them using the Fourier transform. This means that despite limiting the set of functions to Fourier functions, the behavior of PDEs with other sets of functions can be approximated to a certain extent. The final benefit is the mature ecosystem around these functions which include fast algorithms for the Fourier transform and even numerical approaches for solving known PDEs like the previously mentioned spectral

method. In summary the generated data consists of features and labels which are Fourier functions implicitly defining the relationship enforced by a PDE and scenario. After establishing the kind of functions to be used in dataset generation, the process can proceed. There are 4 steps involved in the data generation:

1. Scenario and PDE Determination: The first step of data generation is determining the scenario related to the PDE or governing equations. First, the PDE to be modeled is determined based on the goals of the dataset. Then, one or more of the parameters is chosen to predict the solution. The chosen parameters will be called the input functions and the solution will be called output functions from here on. The second part of the scenario is the domain or more simply the physical space occupied by the system to be modeled. The domain will be used to compute the function values in relation to the physical space from coefficients of Fourier functions.

2. Parameter Determination: The second step is determining all parameters other than that input parameter based on the scenario and governing equation. These parameters may be coefficients such as material properties like density or viscosity, or forcing terms which model external influence on the system like a heat source in the case of the heat equation. Depending on the parameters, solutions of PDEs may behave very differently, such as the appearance of discontinuities in the solution to low viscosity Burgers' equation. Because of this, the choice of parameters is guided by what the dataset seeks to do.

3. Random Coefficient Generation: The third step of data generation is randomly generating the solutions for the chosen equation. Generating random functions in the space of Fourier functions exploits the fact that the coefficients characterize the function completely. By randomly assigning coefficients c_k, many functions can be generated randomly with very little cost. Since the coefficients need to be complex numbers as in equation (2.1), both real and imaginary components are generated independently by assigning a random value to each component for each wave number k. They are then put together again into complex numbers.

Since only real functions are of interest in this study, the generation cost can be approximately halved. This is because for real functions the coefficients for negative wave numbers k are complex conjugate of the positive wave numbers.

This means that once the positive coefficients are generated, one only needs to compute their complex conjugate and concatenate the result with the coefficients of positive wave numbers. For dimensions higher than one, a simpler approach is used. The coefficients are generated for all wave numbers including the negative ones. The inverse Fourier transform is computed and this results in complex functions. The real components of these functions are kept and the Fourier transform is applied to get the coefficients of the real functions. Finally, these generated coefficients then be used with the basis functions as input functions.

4. Forcing Term Computation: Finally, in order to ensure that the generated solution and chosen parameters satisfy the equation, the forcing term is computed as the residual of the equation of interest with the parameters that was previously determined. This computation is done using the spectral method.

5. Function Value and Noise Computation: The generated solution and forcing functions are labeled as input and output functions. The values of both functions in the domain can then be computed using the scenario determined in the first step. Once all coefficients are generated, the function values are computed with an inverse Fourier transform. The real component of the function values are retained, and the imaginary component are zeroed out. Noise is added to the function values here as needed. The processed function values are then converted back into coefficients using a Fourier transform. This processing is necessary to ensure that the coefficients are only describing the real function.

These four steps are the general processes involved in generating datasets for this study. Further specifics of the generation process of each dataset is explained in their respective subsections.

4.1.1 Data Generation: Anti-derivative

The first dataset is a simple one dimensional derivative. This was chosen as a simple proof of concept of the ability to solve a differential equation. The equation is related to many real-world problems such as acceleration and speed. One can imagine a train in an ideal world where acceleration is directly translated into speed. When the train accelerates at time t by some amount a, we can expect the train to have some speed u. In this ideal world, the relationship between speed and acceleration can be modeled

with a simple differential equation (4.1). This scenario is found in many real systems albeit often with many more details such as different components of acceleration from friction, drag, gravity, and other factors. As previously mentioned in section 4.1, both velocity u and acceleration a are modeled with Fourier series in equations (4.2) and (4.3) respectively.

$$\frac{\mathrm{d}u(t)}{\mathrm{d}t} = a(t) \tag{4.1}$$

$$u(t) = \sum_k \hat{u}_k e^{2\pi i k t} \tag{4.2}$$

$$a(t) = \sum_k \hat{a}_k e^{2\pi i k t} \tag{4.3}$$

The domain of the scenario is a two-hour time window. This number was chosen because it is around the ideal length of travel time on high speed rail in comparison to air travel and car travel (Givoni, 2006; Nash, 1991; X. Wang & Zhang, 2019). This is the first step in generating this dataset.

In the second step, as the derivative equation (4.1) does not contain any parameters other than the acceleration which is the input parameter, there are no other parameters to determine. Therefore, the data generation process proceeds to generating coefficients for the speed functions $\hat{u}$. The coefficients are assigned randomly from a Gaussian distribution with a mean of zero and standard deviation of one. This choice was made such that most wave numbers will have a coefficient of close to zero leaving a sparse set of wave numbers to mostly affect the resulting function. In total, 5000 unique functions are generated with 100 complex coefficients each.

In the third step, the output function coefficients are computed. To find the relation between $\hat{u}_k$ and $\hat{a}_k$, we need to find substitutes for each term in equation (4.1). To do this, we take the derivative of equation (4.2) which result in equation (4.4). Using this we can substitute the terms in equation (4.1) with equations (4.3) and (4.4) giving equation (4.5). Finally, after some algebraic manipulation we obtain the relationship between the input a^* and output function u^* in terms of their coefficients in equation (4.6). One also needs to choose the integration constant $\hat{u}_0$ because at $k = 0$

equation (4.6) becomes a division by zero.

$$\frac{du(x)}{dx} = \sum_k \hat{u}_k \times (2\pi ik) e^{2\pi ikx} \tag{4.4}$$

$$\sum_k \hat{u}_k \times (2\pi ik) e^{2\pi ikx} = \sum_k \hat{a}_k e^{2\pi ikx} \tag{4.5}$$

$$\hat{u}_k = \hat{a}_k / (2\pi ik) \tag{4.6}$$

In the final step, using randomly generated values of $\hat{a}_k$, the corresponding values of $\hat{u}_k$ are computed with equation (4.6). These coefficients are then used to compute the function values inside the domain. The two-hour time window is represented by a grid of 500 discrete points. With the coefficients and evaluation points ready, the function values are computed using equations (4.2) and (4.3). Next, noise is added to the function values in order to motivate models learning on the dataset to generalize on noise. To also allow evaluation of how well the model performs with different levels of noise, the samples are duplicated into three copies for each high, medium and low noise levels. The function values of each copy is perturbed with noise from a Gaussian distribution with zero mean and standard deviation of some percentage of the average function value standard deviation. The percentages of each high, medium, and low noise levels are 5, 10, and 50 percent. The perturbed function values are then transformed back into their coefficients. An example generated function and its different perturbed versions is shown in figure 4.1a. The horizontal axis indicates time which is displayed in units of hours. The vertical axis indicates velocity in abstract units.
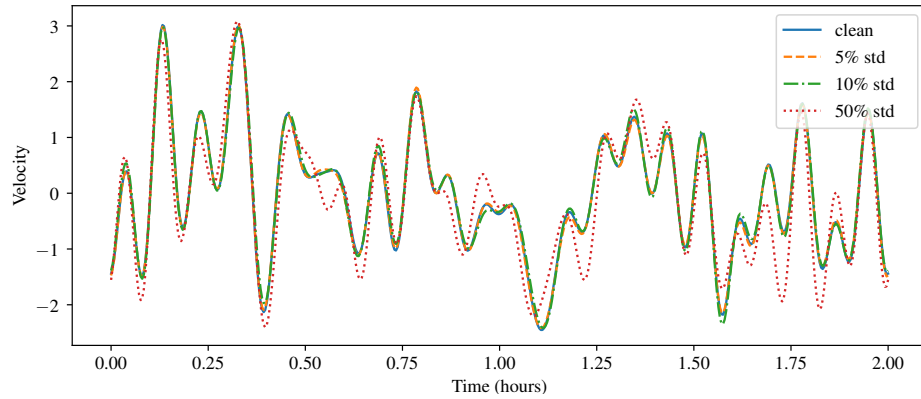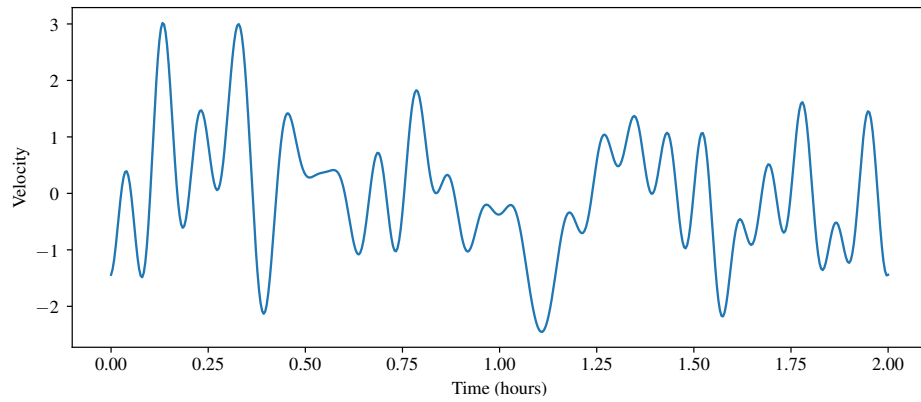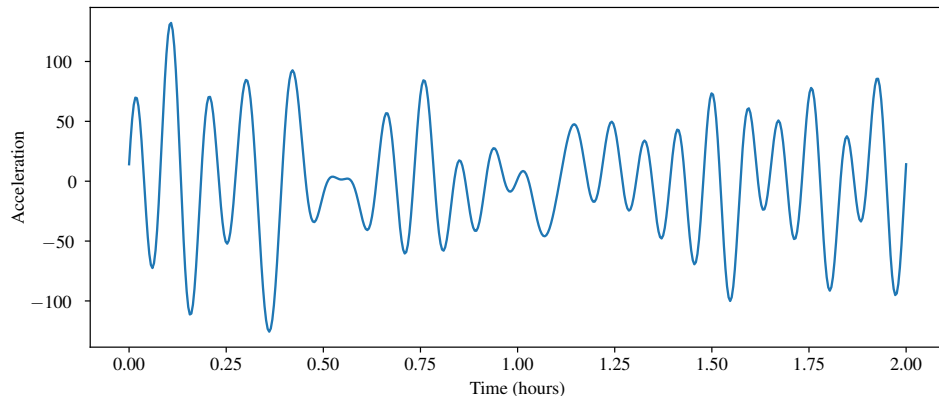
Figure 4.1: (a) A generated function with no noise (clean), low noise level (5%), medium noise level (10%), and high noise levels (50%). (b) Antiderivative function with no noise. (c) Derivative function with no noise.

4.1.2   Data Generation: Burgers' Equation

The second dataset generated in this study concerns the Burgers' equation. This equation has been used to model a variety of cases including fluid dynamics, traffic flow, and, shock waves (Bec & Khanin, 2007; Bonkile et al., 2018; Jameson, 2007; Orlandi, 2000). This equation is also used as a base problem for testing the effectivity of numerical methods in solving non-linear PDEs (Banks et al., 2012; Barter, 2008; Bonkile et al., 2018; Tabatabaei et al., 2007). The nonlinear term in the equation creates steep gradients and even shock waves which are discontinuous with low viscosity conditions. These challenges test the stability of numerical solvers. This is the scenario and reason for the choice of this PDE. To control the viscosity and therefore the steepness of gradients, the formulation of the Burgers' equation considered in this study is the forced viscous Burgers' equation in one dimension. For a velocity of $u(x,t)$, viscosity of v, and forcing term of $f(x,t)$, the formulation of the forced Burgers' equation is shown in equation (4.7).

$$\frac{\partial u}{\partial t} + u\frac{\partial u}{\partial x} - v\frac{\partial^2 u}{\partial x^2} = f \tag{4.7}$$

The domain of we consider here is based on an exact solution of the equation. This is done so that the learned solver model can be tested on the exact solution. There are a number of sources in literature on the exact solution of the Burgers' equation (Benton & Platzman, 1972; Wazwaz, 2010; Wood, 2006). In this study, we use a specific solution by Wood (2006) which can be seen in equation (4.8). This solution is periodic in space but not in time. Therefore, the domain in space will be based on the spatial periodicity of the exact solution which is two. As such, the space domain spans from 0 to 2. The time domain, on the other hand, is chosen to span from 0 to 10 such that enough details of the solution is included.

$$u(x,t) = \frac{2v\pi e^{-\pi^2 vt}\sin(\pi x)}{a + e^{-\pi^2 vt}\cos(\pi x)} \tag{4.8}$$

Continuing to the second step, the parameters involved in equation (4.7) are the viscosity v and the forcing term f. Since the forcing term is dependent on the solution in this step, the only parameter to determine is the viscosity. We choose three constant viscosity values which are 0.1, 0.01, and 0.0. These values are chosen so that a variety of behaviors are represented in the dataset from viscous to inviscid flow.

After this, in the third step, the solution functions u are generated with 20 modes in time and 20 in space and 500 unique function samples for each viscosity value. In total 1500 unique solution functions are generated.

As for the forcing term, it is computed using equation (4.7). The solution field u and forcing term f are modeled with Fourier series in equations (4.9) and (4.10) where k is a vector of spatial and temporal wave number such that k_t is the temporal wave number and k_x is the spatial wave number.

$$u(x,t) = \sum_{k_x}\sum_{k_t} \hat{u}_k e^{2\pi i(k_x x + k_t t)} \tag{4.9}$$

$$f(x,t) = \sum_{k_x}\sum_{k_t} \hat{f}_k e^{2\pi i(k_x x + k_t t)} \tag{4.10}$$

Substituting the terms in equation (4.7) with the respective Fourier series, we get the equation equation (4.12). A problem one notices is the nonlinear term $u\frac{\partial u}{\partial x}$. A naive approach would multiply all wave numbers terms with each other.

$$u\frac{\partial u}{\partial x} = \left(\sum_{k_x}\sum_{k_t} \hat{u}_k e^{2\pi i(k_x x + k_t t)}\right) \times \left(\sum_{k_x}\sum_{k_t} (2\pi i k_x)\hat{u}_k e^{2\pi i(k_x x + k_t t)}\right) \tag{4.11}$$

This is computationally expensive operation with N^2 multiplications (Larios, 2021; Orszag, 1972; Roberts & Bowman, 2011; Shen, 2011). To avoid this complexity, the term is first reformulated into $\frac{\partial}{\partial x}(u^2/2)$. Then to avoid aliasing, the coefficients are padded with 50% zeros such that the padded coefficients are 3/2 times the size of the original (Larios, 2021; Orszag, 1971). Then we use the padded coefficients to compute the function values in the physical domain using the inverse transform. And then, the point wise squaring operation is performed and transform the results back to spectral domain. The resulting coefficients are then trimmed back to their original size before padding. Finally, we multiply the resulting coefficients $\hat{u}u_k$ by the derivative constants. This operation is much more efficient with only 1.5N multiplications. This is still more efficient than the naive approach even after taking into account the transforms involved adds $O(2 \times 1.5N ln(1.5N))$ operations if using a Fast Fourier Transform algorithm.

Putting all the above together, results in equation (4.12).

$$\sum_{k_x}\sum_{k_t}\hat{f}_k e^{2\pi i(k_x x + k_t t)} = \sum_{k_x}\sum_{k_t}(2\pi i k_t)\hat{u}_k e^{2\pi i(k_x x + k_t t)}$$
$$+ \sum_{k_x}\sum_{k_t}(2\pi i k_x)\hat{u}\hat{u}_k e^{2\pi i(k_x x + k_t t)} \qquad (4.12)$$
$$- \nu\sum_{k_x}\sum_{k_t}(2\pi i k_x)^2\hat{u}_k e^{2\pi i(k_x x + k_t t)}$$

$$\sum_{k_x}\sum_{k_t}\hat{f}_k = \sum_{k_x}\sum_{k_t}(2\pi i k_t)\hat{u}_k + (2\pi i k_x)\hat{u}\hat{u}_k - \nu(2\pi i k_x)^2\hat{u}_k \qquad (4.13)$$

$$\hat{f}_k = (2\pi i k_t)\hat{u}_k + (2\pi i k_x)\hat{u}\hat{u}_k - \nu(2\pi i k_x)^2\hat{u}_k \qquad (4.14)$$

After simplifying the equation, we get the coefficients as in equation (4.14). Using this equation, the exact forcing term corresponding to the randomly generated solution can be computed with relatively low cost. Finally, the function values of both the solutions and forcing terms are perturbed by adding Gaussian noise with a mean of zero and standard deviation of 10% of the function value standard deviation to the values of the inverse transform. The perturbed coefficients are then recomputed from the sum and the final perturbed functions are obtained.

4.2 Data Retrieval

The second data acquisition approach retrieves data of a from an external source. This approach downloads the external data to a local machine. In this study, the specific dataset that will be used is the ERA5 dataset from the European Center for Medium-range Weather Forecast (ECMWF). Specifically, the ERA5 dataset is a reanalysis which means it combines observational data from all over the world in order to present a more complete picture of the weather system. This combination process, named 4D variational data assimilation, takes into account the physics known to be involved in the system. As an example weather station data which take measurements such as wind speed, humidity, and temperature of the immediate surroundings of the weather station is very limited in giving a broader picture of weather over a larger area. Even technologies like earth observations satellites are limited to what they can observe at any single point in time as most cannot see the entirety of the earth's

surface at once. The ERA5 dataset specifically, assimilates previous forecasts with observational data every 12 hours. The ERA5 reanalysis is available at the following url (https://cds.climate.copernicus.eu/datasets/reanalysis-era5-single-levels).

The original dataset made available by ECMWF has a grid size of $0.25°$ by $0.25°$ in latitude and longitude for atmospheric data. Data of oceanic waves are also available at a coarser grid size of $0.5°$ by $0.5°$. However, we will use a cloud optimized version called WeatherBench2 that is easier to retrieve (Rasp et al., 2023). This is because the data is available in several more grid sizes such as $1.5°$. In addition, the data can be readily downloaded to a local machine without waiting for further processing on the server. The guide for working with the dataset is available at (https://weatherbench2. readthedocs.io/en/latest/data-guide.html). In our specific case, we use the version labeled `1959-2023_01_10-6h-64x32_equiangular_conservative.zarr`. This dataset spans from year 1959 to 2023 with a grid size of $5.625°$ or resolution of 64 by 32 which spans from $-87.19°$ to $87.19°$ in latitude and $0.0°$ to $354.4°$ in longitude. The dataset is also downsampled to 6 hour intervals that starts at midnight UTC January 1959 1st and ends at 18:00 UTC January 10th 2023.

While the original data provided by ECMWF is available in NetCDF or GRIB, the WeatherBench2 version is provided using the Zarr format in a Google Cloud Storage Bucket. This setup allows the use of libraries that support the Zarr format to access the dataset remotely. The particular library we use is called *Xarray* version `2024.9.0` (Hoyer & Hamman, 2017; Hoyer et al., 2024). Accessing the remote dataset is done as shown in figure 4.2. In order to ensure access is granted, the storage option token is set to anon so that the method uses the anonymous only public access mode of authentication ("GCSFS — GCSFs 2023.12.2post1+1.G8e500c6.Dirty Documentation," n.d.).

```python
import xarray as xr
xr.open_zarr(
    "DATASET_ADDRESS",
    storage_options = {"token": "anon"},
)
```

Figure 4.2: Example of using Xarray to access a publicly accessible Zarr dataset hosted on Google Cloud Storage Bucket.

The dataset includes 62 variables with some variables also spanning 13 discrete vertical levels in the atmosphere. For our use, we will only be using the 2-meter temperature of the atmosphere. This is the air temperature 2 meters above the surface be it land, sea, or inland water. This variable is in units of kelvin. The values of this variable is obtained by interpolating between the model values at the surface and the lowest model vertical level. We also limit the time range to .... An example of the data that was retrieved can be seen in .

4.3 Design of Computational Model

In this section, we will discuss the design of the computational model that is proposed in this work. An illustration of the computational model is show in figure 4.3.
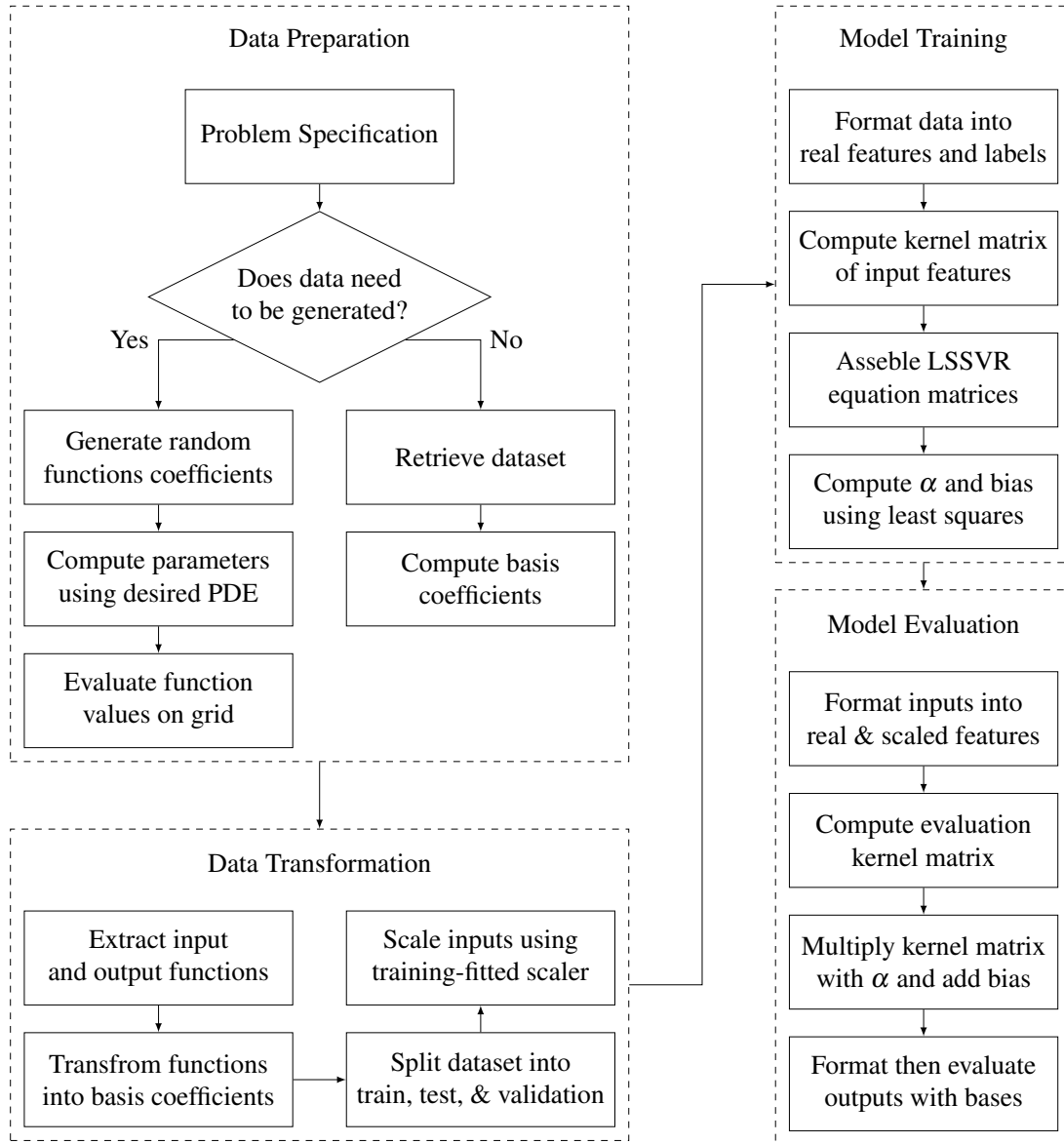
Figure 4.3: Computational Model of SpectralSVR

The computational is divided into four large phases. The processes within these phases are based on the broad outcomes of the phase. The phases themselves are data preparation, data transformation, model training, and model evaluation. Each phase is composed of processes that together works toward the output of the larger process.

4.3.1 Data Preparation

The data preparation phase starts off the computation by preparing the data that is to be used in this study. This phase starts with specifying the problem to be solved. This translates to the dataset that will be used to represent these problems. The first type of dataset is generated based on a known governing equation or PDE. Using said equation, the goal is to generate pairs of functions that are related to each other by the equation. In the next process, generally, the solution function is first generated randomly. This is done by generating random numbers to use as coefficients of a set of basis functions. Then parameters of the equation is determined, such as term coefficients. Using the generated solution and predetermined parameters, a forcing term can generally be computed to satisfy the equation. Augmentations such as noise can then be added in the final process which is evaluating the function values. These processes result in a generated dataset. To verify the generation method and any models trained on the generated data one can use exact analytical solutions of the governing equations.

The second type of dataset is retrieved from an external source. The data source is first identified as the WeatherBench 2 collection of datasets. Then the retrieval process starts. The specific dataset from the collection of different dataset versions is first determined. And then, we determine the filters for the portions of the dataset we are interested in. Once the dataset and filters are determined, the dataset is accessed using the *Xarray* library (Hoyer & Hamman, 2017; Hoyer et al., 2024). The filtered version is then stored locally. Using the stored dataset, the basis coefficients of the data is computed using a discrete Fourier transform.

4.3.2 Data Transformation

4.3.3 Model Training

4.3.4 Model Evaluation

4.4 Implementation of Computational Model

4.5 Experimental Scenarios

4.6 Experimental Results

CHAPTER V

SIMPULAN DAN SARAN

5.1 Simpulan

Bab ini merupakan pamungkas berupa rincian rangkuman yang merupakan simpulan dari analisis yang telah dilakukan. Simpulan ini menyajikan sejumlah hal penting yang disampaikan secara ringkas, padat, dan utuh, yang menjawab tujuan penelitian yang dituliskan pada Bab Pendahuluan. Sangat mungkin ada beberapa konsekuensi dan implikasi yang ditimbulkan dari simpulan yang dihasilkan, yang sepatutnya menjadi perhatian pada penelitian berikutnya. Judul bab dapat disesuaikan, namun umumnya ada *Simpulan* yang memang mendominasi isi bab ini.

5.2 Saran

Sejumlah ide yang muncul ketika melaksanakan penelitian TA dapat menjadi bahan atau topik untuk pekerjaan selanjutnya. Hal ini dapat berupa perbaikan atau ragam lain dari apa yang telah dilakukan sepanjang penelitian. Sub bab ini menjadi sumber informasi penting bagi, utamanya mahasiswa, yang akan melakukan penelitian lanjutan.

Bibliography

Aarts, L. P., & Van Der Veer, P. (2001). Neural Network Method for Solving Partial Differential Equations. *Neural Processing Letters*, *14*(3), 261–271. https://doi.org/10.1023/A:1012784129883

Anselmo, L., & Pardini, C. (2005). Computational methods for reentry trajectories and risk assessment. *Advances in Space Research*, *35*(7), 1343–1352. https://doi.org/10.1016/j.asr.2005.04.089

Anwar, M. R., Liu, D. L., Macadam, I., & Kelly, G. (2013). Adapting agriculture to climate change: A review. *Theoretical and Applied Climatology*, *113*(1-2), 225–245. https://doi.org/10.1007/s00704-012-0780-1

Astitha, M., & Nikolopoulos, E. (Eds.). (2023). *Extreme weather forecasting : State of the science, uncertainty and impacts*. Elsevier.
OCLC: 1347429101.

Banks, J., Hittinger, J., Connors, J., & Woodward, C. (2012). Numerical error estimation for nonlinear hyperbolic PDEs via nonlinear error transport. *Computer Methods in Applied Mechanics and Engineering*, *213–216*, 1–15. https://doi.org/10.1016/j.cma.2011.11.021

Barter, G. E. (2008). *Shock capturing with PDE-based artificial viscosity for an adaptive, higher-order discontinuous Galerkin finite element method* [Doctoral dissertation, Massachusetts Institute of Technology]. Retrieved November 9, 2024, from http://dspace.mit.edu/handle/1721.1/7582

Basir, S., & Senocak, I. (2022). Critical Investigation of Failure Modes in Physics-informed Neural Networks. *AIAA SCITECH 2022 Forum*. https://doi.org/10.2514/6.2022-2353

Bec, J., & Khanin, K. (2007). Burgers turbulence. *Physics Reports*, *447*(1-2), 1–66. https://doi.org/10.1016/j.physrep.2007.04.002

Benton, E. R., & Platzman, G. W. (1972). A table of solutions of the one-dimensional Burgers equation. *Quarterly of Applied Mathematics*, *30*(2), 195–212. https://doi.org/10.1090/qam/306736

Berliner, L. M. (2003). Physical-statistical modeling in geophysics. *Journal of Geophysical Research: Atmospheres*, *108*(D24), 2002JD002865. https://doi.org/10.1029/2002JD002865

Bernardi, M., Sánchez, H. D., Sheth, R. K., Brownstein, J. R., & Lane, R. R. (2022). Stellar population analysis of MaNGA early-type galaxies: IMF dependence and systematic effects. *Monthly Notices of the Royal Astronomical Society*, *518*(3), 4713–4733. https://doi.org/10.1093/mnras/stac3287

Bittner, K., & Spence, I. (2006). *Managing Iterative Software Development Projects*. Pearson Education, Limited.
OCLC: 1348491270.

Bonev, B., Kurth, T., Hundt, C., Pathak, J., Baust, M., Kashinath, K., & Anandkumar, A. (2023, July 23–29). Spherical Fourier neural operators: Learning stable dynamics on the sphere. In A. Krause, E. Brunskill, K. Cho, B. Engelhardt, S. Sabato, & J. Scarlett (Eds.), *Proceedings of the 40th international conference on machine learning* (pp. 2806–2823, Vol. 202). PMLR. https://proceedings.mlr.press/v202/bonev23a.html

Bonkile, M. P., Awasthi, A., Lakshmi, C., Mukundan, V., & Aswin, V. S. (2018). A systematic literature review of Burgers' equation with recent advances. *Pramana*, *90*(6), 69. https://doi.org/10.1007/s12043-018-1559-4

Brauer, F., Castillo-Chávez, C., & Feng, Z. (2019). *Mathematical models in epidemiology*. Springer.

Buoni, M., & Petzold, L. (2007). An efficient, scalable numerical algorithm for the simulation of electrochemical systems on irregular domains. *Journal of Computational Physics*, *225*(2), 2320–2332. https://doi.org/10.1016/j.jcp.2007.03.025

C3S. (2018). *ERA5 hourly data on single levels from 1940 to present*. https://doi.org/10.24381/CDS.ADBB2D47

Chen, J.-S., Hillman, M., & Chi, S.-W. (2017). Meshfree Methods: Progress Made after 20 Years. *Journal of Engineering Mechanics*, *143*(4), 04017001. https://doi.org/10.1061/(ASCE)EM.1943-7889.0001176

Chen, T., & Chen, H. (1995). Universal approximation to nonlinear operators by neural networks with arbitrary activation functions and its application to dynamical systems. *IEEE Transactions on Neural Networks*, *6*(4), 911–917. https://doi.org/10.1109/72.392253

Climate change 2022: Impacts, adaptation and vulnerability : Working Group II contribution to the Sixth Assessment Report of the Intergovernmental Panel on

Climate Change. (2023). Cambridge University Press.
OCLC: 1391150208.

Cogato, A., Meggio, F., De Antoni Migliorati, M., & Marinello, F. (2019). Extreme Weather Events in Agriculture: A Systematic Review. *Sustainability*, *11*(9), 2547. https://doi.org/10.3390/su11092547

Cybenko, G. (1989). Approximation by superpositions of a sigmoidal function. *Mathematics of Control, Signals, and Systems*, *2*(4), 303–314. https://doi.org/10.1007/BF02551274

Du, Y., Chalapathi, N., & Krishnapriyan, A. S. (2024). Neural spectral methods: Self-supervised learning in the spectral domain. *The Twelfth International Conference on Learning Representations*. https://openreview.net/forum?id=2DbVeuoa6a

Fanaskov, V. S., & Oseledets, I. V. (2023). Spectral Neural Operators. *Doklady Mathematics*, *108*(S2), S226–S232. https://doi.org/10.1134/S1064562423701107

Galkin, I. A., Reinisch, B. W., Vesnin, A. M., Bilitza, D., Fridman, S., Habarulema, J. B., & Veliz, O. (2020). Assimilation of sparse continuous near-earth weather measurements by NECTAR model morphing. *Space weather : the international journal of research & applications*, *18*(11), e2020SW002463. https://doi.org/10.1029/2020SW002463

Gao, H., Sun, L., & Wang, J.-X. (2021a). PhyGeoNet: Physics-informed geometry-adaptive convolutional neural networks for solving parameterized steady-state PDEs on irregular domain. *Journal of Computational Physics*, *428*, 110079. https://doi.org/10.1016/j.jcp.2020.110079

Gao, H., Sun, L., & Wang, J.-X. (2021b). Super-resolution and denoising of fluid flow using physics-informed convolutional neural networks without high-resolution labels. *Physics of Fluids*, *33*(7), 073603. https://doi.org/10.1063/5.0054312

Gaul, L., Klein, P., & Plenge, M. (1991). Simulation of wave propagation in irregular soil domains by BEM and associated small scale experiments. *Engineering Analysis with Boundary Elements*, *8*(4), 200–205. https://doi.org/10.1016/0955-7997(91)90014-K

GCSFS — GCSFs 2023.12.2post1+1.g8e500c6.dirty documentation. (n.d.). Retrieved November 26, 2024, from https://gcsfs.readthedocs.io/en/latest/

Getting Started on Kaggle | Kaggle. (n.d.). Retrieved September 9, 2024, from https://www.kaggle.com/docs/notebooks#technical-specifications

Givoni, M. (2006). Development and Impact of the Modern High-speed Train: A Review. *Transport Reviews*, *26*(5), 593–611. https://doi.org/10.1080/01441640600589319

Gong, X., Herty, M., Piccoli, B., & Visconti, G. (2023). Crowd Dynamics: Modeling and Control of Multiagent Systems. *Annual Review of Control, Robotics, and Autonomous Systems*, *6*(1), 261–282. https://doi.org/10.1146/annurev-control-060822-123629

Govind, R., Garg, N., & Mittal, V. (2020). Weather, Affect, and Preference for Hedonic Products: The Moderating Role of Gender. *Journal of Marketing Research*, *57*(4), 717–738. https://doi.org/10.1177/0022243720925764

Haifeng Wang & Dejin Hu. (2005). Comparison of SVM and LS-SVM for Regression. *2005 International Conference on Neural Networks and Brain*, *1*, 279–283. https://doi.org/10.1109/ICNNB.2005.1614615

Holmes, E. E., Lewis, M. A., Banks, J. E., & Veit, R. R. (1994). Partial Differential Equations in Ecology: Spatial Interactions and Population Dynamics. *Ecology*, *75*(1), 17–29. https://doi.org/10.2307/1939378

Hornik, K., Stinchcombe, M., & White, H. (1989). Multilayer feedforward networks are universal approximators. *Neural Networks*, *2*(5), 359–366. https://doi.org/10.1016/0893-6080(89)90020-8

Hoyer, S., & Hamman, J. (2017). Xarray: N-D labeled arrays and datasets in Python. *Journal of Open Research Software*, *5*(1). https://doi.org/10.5334/jors.148

Hoyer, S., Roos, M., Joseph, H., Magin, J., Cherian, D., Fitzgerald, C., Hauser, M., Fujii, K., Maussion, F., Imperiale, G., Clark, S., Kleeman, A., Nicholas, T., Kluyver, T., Westling, J., Munroe, J., Amici, A., Barghini, A., Banihirwe, A., … Littlejohns, O. (2024, September 11). *Xarray* (Version v2024.09.0). https://doi.org/10.5281/ZENODO.13750907

Hughes, R. (2000). The flow of large crowds of pedestrians. *Mathematics and Computers in Simulation*, *53*(4-6), 367–370. https://doi.org/10.1016/S0378-4754(00)00228-7

Jameson, A. (2007). Energy Estimates for Nonlinear Conservation Laws with Applications to Solutions of the Burgers Equation and One-Dimensional Vicous Flow in a Shock Tube by Central Difference Schemes. *18th AIAA*

Computational Fluid Dynamics Conference. https://doi.org/10.2514/6.2007-4620

Jia, Y., Liu, K., & Zhang, X. S. (2024). Modulate stress distribution with bio-inspired irregular architected materials towards optimal tissue support. *Nature Communications*, *15*(1), 4072. https://doi.org/10.1038/s41467-024-47831-2

Jin, M., Wang, L., Ge, F., & Yan, J. (2023). Detecting the interaction between urban elements evolution with population dynamics model. *Scientific Reports*, *13*(1), 12367. https://doi.org/10.1038/s41598-023-38979-w

Krishnapriyan, A., Gholami, A., Zhe, S., Kirby, R., & Mahoney, M. W. (2021). Characterizing possible failure modes in physics-informed neural networks. In M. Ranzato, A. Beygelzimer, Y. Dauphin, P. Liang, & J. W. Vaughan (Eds.), *Advances in neural information processing systems* (pp. 26548–26560, Vol. 34). Curran Associates, Inc. https://proceedings.neurips.cc/paper_files/paper/2021/file/df438e5206f31600e6ae4af72f2725f1-Paper.pdf

Kurth, T., Subramanian, S., Harrington, P., Pathak, J., Mardani, M., Hall, D., Miele, A., Kashinath, K., & Anandkumar, A. (2023). FourCastNet: Accelerating Global High-Resolution Weather Forecasting Using Adaptive Fourier Neural Operators. *Proceedings of the Platform for Advanced Scientific Computing Conference*, 1–11. https://doi.org/10.1145/3592979.3593412

Kwasniok, F. (2012). Data-based stochastic subgrid-scale parametrization: An approach using cluster-weighted modelling. *Philosophical Transactions of the Royal Society A: Mathematical, Physical and Engineering Sciences*, *370*(1962), 1061–1086. https://doi.org/10.1098/rsta.2011.0384

Larios, A. (2021, April 28). *MATH 934 – BURGERS EQUATION PROJECT* [handout]. Retrieved September 16, 2024, from https://www.math.unl.edu/~alarios2/courses/2017_spring_M934/documents/burgersProject.pdf

Leake, C., Johnston, H., Smith, L., & Mortari, D. (2019). Analytically Embedding Differential Equation Constraints into Least Squares Support Vector Machines Using the Theory of Functional Connections. *Machine Learning and Knowledge Extraction*, *1*(4), 1058–1083. https://doi.org/10.3390/make1040060

Lepage, S., & Morency, C. (2021). Impact of Weather, Activities, and Service Disruptions on Transportation Demand. *Transportation Research Record:*

Journal of the Transportation Research Board, *2675*(1), 294–304. https://doi.org/10.1177/0361198120966326

Li, Z., Kovachki, N. B., Azizzadenesheli, K., liu, B., Bhattacharya, K., Stuart, A., & Anandkumar, A. (2021). Fourier neural operator for parametric partial differential equations. *International Conference on Learning Representations*. https://openreview.net/forum?id=c8P9NQVtmnO

Lu, L., Jin, P., Pang, G., Zhang, Z., & Karniadakis, G. E. (2021). Learning nonlinear operators via DeepONet based on the universal approximation theorem of operators. *Nature Machine Intelligence*, *3*(3), 218–229. https://doi.org/10.1038/s42256-021-00302-5

Malhi, G. S., Kaur, M., & Kaushik, P. (2021). Impact of Climate Change on Agriculture and Its Mitigation Strategies: A Review. *Sustainability*, *13*(3), 1318. https://doi.org/10.3390/su13031318

Mehrkanoon, S., & Suykens, J. A. (2015). Learning solutions to partial differential equations using LS-SVM. *Neurocomputing*, *159*, 105–116. https://doi.org/10.1016/j.neucom.2015.02.013

Mengaldo, G., Wyszogrodzki, A., Diamantakis, M., Lock, S.-J., Giraldo, F. X., & Wedi, N. P. (2019). Current and Emerging Time-Integration Strategies in Global Numerical Weather and Climate Prediction. *Archives of Computational Methods in Engineering*, *26*(3), 663–684. https://doi.org/10.1007/s11831-018-9261-8

Monmonier, M. S. (1999). *Air apparent: How meteorologists learned to map, predict, and dramatize weather*. Univ. of Chicago Press.

Moon, S., Kang, M. Y., Bae, Y. H., & Bodkin, C. D. (2018). Weather sensitivity analysis on grocery shopping. *International Journal of Market Research*, *60*(4), 380–393. https://doi.org/10.1177/1470785317751614

Mukherjee, S., Goswami, D., & Chatterjee, S. (2015). A Lagrangian Approach to Modeling and Analysis of a Crowd Dynamics. *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, *45*(6), 865–876. https://doi.org/10.1109/TSMC.2015.2389763

Muller, A. P. O., Costa, J. C., Bom, C. R., Klatt, M., Faria, E. L., de Albuquerque, M. P., & de Albuquerque, M. P. (2023). Deep pre-trained FWI: Where supervised learning meets the physics-informed neural networks. *Geophysical Journal International*, *235*(1), 119–134. https://doi.org/10.1093/gji/ggad215

Nash, C. (1991, January). *The case for high speed rail* (Working Paper No. Working Paper 323). Institute of Transport Studies, University of Leeds / ARRAY(0x5581272c4498). Leeds, UK. https://eprints.whiterose.ac.uk/2236/

Ni, P., Sun, L., Yang, J., & Li, Y. (2022). Multi-End Physics-Informed Deep Learning for Seismic Response Estimation. *Sensors*, *22*(10), 3697. https://doi.org/10.3390/s22103697

Nurmi, P., Perrels, A., & Nurmi, V. (2013). Expected impacts and value of improvements in weather forecasting on the road transport sector. *Meteorological Applications*, *20*(2), 217–223. https://doi.org/10.1002/met.1399

Orlandi, P. (2000). The Burgers equation. In P. Orlandi (Ed.). R. Moreau (**typeredactor**), *Fluid Flow Phenomena* (pp. 40–50, Vol. 55). Springer Netherlands. https://doi.org/10.1007/978-94-011-4281-6_4

Orszag, S. A. (1971). On the Elimination of Aliasing in Finite-Difference Schemes by Filtering High-Wavenumber Components. *Journal of the Atmospheric Sciences*, *28*(6), 1074–1074. https://doi.org/10.1175/1520-0469(1971)028<1074: OTEOAI>2.0.CO;2

Orszag, S. A. (1972). Comparison of Pseudospectral and Spectral Approximation. *Studies in Applied Mathematics*, *51*(3), 253–259. https://doi.org/10.1002/sapm1972513253

Preston-Werner, T. (n.d.). *Semantic Versioning 2.0.0*. Semantic Versioning. Retrieved September 9, 2024, from https://semver.org/

Raissi, M., Perdikaris, P., & Karniadakis, G. (2019). Physics-informed neural networks: A deep learning framework for solving forward and inverse problems involving nonlinear partial differential equations. *Journal of Computational Physics*, *378*, 686–707. https://doi.org/10.1016/j.jcp.2018.10.045

Rasp, S., Hoyer, S., Merose, A., Langmore, I., Battaglia, P., Russel, T., Sanchez-Gonzalez, A., Yang, V., Carver, R., Agrawal, S., Chantry, M., Bouallegue, Z. B., Dueben, P., Bromberg, C., Sisk, J., Barrington, L., Bell, A., & Sha, F. (2023). WeatherBench 2: A benchmark for the next generation of data-driven global weather models.

Rathore, P., Lei, W., Frangella, Z., Lu, L., & Udell, M. (2024). *Challenges in Training PINNs: A Loss Landscape Perspective* (2). https://doi.org/10.48550/ARXIV.2402.01868

Reinhardt, J. C., Chen, X., Liu, W., Manchev, P., & Paté-Cornell, M. E. (2016). Asteroid Risk Assessment: A Probabilistic Approach. *Risk Analysis*, *36*(2), 244–261. https://doi.org/10.1111/risa.12453

Roberts, M., & Bowman, J. C. (2011). Dealiased convolutions for pseudospectral simulations. *Journal of Physics: Conference Series*, *318*(7), 072037. https://doi.org/10.1088/1742-6596/318/7/072037

Shen, J. (2011). *Spectral Methods: Algorithms, Analysis and Applications*. Springer. https://doi.org/10.1007/978-3-540-71041-7

Shrestha, A., & Mahmood, A. (2019). Review of Deep Learning Algorithms and Architectures. *IEEE Access*, *7*, 53040–53065. https://doi.org/10.1109/ACCESS.2019.2912200

Soydaner, D. (2020). A Comparison of Optimization Algorithms for Deep Learning. *International Journal of Pattern Recognition and Artificial Intelligence*, *34*(13), 2052013. https://doi.org/10.1142/S0218001420520138

Spanos, A. (2006). Where do statistical models come from? Revisiting the problem of specification. In *Institute of Mathematical Statistics Lecture Notes - Monograph Series* (pp. 98–119). Institute of Mathematical Statistics. https://doi.org/10.1214/074921706000000419

Stott, P. A., Christidis, N., Otto, F. E. L., Sun, Y., Vanderlinden, J.-P., van Oldenborgh, G. J., Vautard, R., von Storch, H., Walton, P., Yiou, P., & Zwiers, F. W. (2016). Attribution of extreme weather and climate-related events. *WIREs Climate Change*, *7*(1), 23–41. https://doi.org/10.1002/wcc.380

Suykens, J. A. K. (Ed.). (2005). *Least squares support vector machines* (Repr). World Scientific.

Tabatabaei, A. H. A., Shakour, E., & Dehghan, M. (2007). Some implicit methods for the numerical solution of Burgers' equation. *Applied Mathematics and Computation*, *191*(2), 560–570. https://doi.org/10.1016/j.amc.2007.02.158

Tian, J., Zhang, Y., & Zhang, C. (2018). Predicting consumer variety-seeking through weather data analytics. *Electronic Commerce Research and Applications*, *28*, 194–207. https://doi.org/10.1016/j.elerap.2018.02.001

Tian, X., Cao, S., & Song, Y. (2021). The impact of weather on consumer behavior and retail performance: Evidence from a convenience store chain in China. *Journal of Retailing and Consumer Services*, *62*, 102583. https://doi.org/10.1016/j.jretconser.2021.102583

Turchin, P. (2001). Does population ecology have general laws? *Oikos*, *94*(1), 17–26. https://doi.org/10.1034/j.1600-0706.2001.11310.x

Uzan, J.-P. (2003). The fundamental constants and their variation: Observational and theoretical status. *Reviews of Modern Physics*, *75*(2), 403–455. https://doi.org/10.1103/RevModPhys.75.403

Vapnik, V. N. (2000). *The Nature of Statistical Learning Theory* (2nd ed). Springer New York. https://doi.org/10.1007/978-1-4757-3264-1

Wang, R., Kashinath, K., Mustafa, M., Albert, A., & Yu, R. (2020). Towards Physics-informed Deep Learning for Turbulent Flow Prediction. *Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, 1457–1466. https://doi.org/10.1145/3394486.3403198

Wang, X., & Zhang, W. (2019). Efficiency and Spatial Equity Impacts of High-Speed Rail on the Central Plains Economic Region of China. *Sustainability*, *11*(9), 2583. https://doi.org/10.3390/su11092583

Wazwaz, A.-M. (2010). *Partial Differential Equations and Solitary Waves Theory*. Springer Berlin Heidelberg. https://doi.org/10.1007/978-3-642-00251-9

Wilby, R. L., & Yu, D. (2013). Rainfall and temperature estimation for a data sparse region. *Hydrology and Earth System Sciences*, *17*(10), 3937–3955. https://doi.org/10.5194/hess-17-3937-2013

Wood, W. L. (2006). An exact solution for Burger's equation. *Communications in Numerical Methods in Engineering*, *22*(7), 797–798. https://doi.org/10.1002/cnm.850

Youxi Wu, Ying Li, Lei Guo, Weili Yan, Xueqin Shen, & Kun Fu. (2005). SVM for Solving Forward Problems of EIT. *2005 IEEE Engineering in Medicine and Biology 27th Annual Conference*, 1559–1562. https://doi.org/10.1109/IEMBS.2005.1616732

Zhang, R., Liu, Y., & Sun, H. (2020). Physics-guided convolutional neural network (PhyCNN) for data-driven seismic response modeling. *Engineering Structures*, *215*, 110704. https://doi.org/10.1016/j.engstruct.2020.110704

Zhao, X., Gong, Z., Zhang, Y., Yao, W., & Chen, X. (2023). Physics-informed convolutional neural networks for temperature field prediction of heat source layout without labeled data. *Engineering Applications of Artificial Intelligence*, *117*, 105516. https://doi.org/10.1016/j.engappai.2022.105516

APPENDIX A

KODE PROGRAM

1.1 PROGRAM SATU

1.2 PROGRAM DUA

APPENDIX B

GAMBAR-GAMBAR